ΑΤΟΜΙΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΥΣΤΗΜΑ ΔΗΜΙΟΥΡΓΙΑΣ ΔΙΑΤΡΟΦΩΝ ΜΕ ΤΗΝ ΧΡΗΣΗ ΠΕΡΙΟΡΙΣΜΩΝ

Λευκή Κυριάκου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2012

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΣΥΣΤΗΜΑ ΔΗΜΙΟΥΡΓΙΑΣ ΔΙΑΤΡΟΦΩΝ ΜΕ ΤΗΝ ΧΡΗΣΗ ΠΕΡΙΟΡΙΣΜΩΝ

Λευκή Κυριάκου

Επιβλέπουσα Καθηγήτρια

Άννα Φιλίππου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2012

Ευχαριστίες

Ευχαριστώ θερμά την καθηγήτρια μου κ. Άννα Φιλίππου η οποία ήταν η επιβλέπουσα καθηγήτρια της διπλωματικής μου εργασίας. Την ευχαριστώ για τη ευκαιρία που μου έδωσε να εργαστώ κάτω από την επίβλεψη της και για όλη την βοήθεια και την καθοδήγηση που μου παρείχε καθόλη τη διάρκεια της εργασίας αυτής.

Ευχαριστώ επίσης τον κ. Γιάννη Δημόπουλο, ο οποίος υπήρξε ο δεύτερος επιβλέποντας της διπλωματικής μου εργασίας, για όλη την βοήθεια και τις συμβουλές του.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου, για την υπομονή και κατανόηση που έδειξαν τα τελευταία χρόνια που έζησα μακριά τους, για την ολοκλήρωση των σπουδών μου. Επίσης τους ευχαριστώ που με στήριζαν σε κάθε βήμα που έκανα και με συμβούλευαν σε κάθε απόφαση που έπαιρνα.

Περίληψη

Στα πλαίσια της διπλωματικής μου εργασίας επέλεξα να υλοποιήσω ένα σύστημα δημιουργίας διατροφών με εργαλείο αυτόματης διατροφής. Η δημιουργία του συστήματος αυτού εξυπηρετεί στην παροχή βοήθειας σε χρήστες που επιθυμούν να χάσουν, να διατηρήσουν ή να βάλουν βάρος. Το σύστημα θα παρέχει στον χρήστη εργαλείο βέλτιστης διατροφής, το οποίο υλοποιήθηκε με προγραμματισμό ικανοποίησης περιορισμών.

Κύριος στόχος του συστήματος είναι η παροχή βοήθειας στον χρήστη για ρύθμιση των διατροφικών του συνήθειων. Για την επίτευξη αυτού του στόχου αναπτύχθηκαν λειτουργίες για δημιουργία διατροφών οι οποίες θα είναι ειδικά προσαρμοσμένες στις διατροφικές ανάγκες του χρήστη. Ανάμεσα σε αυτές είναι η λειτουργία δημιουργίας αυτόματης διατροφής καθώς επίσης και η δημιουργία διατροφής από τον ίδιο τον χρήστη. Το σύστημα επίσης παρέχει λειτουργίες εισαγωγής νέων φαγητών για χρήση από τους χρήστες για διεύρυνση των επιλογών τους. Τέλος, έγινε επεξεργασία των δεδομένων των φαγητών για εύρεση ρεαλιστικής λύσης στην λειτουργία αυτόματης διατροφής, αφού στην βάση υπάρχουν φαγητά τα οποία δεν είναι βρώσιμα.

Για την ολοκληρωση της διπλωματικής αυτής, ακολουθήθηκαν όλα τα στάδια ανάπτυξης λογισμικού: φάση απαιτήσεων, προδιαγραφών, σχεδιασμού και υλοποίησης. Επίσης περιγράφεται το πρόβλημα για εύρεση βέλτιστης διατροφής, η μοντελοποίηση του και η επίλυση του με προγραμματισμό ικανοποίησης περιορισμών.

Η διαπροσωπεία του συστήματος έγινε μέ την χρήση του NetBeans και το πρόγραμμα βέλτιστης διατροφής σε MiniZinc.

Περιεχόμενα

Κεφάλαιο **1 Εισαγωγή**.....**1**

1.1 Εισαγωγή 1

1.2 Σκοπος του συστήματος 4

1.3 Εργαλεία Ανάπτυξης 5

Κεφαλαιο 2 **Διατροφή και Ανάγκες σε θρεπτικές Ουσίες**.....**6**

2.1 Μεταβολισμός, Δείκτης Μάζας Σώματος και φόρμουλες 6

2.2 Ενέργεια, Θρεπτικά Συστατικά και Ανάγκες 8

2.3 Δημιουργία Διατροφής 13

Κεφάλαιο **3 Ανάλυση Απαιτήσεων** **16**

3.1 Σκοπός Συστήματος 16

3.2 Γενική Περιγραφή Συστήματος 16

3.3 Συγκεκριμένες Απαιτήσεις 20

Κεφάλαιο **4 Προδιαγραφές Συστήματος**.....**31**

4.1 Εισαγωγή 31

4.2 Τεχνική Gane & Sarson 32

4.3 Αντικειμενοστραφής Ανάλυση 63

Κεφάλαιο **5 Σχεδιασμός του Συστήματος**.....**66**

5.1 Εισαγωγή 66

5.2 Αντικειμενοστραφής Σχεδίαση 66

Κεφάλαιο **6 Υλοποίηση Συστήματος** **89**

6.1 Εισαγωγή 89

6.2 Περιγραφή Εργαλείων Ανάπτυξης 89

6.3 Περιγραφή Γλωσσών Προγραμματισμού 91

6.4 Περιγραφή Βάσης Δεδομένων 92

6.5 Προγραμματισμός Ικανοποίησης Περιορισμών 98
6.6 Δυσκολίες Υλοποίησης και Εύρεση Λύσης 104

Κεφάλαιο **7 Συμπεράσματα 108**
7.1 Εισαγωγή 108
7.2 Συμπεράσματα 108
7.3 Μειονεκτήματα Συστήματος 109
7.4 Μελλοντική Εργασία 109

Β ι β λ ι ο γ ρ α φ ί α **111**

Π α ρ ά ρ τ η μ α Α..... **Α-1**

Π α ρ ά ρ τ η μ α Β..... **Β-1**

Π α ρ ά ρ τ η μ α Γ..... **Γ-1**

Π α ρ ά ρ τ η μ α Δ..... **Δ-1**

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγή
1.2 Σκοπος του συστήματος
1.3 Εργαλεία Ανάπτυξης

1.1 Εισαγωγή

Οι γρήγοροι ρυθμοί και η ρουτίνα είναι τα βασικά χαρακτηριστικά της κοινωνίας του 21ου αιώνα. Η αυτοματοποίηση των πάντων έχει κάνει τον σύγχονο άνθρωπο να βασίζεται για όλα στην τεχνολογία και μια από τις συνέπειες είναι η παραμελίση της σωστής διατροφής αφού προτιμά τα εύκολα και γρήγορα γεύματα χωρίς να σκέφτεται τις επιπτώσεις που έχει αυτό στην υγεία του.

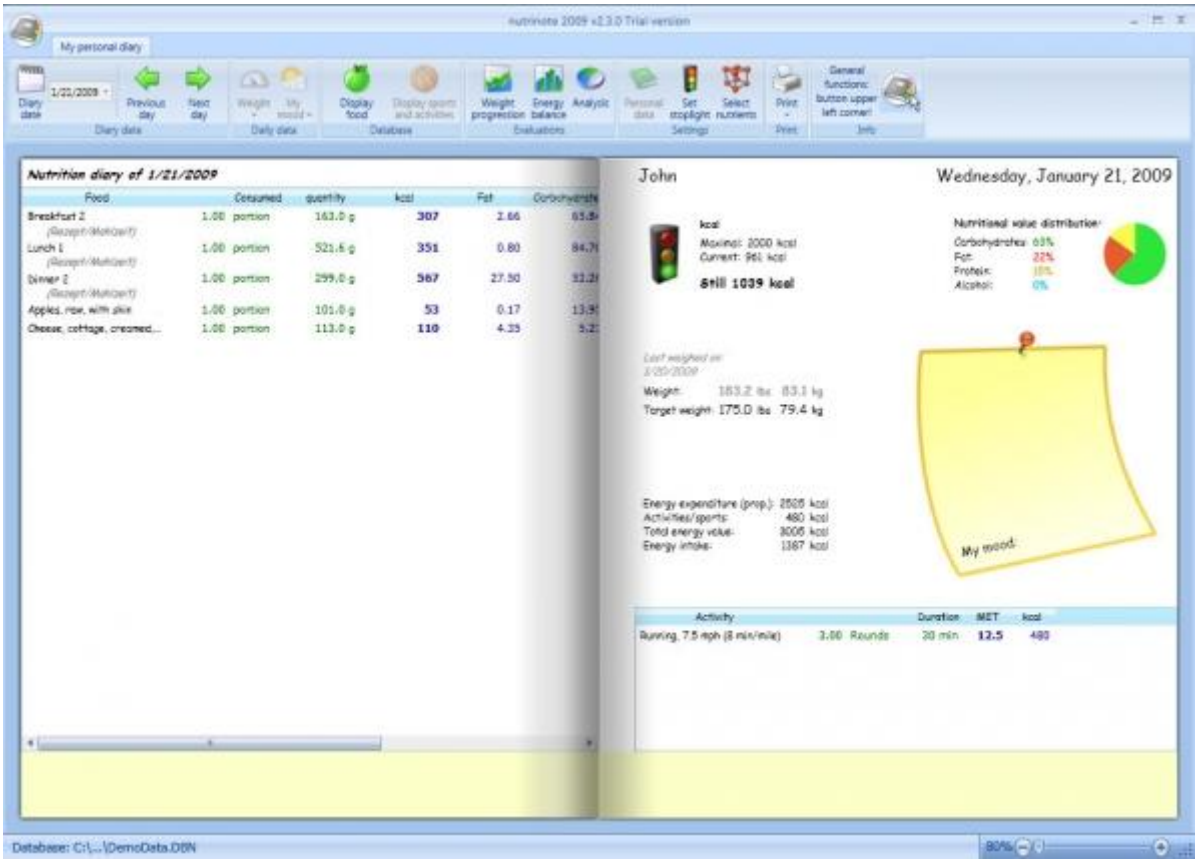
Έτσι αποφάσισα να αναπτύξω ένα χρήσιμο εργαλείο για καταγραφή των γευμάτων που τρώει ένας άνθρωπος σε μια μέρα, στο οποίο υπολογιζονται οι επιτρεπόμενες θερμίδες και άλλα θρεπτικά συστατικά ούτως ώστε να μπορεί να βοηθηθεί ένας χρήστης ο οποίος επιθυμεί να μπεί σε πρόγραμμα σωστής διατροφής.

Διατροφή είναι η παροχή των απαραίτητων στοιχείων σε οργανισμους για την διατήρηση της ζωής. Πιο συγκεκριμένα για τον άνθρωπο είναι ο τρόπος με τον οποίο το φαγητό που τρώει, τρέφει το σώμα του. Καλή διατροφή σημαίνει ότι το σώμα παίρνει όλα τα απαραίτητα συστατικά, βιταμίνες και ορυκτά που χρειάζεται για να έχει την καλύτερη δυνατή απόδοση. Αυτό επιτυχγάνεται με μια υγιεινή δίαιτα. Δίαιτα είναι ο συνδυασμός των τροφών που τρώει ένας άνθρωπος καθημερινά. Στόχος μιας καλής διατροφής είναι η επιτυχής επίτευξη και διατήρηση του ιδανικού βάρους ενός ανθρώπου καθώς επίσης και η βελτίωση των καρδιαγγειακών και άλλων λειτουργιών του ανθρώπινου σώματος, της καλύτερης επούλωσης των πληγών και τραυμάτων, η μείωση του κινδύνου για ασθένειες

όπως καρδιοπάθεια, διαβήτη κλπ, και αύξηση της ενέργειας του σώματος για να παλέψει σε περίπτωση που προσβληθεί από ασθένεια.

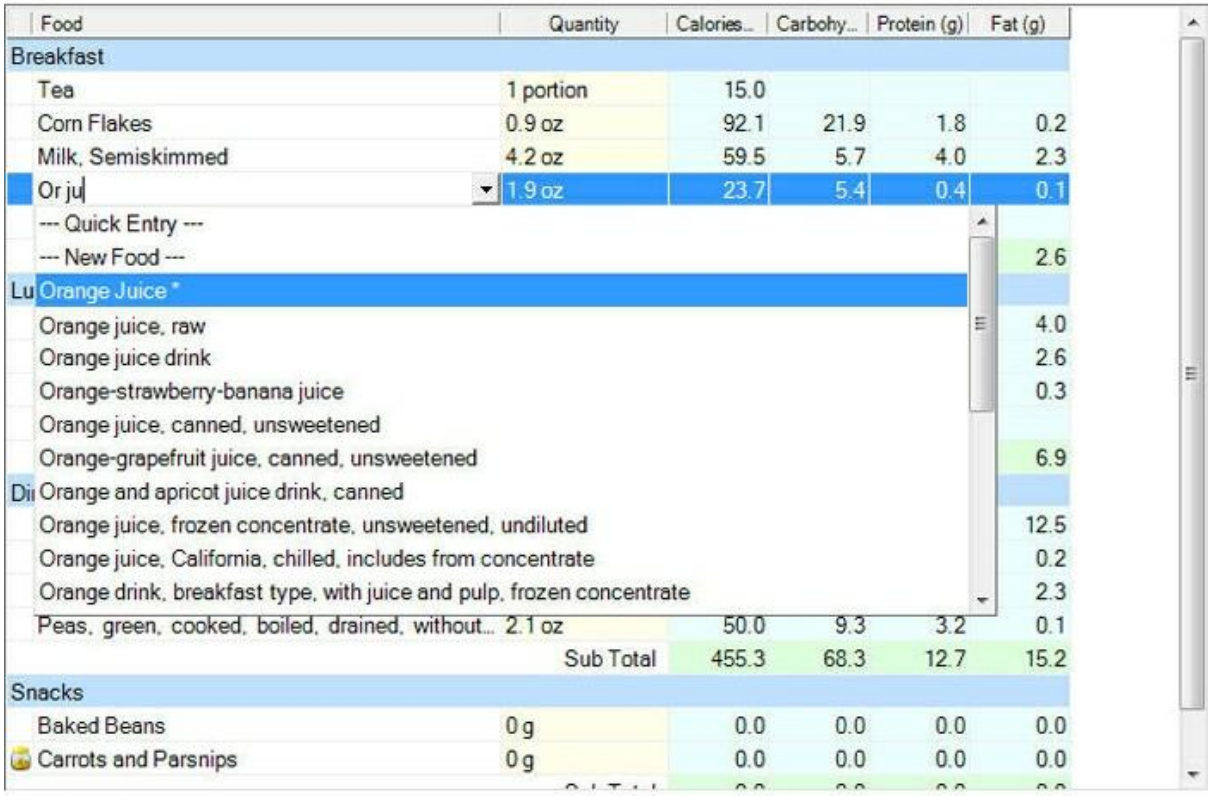
Μια καλή διατροφή χρειάζεται ποικιλία και ισορροπία στις ποσότητες και στα είδη των τροφών καθώς επίσης και φυσική άσκηση. Η παχυσαρκία και η περίσσεια βάρους είναι τα συμπτώματα που προκύπτουν εάν η ισορροπία αυτή δεν υπάρχει. Σε αυτή την περίπτωση δημιουργούνται και άλλα προβλήματα υγείας όπως διαβήτης Β, ψηλή πίεση, ψηλή χολιστερόλη, άσθμα κα αρθρίτιδα.

Υπάρχοντα λογισμικά όπως το NutriNote προσφέρουν στον χρήστη την δυνατότητα δημιουργίας διατροφής σύμφωνα με τις δικές του ανάγκες καθώς επίσης και παρακολούθηση της προόδου του και καταγραφή της διάθεσης του κατά την διάρκεια της περιόδου που γίνεται η καταγραφή της διατροφής του.



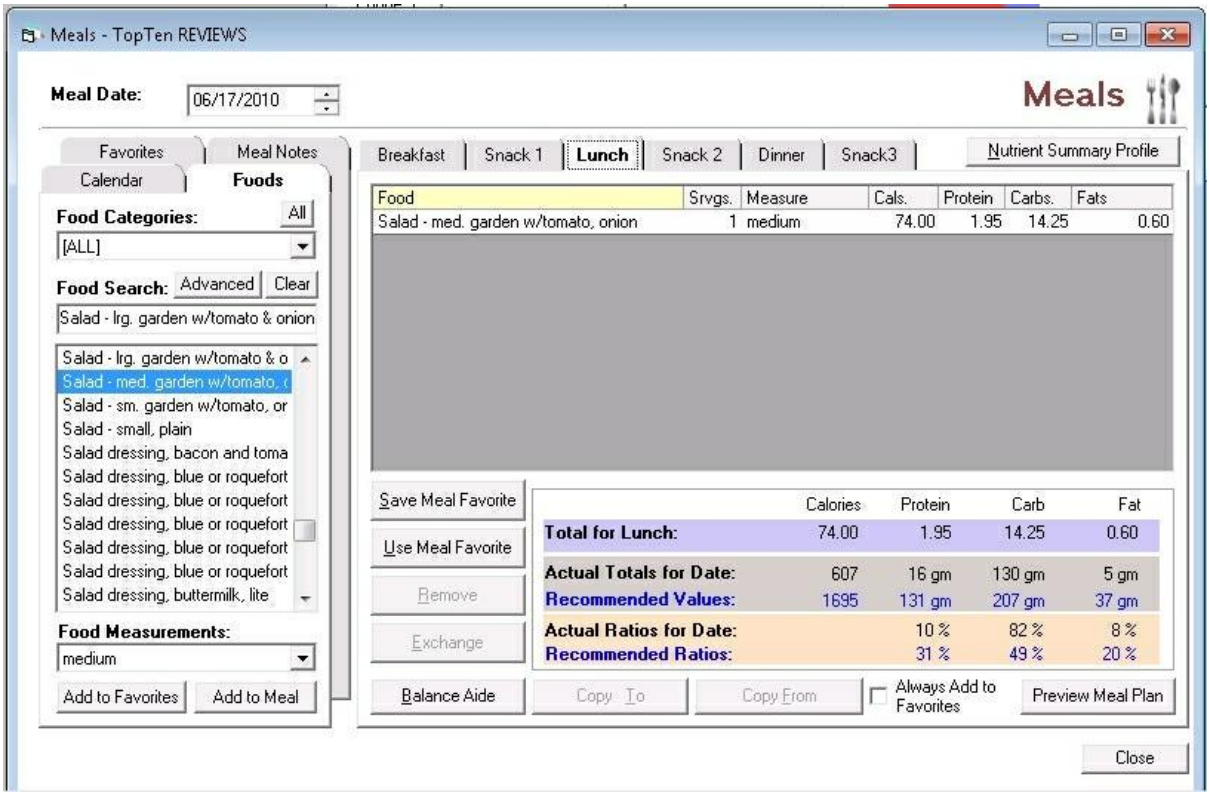
Εικόνα 1.1: Παράδειγμα Διατροφής με την χρήση του λογισμικού NutriNote

Το DietOrganizer είναι επίσης ένα εργαλείο που προσφέρει όλες τις βασικές λειτουργίες ενός εργαλείου δημιουργίας διατροφής. Παρέχει επίσης διαφορετικό χρωματισμό σε περίπτωση που τα φαγητά του χρήστη ξεπερνούν τις απαιτήσεις του σε ενέργεια καθώς επίσης και ειδική αριθμομηχανή για υπολογισμό διαφορετικής ποσότητας από γεύμα σε γεύμα.



Εικόνα 1.2: Δημιουργία Διατροφής με την χρήση του λογισμικού DietOrganizer

Ακόμα ένα εργαλείο που χρησιμοποιείται για την δημιουργία διατροφών είναι το DietMaster 2100. Πρόκειται για ένα εξελειγμένο εργαλείο που υπάρχει το οποίο προσφέρει στον χρήστη διατροφές από ειδικούς διατροφολόγους ειδικά διαμορφωμένες στις ανάγκες συγκεκριμένων ομάδων χρηστών. Επίσης περιλαμβάνει βάσεις δεδομένων για τρόφιμα από την Αυστραλία και τον Καναδά.



Εικόνα 1.3: Δημιουργία Διατροφής με την χρήση του λογισμικού DietMaster 2100

Για τους σκοπούς της διπλωματικής αυτής εργασίας θα αναπτυχθεί ένα κενοτομικό σύστημα το οποίο εξυπηρετεί στην δημιουργία ιδανικών διατροφών για ένα άτομο. Θα παρέχει τις βασικές λειτουργίες για δημιουργία διατροφής από τον ίδιο τον χρήστη με την χρήση βάσης δεδομένων για τρόφιμα από τις Ηνωμένες Πολιτείες Αμερικής. Η κενοτομία του συστήματος αυτού φαίνεται με την δυνατότητα της αυτόματης δημιουργίας μιας διατροφής για τον χρήστη.

1.2 Σκοπός του συστήματος

Σκοπός του συστήματος αυτού είναι η μελέτη των διατροφικών αναγκών ένος ανθρώπου, με βάση τα προσωπικά του χαρακτηριστικά (φύλο, ηλικία, φυσική κατάσταση) και η δημιουργία διατροφής για απώλεια, διατήρηση ή αύξηση του βάρους του. Συγκεκριμένα πρόκειται για υλοποίηση ένος συστήματος το οποίο θα είναι εγκατεστημένο στον υπολογιστή. Ο χρήστης θα δημιουργεί το δικό του προφίλ στο οποίο θα εισάγει τα

ζητούμενα δεδομένα και το σύστημα θα υπολογίζει τα απαραίτητα θρεπτικά συστατικά που πρέπει να παίρνει κάθε μέρα.
Με βάση αυτές τις πληροφορίες, ο χρήστης μπορεί να καταγράφει τις διατροφικές του συνήθειες και να παρακολουθεί κατά πόσο αυτές είναι καλες ή όχι.
Επιπλέον το σύστημα σκοπεύει στην αξιοποίηση των δεδομένων που παρέχει η βάση δεδομένων για ευρεση του καλύτερου συνδιασμού τροφών που να αντιστοιχεί στις ημερίσιες ανάγκες και προτιμήσεις του χρήστη.

Η διπλωματική εργασία αποτελείται από τα εξής μέρη:
α. Τη Βάση Δεδομένων από την οποία ανακτώνται και αποθηκεύονται όλα τα δεδομένα του συστήματος.
β. Τη Διαπροσωπεία του συστήματος η οποία είναι εγκατεστημένη στον υπολογιστή και στην οποία έχουν πρόσβαση όσοι χρήστες επιθυμούν.
γ. Το πρόγραμμα ικανοποίησης Περιορισμών το οποίο εκτελείται μέσω της διαπροσωπείας.

Με την ολοκλήρωση του έργου αυτού αναμένω να αποκτήσω περισσότερη εμπειρία στην ανάπτυξη λογισμικού, καθώς επίσης και στην διεξαγωγή έρευνας σε ένα ειδικευμένο τομέα, στην συγκεκριμένη περίπτωση στον τομέα Διατροφολογίας. Με την διεκπεραίωση της διπλωματικής αυτής περιμένω να αποκομίσω πολλές γνώσεις σε θέματα Διατροφής.

1.3 Εργαλεία Ανάπτυξης

Για την ανάπτυξη κάθε μέλους του συστήματος χρησιμοποιήθηκαν διαφορετικά εργαλεία. Η εισαγωγή και ανάπτυξη της βάσης δεδομένων έγινε με την χρήση του
MySQL Workbench 5.2 CE, η υλοποίηση της διαπρωσωπείας με την χρήση του NetBeans IDE 7.1 σε γλώσσα προγραμματισμού Java και τέλος το πρόγραμμα ικανοποίησης περιορισμών δημιουργήθηκε με την γλώσσα MiniZinc.

Κεφάλαιο 2

Διατροφή και Ανάγκες σε θρεπτικές Ουσίες

2.1 Μεταβολισμός, Δείκτης Μάζας Σώματος και φόρμουλες

2.2 Ενέργεια, Θρεπτικά Συστατικά και Ανάγκες

2.3 Δημιουργία Διατροφής

2.1 Μεταβολισμός, Δείκτης Μάζας Σώματος και φόρμουλες

2.1.1 Μεταβολισμός

Μεταβολισμός (BMR) είναι ο ρυθμός με τον οποίο το σώμα καίει την ημερήσια ενέργεια του σε κατάσταση ηρεμίας, δηλαδή χωρίς την παρουσία άσκησης.
Η ενέργεια αυτή χρησιμοποιείται μόνο για την λειτουργία των βασικών ζωτικών οργάνων, όπως της καρδίας, των πνευμόνων, νευρικού συστήματος, νεφρών, γεννητικών οργάνων, μυών, αρτηριών και του δέρματος.

Ο μεταβολισμός διαφέρει από άτομο σε άτομο λόγω της διαφορετικής φυσικής κατάστασης, ακόμα και αν έχουν την ίδια άλιπη μάζα σώματος.

Με την πάροδο του χρόνου αναπτύχθηκαν διάφορες φόρμουλες για τον υπολογισμό του μεταβολισμού. Η πιο γνωστή είναι αυτή των Harris-Benedict η οποία δημοσιεύτικε το 1919 από το Ινστιτούτο Carnegie στην Washington από τους James Arthur Harris and Francis Gano Benedict.

Η φόρμουλα υπολογισμού του BMR για τους άντρες είναι:

$$P = \left(\frac{13.7516m}{1\ \mathrm{kg}} + \frac{5.0033h}{1\ \mathrm{cm}} - \frac{6.7550a}{1\ \mathrm{year}} + 66.4730 \right) \frac{\mathrm{kcal}}{\mathrm{day}}$$

και για τις γυναίκες:

$$P = \left(\frac{9.5634m}{1\ \text{kg}} + \frac{1.8496h}{1\ \text{cm}} - \frac{4.6756a}{1\ \text{year}} + 655.0955 \right) \frac{\text{kcal}}{\text{day}}$$

όπου P είναι η συνολική ενέργεια ανά ημέρα σε συνθήκες πλήρης ανάπαυσης, m είναι το βάρος, h το ύψος και a η ηλικία.

Το 1990 ο Mifflin et al εισήγαγε την εξής φόρμουλα που θεωρείται πιο αξιόπιστη από αυτή των Harris-Benedict:

$$P = \left(\frac{10.0m}{1\ \text{kg}} + \frac{6.25h}{1\ \text{cm}} - \frac{5.0a}{1\ \text{year}} + s \right) \frac{\text{kcal}}{\text{day}}$$

όπου s είναι +5 για τους άντρες και −161 για τις γυναίκες.
Άλλες φόρμουλες είναι αυτές των Katch-McArdle και Cunningham οι οποίες τείνουν να είναι πιο ακριβείς αφού βασίζονται στο επίπεδο της φυσικής κατάστασης του κάθε ατόμου. Η πιο διαδεδομένη φόρμουλα εξακοθουθεί να είναι η αρχική των Harris-Benedict και για αυτό το λόγο επέλεξα να την χρησιμοποιήσω.

2.1.2 Δείκτης Μάζας Σώματος

Ο Δείκτης Μάζας Σώματος (BMI) είναι ένας ευρετικός δείκτης λίπους του ανθρώπινου σώματος. Δεν δείχνει το ποσοστό του λίπους αλλά καθορίζει κατά πόσο ένας άνθρωπος είναι υπέρβαρος ή όχι ανάλογα με το ύψος του. Η φόρμουλα υπολογισμού του BMI αναπτύχθηκε μεταξύ των 1830 και 1850 από τον Βέλγο Adolphe Quetelet. Για τον υπολογισμό του BMI χρησιμοποιείται το βάρος και το ύψος του ατόμου στο τετράγωνο.

Η φόρμουλα έχει ως εξής:

$$\text{BMI} = \frac{\text{mass(kg)}}{(\text{height(m)})^2}$$

Κατηγορίες BMI:

Κατηγορία	Εμβέλεια BMI
Σε μεγάλο βαθμό Λιποβαρής	Λιγότερο από 15.0
Πολύ Λιποβαρής	15.0 - 16.0
Λιποβαρής	16.0 - 18.5
Υγειής	18.5 - 25.0
Υπέρβαρος	25.0 - 30.0
Μετρίως Παχύσαρκος	30.0 - 35.0
Πολύ Παχύσαρκος	35.0 - 40.0
Σε μεγάλο βαθμό Παχύσαρκος	Περισσότερο από 40.0

Πίνακας 2.1: Κατηγορίες BMI

2.2 Ενέργεια, Θρεπτικά Συστατικά και Ανάγκες

2.2.1 Ενέργεια

Ενέργεια των τροφών είναι η συνολική ενέργεια που παίρνει ένας οργανισμός από τις τροφές και η οποία προέρχεται από την λειτουργία της κυτταρικής αναπνοής. Την μετρούμε σε θερμίδες ή kiloJoules (μονάδα μέτρησης της ενέργειας).

Η ενέργεια προέρχεται από την αντίδραση των θρεπτικών συστατικών με το οξυγόνο των κυττάρων, απο την οποία αντίδραση, απελευθερώνεται ενέργεια. Τα μεγαλύτερα ποσά ενέργειας ανά μάζα έχουν τα λίπη και η αιθανόλη , 9 και 7 kcal/g (38 και 30 kJ/g),αντίστοιχα. Οι πρωτείνες και οι περισσότεροι υδατάνθρακες έχουν 4 kcal/g (17 kJ/g) ενώ οι υδατάνθρακες που δεν απορροφούνται εύκολα όπως οι φιτικές ίνες δεν έχουν μεγάλη ενέργεια.

Η ενέργεια αναγράφεται στις ετικέτες όλων των τροφίμων που κυκλοφορούν στην αγορά σε θερμίδες και kiloJoules. Το 2011 πρωτάθηκε από το American Journal of Public Health να αναγράφεται και ο χρόνος που χρειάζεται για να καταναλωθεί η ενέργεια αυτή από τον οργανισμό. Με αυτό τον τρόπο θα μπορεί να ξέρει κάποιος πόση γυμναστική θα πρέπει να κάνει για να κάψει τις θερμίδες, εαν ξεπεράσει τις θερμίδες της συγκεκριμένης μέρας.

Για τον υπολογισμό των απαραίτητων θερμίδων που πρέπει να καταναλωνει ένας άνθρωπος σε μια μέρα χρησιμοποιείται η φόρμουλα των Harris-Benedict την οποία ανέφερα και στον υπολογισμό του BMR. Οι Harris-Benedict εισήγαγαν κάποιες σταθερές οι οποίες καθορίζουν το επίπεδο άσκησης του κάθε ανθρώπου, οι οποίες εαν πολλαπλασιαστούν με το BMR του ατόμου τότε βρίσκουμε τις αναγκαίες θερμίδες που

πρέπει να παίρνει για να διατηρήσει το τρέχον του βάρος. Την απαραίτητη ενέργεια για διατήρηση του βάρους βλέπουμε στον Πίνακα 2.2.

Οι σταθερές έχουν ως εξής:

Επίπεδο Άσκησης	Ενέργεια=BMR*Σταθερά
Λίγη ή καθόλου άσκηση	Ενέργεια=BMR*1.2
Ελαφριά Άσκηση (1 - 3 μέρες την εβδομάδα)	Ενέργεια=BMR*1.375
Μέτρια Άσκηση (3 -5 μέρες την εβδομάδα)	Ενέργεια=BMR*1.55
Έντονη Άσκηση (6 - 7 μέρες την εβδομάδα)	Ενέργεια=BMR*1.725
Πολύ Έντονη Άσκηση (2 φορές την μέρα)	Ενέργεια=BMR*1.9

Πίνακας 2.2: Σταθερές Harris-Benedict και ενέργεια για Διαρήρηση Βάρους

Σε περίπτωση που ένα άτομο θέλει να χάσει βάρος πρέπει να μειώσει τις θερμίδες που καταναλώνει σε μια μέρα.
3500 θερμίδες αντιστοιχούν σε περίπου 0.5 kg (συγκεκριμένα σε 0.4536 kg). Ο πιο υγιεινός τρόπος να χάσεις βάρος είναι αυτός με τον πιο αργό ρυθμό. Έτσι καλό είναι να χάνει 0.5 kg την εβδομάδα. Για να επιτευχθεί αυτό πρέπει να μειωθούν οι ημερήσιες θερμίδες ανα 500 κάθε μέρα.
Ομοίως, για να αυξήσει κάποιος το βάρος του, πρέπει να αυξήσει τις θερμίδες που καταναλώνει κατά 500 την ημέρα.

2.2.2 Υδατάνθρακες

Οι Υδατάνθρακες βρίσκονται σε διάφορες τροφές όπως το ψωμί, ρύζι, γάλα, πατάτες, μπισκότα, μακαρόνια, αναψυκτικά, καλαμπόκι και γλυκά. Επίπλεον τους βρίσκουμε σε διάφορες μορφές όπως ζάχαρα, φιτικές ίνες και άμυλα.
Πρόκειται για οργανικές ουσίες που αποτελούνται από αλυσίδες μορίων ζάχαρης. Είναι σημαντικοί για το ανθρώπινο σώμα γιατί δίνουν ενέργεια για κάθε λειτουργία του οργανισμού.
Χωρίζονται σε δύο κύριες κατηγορίες, τους απλούς υδατάνθρακες και τους σύνθετους. Στους απλούς ανήκουν η ζάχαρη όπως φρουκτόζη, ζάχαρη σταφυλιού ή καλαμποκιού (γλυκόζη ή δεξτρόζη) και επιτραπέζια ζάχαρη (σακχαρόζη). Στους πιο σύνθετους ανήκουν οι υδατάνθρακες που αποτελούνται από 3 ή περισσότερες αλυσίδες. Σε αυτή την κατηγορία

ανήκουν τα φρούτα (βερίκοκα, τα πορτοκάλια, τα δαμάσκηνα, τα αχλάδια, τα δαμάσκηνα και τα γκρέιπφρουτ.), τα λαχανικά (μπρόκολο, κουνουπίδι, σπανάκι, γογγύλια, μελιντζάνες, γλυκοπατάτες, καλαμπόκι, κρεμμύδια, μαρούλι, σέλινο, αγγούρια, λάχανο, αγκινάρες και σπαράγγια), τα όσπρια (φασόλια, φακές, μπιζέλια, φασόλια garbanzo, μαύρα φασόλια,φασόλια σόγιας και φασόλια Pinto) και τα δημητριακά (ρύζι, βρώμη, κριθάρι, καλαμποκάλευρο, μακαρόνια, πρωινά δημητριακά και ψωμί).

Οι σύνθετοι υδατάνθρακες είναι πιο υγιεινοί από τους απλούς αφού χρειάζεται να διασπαστούν πρώτα σε μικτότερα μόρια για να απορροφηθούν από τον οργανισμό. Αυτό δίνει στον οργανισμό τον απαραίτητο χρόνο να χρησιμοποιήσει την ενέργεια αυτή πριν απορροφηθεί. Οι φυτικές ίνες αποτελούν εξαίρεση αφού είναι συνεδεμένες με τέτοιο τρόπο που δεν διασπώνται και στο τέλος δεν απορροφώνται από τον οργανισμό. Μπορεί να μην παρέχουν ενέργεια στον οργανισμό αλλά είναι χρήσιμες γιατί προσδένονται σε λιπαρές ουσίες και τις απομακρύνουν από τον οργανισμό και βοηθούν στην λειτουργία του εντέρου προτρέποντας την δυσκοιλιότητα.

Οι ανάγκες σε ενέργεια από τους υδατάνθρακες είναι το 57% των συνολικών ημερήσιων θερμίδων. Η ενεργειακή απόδοση για τους υδατάνθρακες είναι 4 θερμίδες ανά γραμμάριο. Έτσι έχουμε την εξής φόρμουλα για υπολογισμό της ημερήσιας ποσότητας υδατανθράκων που θα πρέπει να καταναλώνει ένας άνθρωπος.

$$\text{Υδατάνθρακες} = (\text{Ημερήσιες\_Θερμίδες} * 57\%) \div 4$$

2.2.3 Λίπη

Τα Λίπη είναι ουσίες που αποτελούνται από τριγλυκερίδια, τριεστέρες της γλυκερίνης και διάφορα λιπαρά οξέα. Τα βρίσκουμε σε στερεά ή υγρή μορφή όπως τα λάδια. Είναι σημαντικά για πολλές μορφής ζωής, αφού εξυπηρετούν σε μεταβολικές λειτουργίες, όπως την μεταφορά βιταμινών, διατηρούν την θερμοκρασία του σώματος σταθερή και συμβάλλουν στην κατασκευή των κυττάρων, καθιστώντας τα έτσι σημαντικά στην δίαιτα οποιουδήποτε ανθρώπου.

Τα λίπη χωρίζονται σε δυο μεγάλες κατηγορίες, τα ζωικά και τα φυτικά. Τα ζωικά τα παίρνουμε από το γάλα και το κρέας ή κάτω από το δέρμα του ζώου. Μερικά

παραδείγματα καταναλώσιμων ζωικών λιπών είναι το ιχθυέλαιο, ο βούτηρος και το λίπος φάλαινας. ενώ φυτικών είναι το φυστικέλαιο, σογιέλαιο, ηλιανθέλαιο, ελαιόλαδο και το κακαοβούτηρο.
Όλα αυτά τα λίπη μπορούν να κατηγοριοποιηθούν σε κορεσμένα και ακόρεστα.

Κορεσμένα είναι τα λίπη που στις αλυσίδες τους έχουν μόνο κορεσμένα οξέα και δεν έχουν διπλούς δεσμους μεταξύ των ατόμων του άνθρακα της αλυσίδας των λιπαρών οξέων ενώ τα **ακόρεστα** είναι αυτά που έχουν τουλάχιστον ένα διπλό δεσμό μεταξύ των ατόμων του άνθρακα. Τα ακόρεστα περιλαμβάνουν μικρότερη ενέργεια από ότι τα κορεσμένα και έτσι λιγότερες θερμίδες καθιστώντας τα προτιμότερα από τα κορεσμένα.

Οι ανάγκες σε ενέργεια από τα ακόρεστα λίπη είναι το 30% των συνολικών ημερήσιων θερμίδων. Η ενεργειακή απόδοση για τα λίπη είναι 9 θερμίδες ανά γραμμάριο. Έτσι έχουμε την εξής φόρμουλα για υπολογισμό της ημερήσιας ποσότητας λιππου θα πρέπει να καταναλώνει ένας άνθρωπος.

$$\text{Λίπη}=(\text{Ημερήσιες\_Θερμίδες}* 30\%)\div 9$$

2.2.4 Πρωτείνες

Οι πρωτείνες είναι βασικές θρεπτικές ουσίες για το ανθρώπινο σώμα αφού πρόκειται για δομικά στοιχεία του οργανισμού και πηγή ενέργειας. Είναι οι ουσίες δηλαδή που δημιουργούν, επισκευάζουν και διατηρούν τους ιστούς του σώματος. Αποτελούνται από αμινοξέα και ενώνονται μεταξύ τους με πεπτιδικές αλυσίδες. Τα αμινοξέα βρίσκονται σε τροφές όπως το κρέας, το ψάρι, το γάλα, τα αβγά, που ανήκουν στο ζωικό βασίλειο αλλά και σε φυτικές τροφές όπως δημητριακά ολικής αλέσεως, όσπρια και ξηρούς καρπούς.
Οι ανάγκες σε ενέργεια από τις πρωτείνες είναι το 13% των συνολικών ημερήσιων θερμίδων. Η ενεργειακή απόδοση για τις πρωτείνες είναι 4 θερμίδες ανά γραμμάριο. Έτσι έχουμε την εξής φόρμουλα για υπολογισμό της ημερήσιας ποσότητας πρωτείνων που θα πρέπει να καταναλώνει ένας άνθρωπος.

Πρωτείνες=(Ημερήσιες_Θερμίδες* 13%)÷ 4

2.2.5 Ορυκτά

Τα ορυκτά είναι χημικά στοιχεία τα οποία χρειάζονται στους ζωντανούς οργανισμούς εκτός από τα τέσσερα βασικά που είναι το οξυγόνο, το υδρογόνο, το διοξύδιο του άνθρακα και το άζωτο. Οι ανάγκες σε ορυκτά σε έναν οργανισμό είναι μικρές σε σχέση με τις πρωτεϊνες, τα λίπη και τους υδατάνθρακες. Μερικά ορυκτά είναι το ασβέστιο, το μαγνήσιο, το νάτριο, το κάλιο, ο σίδηρος, το σελίνιο, ο ψευδάργυρος και το ιώδιο.
Στον πίνακα 2.2 αναγράφονται οι εκτιμώμενες ανάγκες για τα ορυκτά σε mg/d σύμφωνα με έρευνες που έγιναν στο Ηνωμένο Βασίλειο.

Ηλικία	Ασβέστιο		Μαγνήσιο		Νάτριο		Κάλιο		Σίδηρος	
	Άντρες	Γυναίκες	Άντρες	Γυναίκες	Άντρες	Γυναίκες	Άντρες	Γυναίκες	Άντρες	Γυναίκες
0-3 μήνες	525		55		210		800		1.7	
4-6 μήνες	525		60		280		850		4.3	
7-9 μήνες	525		75		320		700		7.8	
10-12 μήνες	525		80		350		700		7.8	
1-3 χρόνια	350		85		500		800		6.9	
4-6 χρόνια	450		120		700		1100		6.1	
7-10 χρόνια	550		200		1200		2000		8.7	
11-14 χρόνια	1000	800	280	280	1600	1600	3100	3100	11.3	14.5
15-18 χρόνια	1000	800	300	300	1600	1600	3500	3500	11.3	14.5
19-50 χρόνια	700	700	300	270	1600	1600	3500	3500	8.7	14.5
50+ χρόνια	700	700	300	270	1600	1600	3500	3500	8.7	8.7

Πίνακας 2.3: Ανάγκες σε Ορυκτά σε mg/d

2.2.6 Βιταμίνες

Οι βιταμίνες είναι οργανικές ουσίες που είναι απαραίτητες σε μικρές ποσότητες σε έναν οργανισμό. Πρόκειται για ουσίες οι οποίες δεν μπορούν να δημιουργηθούν στον οργανισμό και πρέπει να τις πάρουμε από την δίαιτα.

Στον πίνακα 2.3 αναγράφονται οι εκτιμώμενες ανάγκες για τις βιταμίνες σε mg/d σύμφωνα με έρευνες που έγιναν στο Ηνωμένο Βασίλειο.

Ηλικία	Βιταμίνη Α		Βιταμίνη C	
	Άντρες	Γυναίκες	Άντρες	Γυναίκες
0-3 μήνες	350		25	
4-6 μήνες	350		25	
7-9 μήνες	350		25	
10-12 μήνες	350		25	
1-3 χρόνια	400		30	
4-6 χρόνια	400		30	
7-10 χρόνια	500		30	
11-14 χρόνια	600	600	35	35
15-18 χρόνια	700	600	40	40
19-50 χρόνια	700	600	40	40
50+ χρόνια	700	600	40	40

Πίνακας 2.4: Ανάγκες σε Βιταμίνες σε mg/d

2.3 Δημιουργία Διατροφής

Μια υγιεινή διατροφή έχει στόχο την επίτευξη της υγείας ενός ανθρώπου. Για να θεωρηθεί μια διατροφή υγιεινή πρέπει να πληρεί κάποιες προϋποθέσεις. Αρχικά πρέπει να περιλαμβάνει την κατανάλωση των απαραίτητων θρεπτικών συστατκών και νερού στις κατάλληλες ποσότητες. Επίσης πρέπει να διατηρεί μια ισορροπία μεταξύ των υδατανθράκων, λιπών, πρωτεϊνών και ενέργειας καθώς επίσης και των μικρό-θρεπτικών συστατικών όπως τα ορυκτά και οι βιταμίνες.

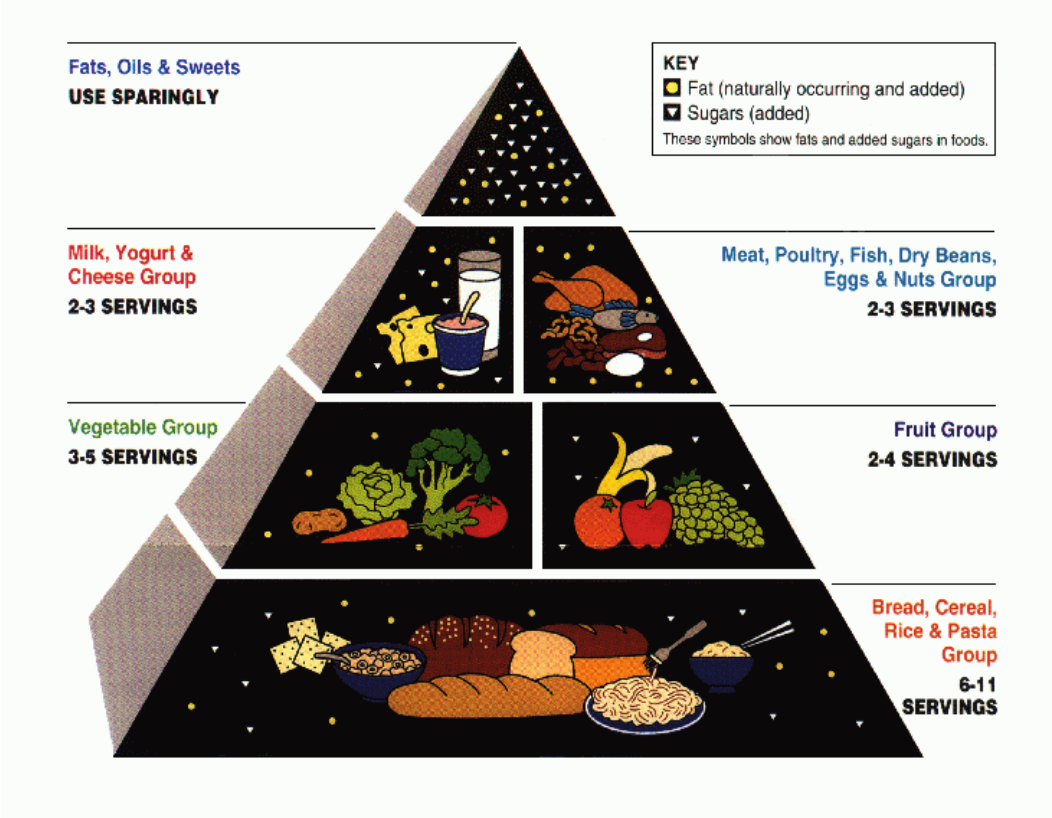
Τα θρεπτικά συστατικά υπάρχουν σε πολλά και διάφορα είδη τροφίμων, έτσι υπάρχουν πολλές διατροφές που μπορούν να θεωρηθούν υγιεινές.

Σε μια διατροφή πρέπει να υπάρχουν μικρά και συχνά γεύματα. Δεν πρέπει να παραλείπεται ποτέ το πρόγευμα, το μεσημεριανό ή το δείπνο και πρέπει να υπάρχουν και ενδιάμεσα γεύματα. Ένα το πρωί και δύο το απόγευμα. Ο σκοπός των ενδιάμεσων

γευμάτων είναι στο να διατηρούν το στομάχι γεμάτο ούτως ώστε να μην γίνεται κατάχρηση άλλων τροφών.

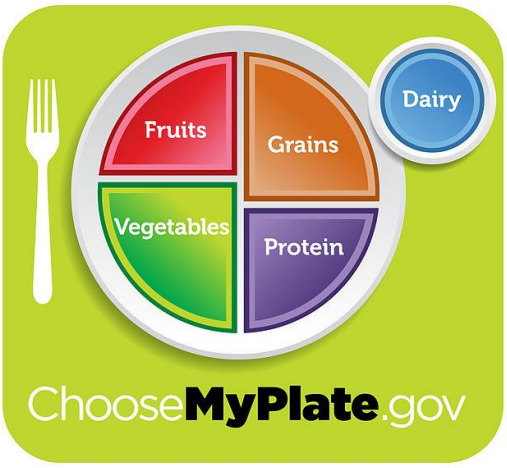
Σε κάθε γεύμα πρέπει να περιλαμβάνονται τροφές από κάθε ομάδα τροφίμων. Οι ομάδες καθορίζονται στην πυραμίδα τροφίμων η οποία αναπτύχθηκε από Υπουργείο Γεωργίας και Υγείας των ΗΠΑ (USDA) το 1992 και είναι οι εξής:

- Λίπη, Έλαια και Γλυκά
- Κρέας Πουλερικά, Ψάρια, όσπρια, Γαλακτοκομικά
- Φρούτα και Λαχανικά
- Ψωμί, Ρύζι, Ζυμαρικά, Δημητριακά και Αμυλούχα Λαχανικά



Εικόνα 2.1: Πυραμίδα Τροφίμων USDA 1992

Το 2011 το υπουργείο Γεωργείας και Υγείας δημιούργησε το MyPlate το οποίο αντικατέστησε την Πυραμίδα Τροφίμων στην διαγραμματική αναπαράσταση μιας υγιεινής διατροφής.



Εικόνα 2.2 MyPlate USDA 2011

Κεφάλαιο 3

Ανάλυση Απαιτήσεων

3.1 Σκοπός Συστήματος
3.2 Γενική Περιγραφή Συστήματος
3.3 Συγκεκριμένες Απαιτήσεις

3.1 Σκοπός Συστήματος

Το λογισμικού που θα αναπτυχθεί θα χρησιμοποιείται για την δημιουργία και καταγραφή διατροφών. Την ευκαρία για χρήση του λογισμικού θα έχουν διάφοροι χρήστες οι οποίοι θα έχουν αγοράσει το λογισμικό γιατί θέλουν να ελέγχουν το τι τρώνε. Στις μέρες μας αυτό είναι χρήσιμο αφου οι ρυθμοί της καθημερινότητας μας δεν μας επιτρέπουν να σκεφτόμαστε το τι τρώμε και παρεκτρεπόμαστε.

Το λογισμικό αυτό θα είναι εγκατεστημένο στον υπολογιστή και θα αλληλεπιδρά με μια βάση δεδομένων από την οποία θα παίρνει τις πληροφορίες για τα φαγητά και θα αποθηκεύει τα στοιχεία του χρήστη.

3.2 Γενική Περιγραφή Συστήματος

3.2.1 Προοπτική Συστήματος

Το λογισμικό που θα αναπτυχθεί θα αποτελείται από μια διεπαφή η οποία θα επικοινωνεί με την βάση δεδομένων για να εκτελεί τις λειτουργίες του. Στην διεπαφή θα έχουν πρόσβαση όσοι χρήστες δημιουργήσουν προφίλ με τα προσωπικά τους στοιχεία, αλλά και χρήστες οι οποίοι δεν έχουν δημιουργήσει προφίλ. Αυτοί οι χρήστες θα μπορούν να προσθέσουν στοιχεία στην βάση δεδομένων και να εκτελούν τις λειτουργίες του συστήματος οι οποίες περιγράφονται πιο κάτω.

3.2.2 Λειτουργίες Συστήματος

Το σύστημα που θα αναπτυχθεί θα υποστηρίζει τις ακόλουθες λειτουργίες.

3.2.2.1 Δημιουργία Νεου Χρηστη

Ο χρήστης θα δημιουργεί το προφίλ του. Θα εισάγει τα προσωπικά του στοιχεία τα οποία θα αποθηκεύονται στην βάση. Θα πρέπει να εισάγει όλα τα απαραίτητα στοιχεία για να υπολογίζονται οι ανάγκες του σε θερμίδες, υδατάνθρακες, λίπη και πρωτείνες.

3.2.2.2 Log In

Ο χρήστης θα μπορεί να κάνει log in στο προφίλ του εισάγωντας ένα username και password και να χρησιμοποιήσει τις πληροφορίες που έχει ήδη εισάγει στο σύστημα.

3.2.2.3 Δημιουργία Νέας Ημερίσιας Διατροφής

Ο χρήστης μπορεί να ψάξει στην βάση για φαγητά και να τα προσθέσει στο ημερήσιο διαιτολόγιο του, το οποίο μπορεί να αποθηκεύσει.

3.2.2.4 Αυτόματη Δημιουργία Διατροφής

Ο χρήστης μπορεί να θέσει κάποιες προτιμήσεις σύμφωνα με τις οποίες γίνεται η δημιουργία ενός αυτόματου διαιτολογίου.

3.2.2.5 Εισαγωγή Νέου φαγητού

Ο χρήστης μπορεί να προσθέσει στην βάση ένα νέο φαγητό. Θα μπορεί να εισάγει και την θρεπτική του αξία όπως αυτή αναγράφεται στην συσκευασία.

3.2.2.6 Εισαγωγή Φαγητού από διάφορα Υλικά

Ο χρήστης θα μπορεί να προσθέσει στην βάση ένα φαγητό το οποίο αποτελείται από πολλά άλλα συστατικά τα οποία είναι ήδη καταχωρημένα στην βάση δεδομένων.

3.2.2.7 Προβολή Προφίλ Χρήστη

Ο χρήστης θα μπορεί να δεί το προφίλ του.

3.2.2.8 Διόρθωση Προφίλ Χρήστη

Ο χρήστης θα μπορεί να αλλάξει οποιαδήποτε πληροφορία εισήγαγε στο προφίλ του και να το ανανεώσει.

3.2.2.9 Διαγραφή Χρήστη

Ο χρήστης θα μπορεί να διαγραψει το προφίλ του.

3.2.2.10 Προβολή Διατροφών Χρήστη

Ο χρήστης θα μπορεί να δεί όλες τις διατροφές που έχει δημιουργήσει.

3.2.2.11 Διόρθωση Διατροφής

Ο χρήστης θα μπορεί να αλλάξει οποιαδήποτε διατροφή έχει ήδη δημιουργήσει.

3.2.2.12 Διαγραφή Διατροφής

Ο χρήστης θα μπορεί να διαγράψει οποιαδήποτε διατροφή του.

3.2.2.13 Αλλαγή Κωδικού Πρόσβασης

Ο χρήστης θα μπορεί να αλλάξει το password του.

3.2.2.14 Log Out

'Εξοδος του χρήστη από το σύστημα.

3.2.3 Χαρακτηριστικά Χρηστών

Το σύστημα αυτό στοχεύει σε χρήστες όλων των ηλικιών οι οποίοι θέλουν να βελτιώσουν τις διατροφικές τους συνήθειες. Πρόκειται για χρήστες οι οποίοι θέλουν να διατηρήσουν το τρέχον τους βάρος, να χάσουν ή να πάρουν βάρος ανάλογα με το BMI τους. Ο κάθε χρήστης θα έχει κωδικό πρόσβασης για να βλέπει και να επεξεργάζεται το προφίλ του στο οποίο θα έχει αποθηκευμένα προσωπικά του στοιχεία.
Χρήστες οι οποίοι δεν έχουν δημιουργήσει προφίλ θα έχουν πρόσβαση μόνο στις λειτουργίες εισαγωγής τροφίμων στην βάση αφού αυτές οι λειτουργίες δεν χρειάζονται προσωπικά δεδομένα από τον χρήστη.

3.2.4 Περιορισμοί, Παραδοχές και Εξαρτήσεις

3.3.4.1. Περιορισμοί Μνήμης

Το σύστημα απαιτεί μνήμη μέχρι 1 GB για την εγκατάσταση του λογισμικού και της Βάσης Δεδομένων.

3.3.4.2 Περιορισμοί Λογισμικού

Το σύστημα πρέπει να λαμβάνει υπόψη τους εξής περιορισμούς:

- Πρέπει να είναι Χρήσιμο για τους Χρήστες. Για να γίνει αυτό πρέπει το σύστημα να ικανοποιεί όλες τις ανάγκες του χρήστη.Επίσης πρέπει να είναι φιλικό προς τον χρήστη και η χρήση του να είναι όσο πιο απλή γίνεται.
- Πρέπει να είναι Αξιόπιστο Σύστημα το οποίο όταν πάρει μια είσοδο να βγάλει αποτέλεσμα και όταν η είσοδος δεν είναι ορθή να αποκριθεί αντίστοιχα.

- Πρέπει οι λειτουργίες να έχουν την βέλτιστη απόδοση και ο χρήστης να μην περιμένει πολύ ώρα μέχρι να δεί κάποιο αποτέλεσμα.

3.3 Συγκεκριμένες Απαιτήσεις

3.3.1 Εξωτερικές Απαιτήσεις Διεπαφής

Το σύστημα θα παίρνει όλες τις πληροφορίες που χρειάζεται για να εκτελέσει τις λειτουργίες του από την βάση δεδομένων με την οποία είναι συνεδεμένο. Επίσης το σύστημα θα χρησιμοποιεί ένα batch file για να τρέχει το πρόγραμμα ικανοποίησης περιορισμών το οποίο θα δημιουργείται μέσα στο σύστημα.

3.3.2 Απαιτήσεις Διεπαφής Χρήστη

Η διεπιφάνεια του χρήστη πρέπει να είναι όσο το δυνατόν πιο καλά σχεδιασμένη για να είναι ευκολη η χρήση της από τον χρήστη. Πρέπει χρήστες με διαφορετικό επίπεδο γνώσης της χρήσης λογισμικού να είναι ικανοί να την χειριστούν και να μην χρειάζεται να θυμούνται τα βήματα χρήσης για κάθε λειτουργία που θέλουν να εκτελέσουν.

3.3.3 Απαιτήσεις Απόδοσης

3.3.3.1 Στατικές Αριθμητικές Απαιτήσεις:

Τερματικά: Χρειάζεται μόνο ένα τερματικό αφού το σύστημα προορίζεται για εγκατάσταση στον υπολογιστή.

Αριθμός Ταυτόχρονων Χρηστών: Μόνο ένας χρήστης μπορεί να χρησιμοποιεί το σύστημα μια συγκεκριμένη χρονική στιγμή.

3.3.3.2 Δυναμικές Αριθμητικές Απαιτήσεις:

Το σύστημα θα είναι όσο πιο γρήγορο γίνεται. Περισσότερη ανάγκη για γρήγορη ανταπόκριση έχει το σύστημα στην δημιουργία αυτόματης διατροφής κατα την οποία θα γίνονται πολλαπλές προσβάσεις στην βάση και πολλαπλές κλήσεις του προγράμματος ικανοποίησης περιορισμών μέχρι να βρεθεί λύση.
Οι υπόλοιπες λειτουργίες του συστήματος δεν πρέπει να καθυστερούν περισσότερο από 2 δευτερόλεπτα.

3.3.4 Περιορισμοί Σχεδιασμού

- Το σύστημα και η βάση πρέπει να υποστηρίζονται σε Windows.
- Για λόγους ασφάλειας και προστασίας των προσωπικών δεδομένων του χρήστη, οι λειτουργίες πρέπει να είναι προσβάσιμες με την εισαγωγή ενός Username και Password για κάθε χρήστη.
- Το σύστημα θα υλοποιηθεί με την χρήση του NetBeans.
- Το σύστημα θα κάνει χρήση της MySQL.
- Το σύστημα θα βγάζει μηνύματα λάθους για κάθε ανεπιτυχή ενέργεια.

3.3.5 Λογικές Απαιτήσεις Βάσης Δεδομένων

Το σύστημα θα χρησιμοποιεί την βάση δεδομένων του Υπουργείου Γεωργίας και Υγείας των ΗΠΑ (USDA) η οποία θα εγκαθείσταται στον υπολογιστή μαζί με το σύστημα. Επίσης θα χρησιμοποιεί μια άλλη βάση για αποθήκευση των δεδομένων του χρήστη. Τα δεδομένα αυτά θα φυλάγονται σε πίνακες και ο χρήστης θα μπορεί να αποθηκεύει, να ανακτά, να ανανεώνει και να διαγράφει τα δεδομένα αυτά εαν είναι εξουσιοδοτημένος να το κάνει.

3.3.6 Χαρακτηριστικά του Συστήματος

3.3.6.1 Δημιουργία Νεου Χρηστη

Σκοπός:

Η δημιουργία νέου προφίλ για κάποιον χρήστη. Η εισαγωγή δηλαδή στην βάση όλων των απαραλιτητων πληροφοριών για να μπορεί να κάνει χρήση των διατροφικών λειτουριών του συστήματος.

Είσοδος:

Ο χρήστης εισάγει το Username και Password του, το ονοματεπώνυμο του, την ηλικία του, το φύλο, το βάρος και το ύψος του. Επίσης επιλέγει το επίπεδο της Άσκησης που έχει και τον στόχο του να χάσει, να βάλει ή να διατηρήσει το βάρος του. Έισάγει επίσης το βάρος στο οποίο θα ήθελε να φτάσει.

Επεξεργασία:

Για να χρησιμοποιείσει τις διάφορες διατροφικές λειτουργίες του συστήματος ο χρήστης πρέπει να δημιουργήσει ένα προφίλ με τα στοιχεία του. Με την εισαγωγή των στοιχείων γίνεται ο υπολογισμός του BMI, του BMR του χρήστη και οι θερμίδες που πρέπει να καταναλώνει σε μια μέρα για να επιτέυξει τον στόχο που έχει επιλέξει.

Έξοδος:

Έαν τα στοιχεία που έχει εισάγει ο χρήστης είναι σωστά, η λειτουργία αυτή δείχνει στον χρήστη την κατηγορία στην οποία ανήκει σύμφωνα με το BMI του, καθώς επίσης και τις θερμίδες που πρέπει να καταναλώνει σε μια μέρα. Τα στοιχεία του χρήστη αποθηκεύονται στην βάση δεδομένων για μελλοντική χρήση. Στο τέλος ανοίγει η φόρμα με το προφίλ του χρήστη. Ο χρήστης μπορεί να ακυρώσει ολόκληρη την διαδικασία εαν το επιθυμεί.

3.3.6.2 Log In

Σκοπός:

Καθορισμός μέσω ενός Username και Password κατά πόσο ένας χρήστης μπορεί να έχει πρόσβαση σε συγκεκριμένες πληροφορίες του συστήματος. Το Username και Password επιλέγονται από τον χρήστη στα πλαίσια της λειτουργίας του συστήματος Δημιουργία Νεου Χρήστη.

Είσοδος:

Ο χρήστης εισάγει το Username και Password του.

Επεξεργασία:

Για όλες τις βασικές λειτουργίες του συστήματος χρειάζεται ο χρήστης να είναι logged in. Έτσι πριν την εμφάνιση του παραθύρου για εκτέλεση μιας λειτουργίας ανοίγει το παράθυρο για να κάνει log in ο χρήστης. Πρέπει δηλαδή να εισάγει δύο συμβολοσειρές στα textboxes του παραθύρου. Το Password εμφανίζεται αποκρυπτογραφημένο με αστερίσκους.

Έξοδος:

Εαν τα στοιχεία είναι σωστά τότε οι χρήστης παραπέμπεται στην λειτουργία που είχε αρχικά επιλέξει. Εαν όχι, εμφανίζεται το κατάλληλο μήνυμα και μπορεί να ξαναπροσπαθήσει εισάγωντας ξανά τα απαιτούμενα στοιχεία. Ο χρήστης μπορεί να ακυρώσει ολόκληρη την λειτουργία εαν το επιθυμεί.

3.3.6.3 Δημιουργία Νέας Ημερίσιας Διατροφής

Σκοπός:

Η δημιουργία μιας διατροφής για μια μέρα η οποία διατροφή θα περιέχει τροφές που να ικανοποιούν τις διατροφικές ανάγκες του χρήστη.

Είσοδος:

Ο χρήστης κάνει χρήση της λειτουργίας Search για να βρεί φαγητά από την βάση δεδομένων και να τα προσθέσει στα γεύματα του.

Επεξεργασία:

Ο χρήστης αφου βρεί το φαγητό που επιθυμεί επιλέγει την ποσότητα και σε ποιο γεύμα θέλει να το προσθέσει και το εισάγει.

Εαν τα θρεπτικά συστατικά και η ενέργεια είναι μέσα στα πλαίσια τότε, μπορεί να επιλέξει και άλλα φαγητά. Έαν όχι εμφανίζεται ανάλογο μήνυμα που τον προειδοποιεί ότι δεν ακολουθεί μια ισορροπημένη διατροφή.

Έξοδος:

Η δημιουργία μιας ισορροπημένης διατροφής, η οποία είναι κατάλληλη για τον συγκεκριμένο χρήστη. Ο χρήστης μπορεί να αποθηκεύσει την διατροφή αυτή στην βάση δεδομένων ή να ακυρώσει ολόκληρη την διαδικασία εαν το επιθυμεί.

3.3.6.4 Αυτόματη Δημιουργία Διατροφής

Σκοπός:

Η αυτόματη δημιουργία μιας διατροφής για μια μέρα η οποία διατροφή θα περιέχει τροφές που να ικανοποιούν τις διατροφικές ανάγκες του χρήστη.

Είσοδος:

Ο χρήστης επιλέγει τα είδη των γευμάτων που θα ήθελε να έχει στην διατροφή του. Κάνει επιλογές για το πρόγευμα, το μεσημεριανό, το δείπνο και για τρία ενδιάμεσα γεύματα.

Επεξεργασία:

Ανάλογα με τις επιλογές του χρήστη, το σύστημα αρχικοποιεί κάποιες μεταβλητές από την βάση και δημιουργεί ένα πρόγραμμα ικανοποίησης περιορισμών σε γλώσσα προγραμματισμού MiniZinc, το οποίο βρίσκει μια λύση στο πρόβλημα. Περισσότερα για αυτή την λειτουργία θα επεξηγηθούν στο Κεφάλαιο 5.

Έξοδος:

Μια διατροφή που ικανοποιεί τις ανάγκες του χρήστη σε θρεπτικές ουσίες και ενέργεια.

Ο χρήστης μπορεί να ακυρώσει ολόκληρη την διαδικασία εαν το επιθυμεί.

Η διατροφή παρουσιάζεται στην φόρμα διατροφής και ο χρήστης μπορεί να την αποθηκεύσει στην βάση. Ο χρήστης μπορεί επίσης να κάνει οποιεσδήποτε αλλαγές επιθυμεί πριν την αποθηκεύσει ή να ακυρώσει ολόκληρη την διαδικασία εαν το επιθυμεί.

3.3.6.5 Εισαγωγή Νέου φαγητού

Σκοπός:

Η αύξηση των δεδομένων που έχει η βάση. Μπορεί να γίνει εισαγωγή προϊόντων που είναι μοναδικά για την χώρα από την οποία προέρχεται ο χρήστης.

Είσοδος:

Ο χρήστης εισάγει το όνομα του προϊόντος, την περιγραφή του, την κατηγορία στην οποία ανήκει και την θρεπτική του αξία όπως αυτή αναγράφεται στην συσκευασία.

Επεξεργασία:

Γίνεται επαλήθευση ότι όλα τα δεδομένα εισάχθηκαν σωστα και ότι τα απαραίτητα στοιχεία δεν λείπουν.

Έξοδος:

Εαν όλα τα στοιχεία συμπληρώθηκαν σωστά γίνεται η εισαγωγή του φαγητού στην βάση. Εαν όχι τότε εμφανίζεται το ανάλογο μήνυμα. Ο χρήστης μπορεί να ακυρώσει ολόκληρη την διαδικασία εαν το επιθυμεί.

3.3.6.6 Εισαγωγή Φαγητού από διάφορα Υλικά

Σκοπός:

Η αύξηση των δεδομένων που έχει η βάση. Μπορεί να γίνει εισαγωγή φαγητών που αποτελούνται από διάφορα υλικά όπως σπιτικά φαγητά, των οποίων η θρεπτική αξία εξαρτάται από την ποσότητα των υλικών τα οποία περιλαμβάνουν.

Είσοδος:

Ο χρήστης επιλέγει με την χρήση του Search τα διάφορα υλικά και τις ποσότητες τους και τα εισάγει στην λιστα με τα υλικά. Εισάγει επίσης ένα όνομα και περιγραφές για το νέο αυτό φαγητό καθώς επίσης και τον αριθμό των μερίδων.

Επεξεργασία:

Από τα υλικά υπολογίζονται οι συνολικές θερμίδες της συνταγής αυτής, και τα θρεπτικά συστατικά (υδατάνθρακες, πρωτεϊνες και λίπη) ανά μερίδα και ανά 100 grams της ολικής συναγής.

Έξοδος:

Εάν τα στοιχεία συμπληρωθούν όλα σωστά, γίνεται αποθήκευση στην βάση της συνταγής αυτής. Εαν όχι εμφανίζεται το ανάλογο μήνυμα. Ο χρήστης μπορεί να ακυρώσει ολόκληρη την διαδικασία εαν το επιθυμεί.

3.3.6.7 Προβολή Προφίλ Χρήστη

Σκοπός:

Η προβολή όλων των στοιχείων του χρήστη.

Είσοδος:

Ο χρήστης εαν έχει μόλις εκκινήσει το σύστημα εισάγει το Username και το Password, αλλιώς το προφίλ ανοιγει αυτόματα.

Επεξεργασία:

Με την εισαγωγή του Username και του Password γίνεται ανάκτηση των δεδομένων του χρήστη από την βάση και προβολή τους στην φόρμα του προφίλ. Ο χρήστης μπορεί να προσθέσει εικόνα στο προφίλ του, να διορθώσει τα στοιχεια του ή ακόμα και να διαγράψει το προφίλ του.

3.3.6.8 Διόρθωση Προφίλ Χρήστη

Σκοπός:

Η ανανέωση των στοιχείων του χρήστη.

Είσοδος:

Ο χρήστης πρέπει να είναι logged In στο σύστημα και να έχει μπεί να δεί το τρέχον του προφίλ.

Επεξεργασία:

Ο χρήστης αλλάζει οποιαδήποτε πληροφορία επιθυμεί και την αποθηκεύει. Γίνεται ξανά ο υπολογισμός του BMI και BMR του χρήστη.

Έξοδος:

Το ανανεωμένο προφίλ του χρήστη γίνεται Update στην βάση δεδομένων. Ο χρήστης μπορεί να ακυρώσει ολόκληρη την διαδικασία εαν το επιθυμεί.

3.3.6.9 Διαγραφή Χρήστη

Σκοπός:

Ο καθαρισμός της βάσης από αχρείαστα στοιχεία.

Είσοδος:

Ο χρήστης πρέπει να είναι Logged In για να διαγράψει το προφίλ του.

Επεξεργασία:

Εισαγωγή και επιβεβαίωση της ταυτότητας του χρήστη ούτως ώστε να μην διαγράφει το προφίλ από κάποιον που δεν έχει εξουσιοδότηση. Το σύστημα ρωτά εαν ο χρήστης είναι σίγουρος για αυτή του την πράξη.

Έξοδος:

Διαγραφή του χρήστη από την βάση. Ο χρήστης μπορεί να ακυρώσει την διαδικασία εαν το επιθυμεί.

3.3.6.10 Προβολή Διατροφών Χρήστη

Σκοπός:

Η προβολή όλων των διατροφών που δημιούργησε ο συγκεκριμένος χρήστης σαν εγγραφές.

Είσοδος:

Εισαγωγή και επιβεβαίωση της ταυτότητας του χρήστη ούτως ώστε να μην γίνει παραβίαση των προσωπικών στοιχείων άλλου χρήστη.

Επεξεργασία:

Με την εισαγωγή του Username και του Password γίνεται ανάκτηση των διατροφών του χρήστη από την βάση και προβολή τους στην φόρμα. Ο χρήστης μπορεί να προβάλει μια διατροφή, να την επεξεργαστει ή ακόμα και να διαγράψει ολόκληρη την διατροφή.

3.3.6.11 Διόρθωση Διατροφής

Σκοπός:

Η ανανέωση μιας διατροφής.

Είσοδος:

Ο χρήστης είναι logged In και αφού επιλέξει να δει τισ διατροφές που έχει δημιουργήσει, επιλέγει την διατροφή που θέλει να διορθώσει.

Επεξεργασία:

Η επιλεγόμενη διατροφή ανοίγει στην φόρμα διατροφής και ο χρήστης μπορεί να προσθέσει και να αφαιρέσει φαγητά. Για κάθε φαγητό που προσθέτει υπολογίζονται τα σύνολα των θρεπτικών συστατικών και των θερμίδων καθώς επίσης και τα υπόλοιπα τους από τον συνολικό αριθμό που δικαιούται σε μια μέρα.

Έξοδος:

Μια διατροφή που ικανοποιεί τις ανάγκες του χρήστη σε θρεπτικές ουσίες και ενέργεια.
Σε περίπτωση που δεν ικανοποιούνται οι ανάγκες του χρήστη εμφανίζεται σχετικό μήνυμα.
Ο χρήστης μπορεί να ακυρώσει ολόκληρη την διαδικασία εαν το επιθυμεί.

3.3.6.12 Διαγραφή Διατροφής

Σκοπός:

Η αφαίρεση μιας διατροφής από την βάση και το προφίλ του χρήστη.

Είσοδος:

Ο χρήστης πρέπει να είναι logged In και να επιλέξει να δει τις διατροφές που έχει δημιουργήσει για να επιλέξει την διατροφή που θέλει να διαγράψει.

Επεξεργασία:

Ο χρήστης επιλέγει την διατροφή που θέλει να διαγράψει και όταν επιλέξει "Διαγραφή" εμφανίζεται προειδοποιητικό μήνυμα σε περίπτωση που αλλάξει γνώμη.

3.3.6.13 Αλλαγή Κωδικού Πρόσβασης

Σκοπός:

Η ανανέωση των στοιχείων του χρήστη για ασφάλεια σε περίπτωση που νιώθει ότι απειλείται η ασφάλεια των στοιχείων του.

Είσοδος:

Ο χρήστης εισάγει το τρέχον του Username και Password.

Επεξεργασία:

Ο χρήστης εισάγει το τρέχον του Username και Password για επιβεβαίωση της ταυτότητας του. Γίνεται έλεγχος εαν πρόκειται για έγγυρα στοιχεία και αν ναι τότε επιλέγει νέο Password. Γίνεται έλεγχος εαν τα δυο Password είναι τα ίδια και αποθηκέυονται στην βάση δεδομένων

Έξοδος:

Με την αλλαγή του Password εμφανίζεται ανάλογο μήνυμα και ο χρήστης παραπέμπεται στο προφίλ του.

3.3.6.14 Log Out

Σκοπός:

Ο έξοδος του χρήστη από το σύστημα.

Είσοδος:

Πάτημα της Επιλογής Log Out.

Έξοδος:

Εμφανίζεται η αρχική φόρμα για Log In του χρήστη.

3.3.7 Άλλες Απαιτήσεις Λογισμικού

Αξιοπιστία

Το σύστημα πρέπει να είναι αξιόπιστο και να βγάζει πάντα αποτέλεσμα στον χρήστη.

Ασφάλεια

Ένα πολύ σημαντικό ζήτημα για το σύστημα είναι το θέμα της ασφάλειας. Αφού το σύστημα προορίζεται για πολλαπλούς χρήστες, πρέπει να προστατεύονται τα στοιχεία των χρηστών. Για αυτό το λόγο και υπάρχει η λειτουργία Log In με προστασία από Username και Password. Καλό θα ήταν επίσης, όταν δεν το λογισμικό παραμένει ανενεργό περισσότερο από συγκεκριμένη ώρα, να πρέπει ο χρήστης να ξανα κάνει Log In για να μην εκτείθονται τα δεδομένα σε κινδύνους.

Συντήρηση

Εαν υπάρχει πλήρης τεκμηρίωση του συστήματος μπορεί να διευκολυνθεί η διαδικασία της συντήρησης του. Η αντικειμενοστραφής σχεδίαση συμβάλει σε μεγάλο βαθμό στην συντήρηση αφού έτσι οι διάφορες λειτουργίες είναι ανεξάρτητες.

Ευκολία στη Μετακίνηση

Το σύστημα θα μπορεί να είναι εγκατεστημένο σε οποιοδήποτε υπολογιστή με Windows.

Κεφάλαιο 4

Προδιαγραφές Συστήματος

4.1 Εισαγωγή

4.2 Τεχνική Gane & Sarson

4.3 Αντικειμενοστραφής Ανάλυση

4.1 Εισαγωγή

Στην φάση των προδιαγραφών γίνεται η σαφής διατύπωση των λειτουργιών του συστήματος. Αυτό γίνεται μέσα από διαγράμματα τα οποία δείχνουν τις ακριβείς ανάγκες του χρήστη τις οποίες προσπαθούμε να καλύψουμε. Η ανάλυση των απαιτήσεων του συστήματος διαυκολύνει το στάδιο των προδιαγραφών, το οποίο με την σειρά του θα διευκολύνει την φάση του σχεδιασμού.

Θα χρησιμοποιηθεί η μεθοδολογία των Gane & Sarson για αντικειμενοστραφή ανάλυση συστημάτων η οποία αποτελείται από τα εξής βήματα:

- Δημιουργία Διαγραμμάτων Ροής Δεδομένων
- Εκλέπτυνση των ροών δεδομένων για την αναλυτική περιγραφή των στοιχείων που περνούν από κάθε ροή δεδομένων.
- Προσδιορισμός λογικής κάθε διαδικασίας
- Προσδιορισμός Αποθήκευσης Δεδομένων, με ορισμό των ακριβείς περιεχομένων κάθε αποθήκευσης (πεδια, μορφή πεδίων).
- Καθορισμός των φυσικών πόρων που θα χρησιμοποιηθούν και ανάλυση τους.
- Ορισμός Προδιαγραφών Εισόδων / Εξόδων.
- Εκτιμήσεις για μέγεθος δεδομένων εισόδου εξόδου
- Καθορισμός Απαιτήσεων σε λογισμικό

4.2 Τεχνική Gane & Sarson

4.2.1 Δημιουργία Διαγραμμάτων Ροής Δεδομένων (DFD)

Τα διαγράμματα ροής δεδομένων είναι μια γραφική αναπαράσταση της ροής των δεδομένων μέσα σε ένα σύστημα και της επεξεργασίας που γίνεται σε αυτά.

Ένα Διάγραμμα Ροής Δεδομένων αποτελείται από τα εξής σύμβολα:

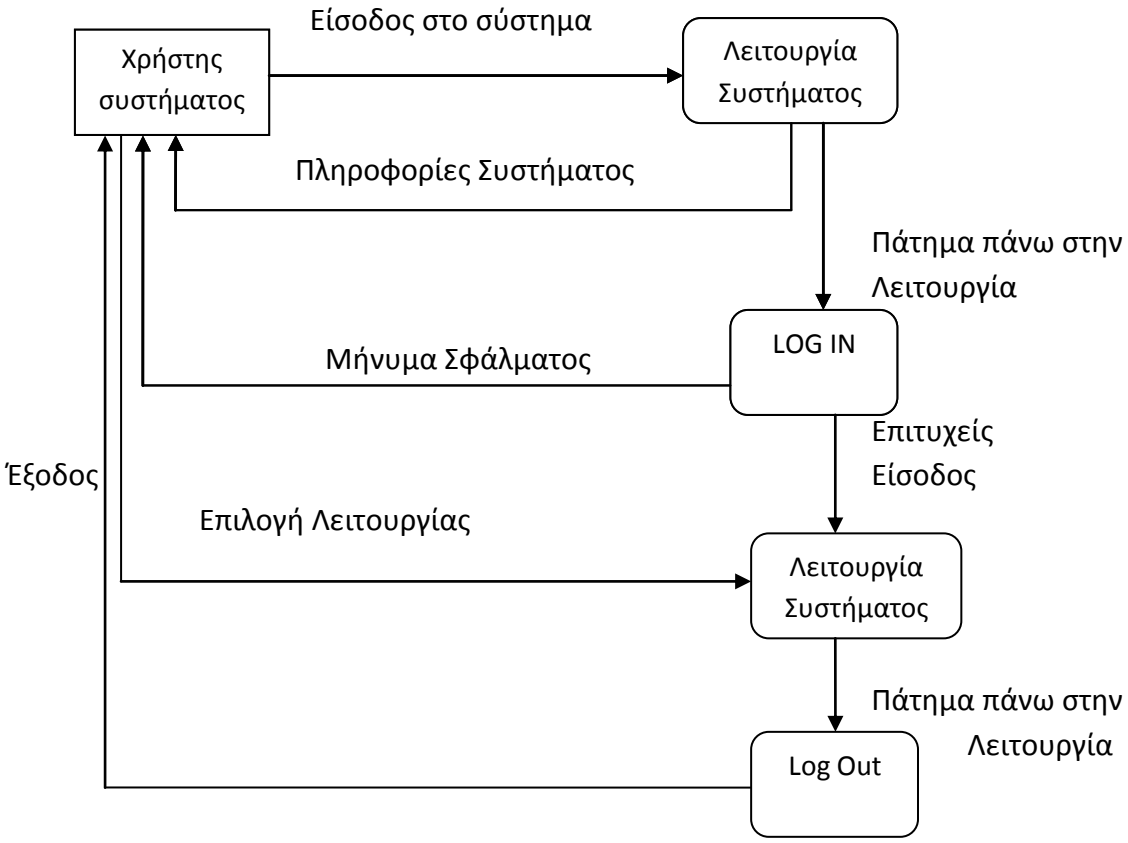
Το ορθογώνιο που αντιπροσωπεύει τον εξωτερικό πράκτορα ο οποίος μπορεί να είναι κάποιος χρήστης, ένα άλλο σύστημα ή οργανισμοί που αλληλεπιδρούν με το σύστημα.

Το ορθογώνιο με τις στρογγυλεμένες γωνιές ή το οβάλ αντιπροσωπεύει τις διαδικασίες, οι οποίες δέχονται δεδομένα σαν είσοδο και βγάζουν δεδομένα στην έξοδο. Πρέπει να υπάρχει πάντα ένα βέλος εισόδου και ένα βέλος εξόδου.

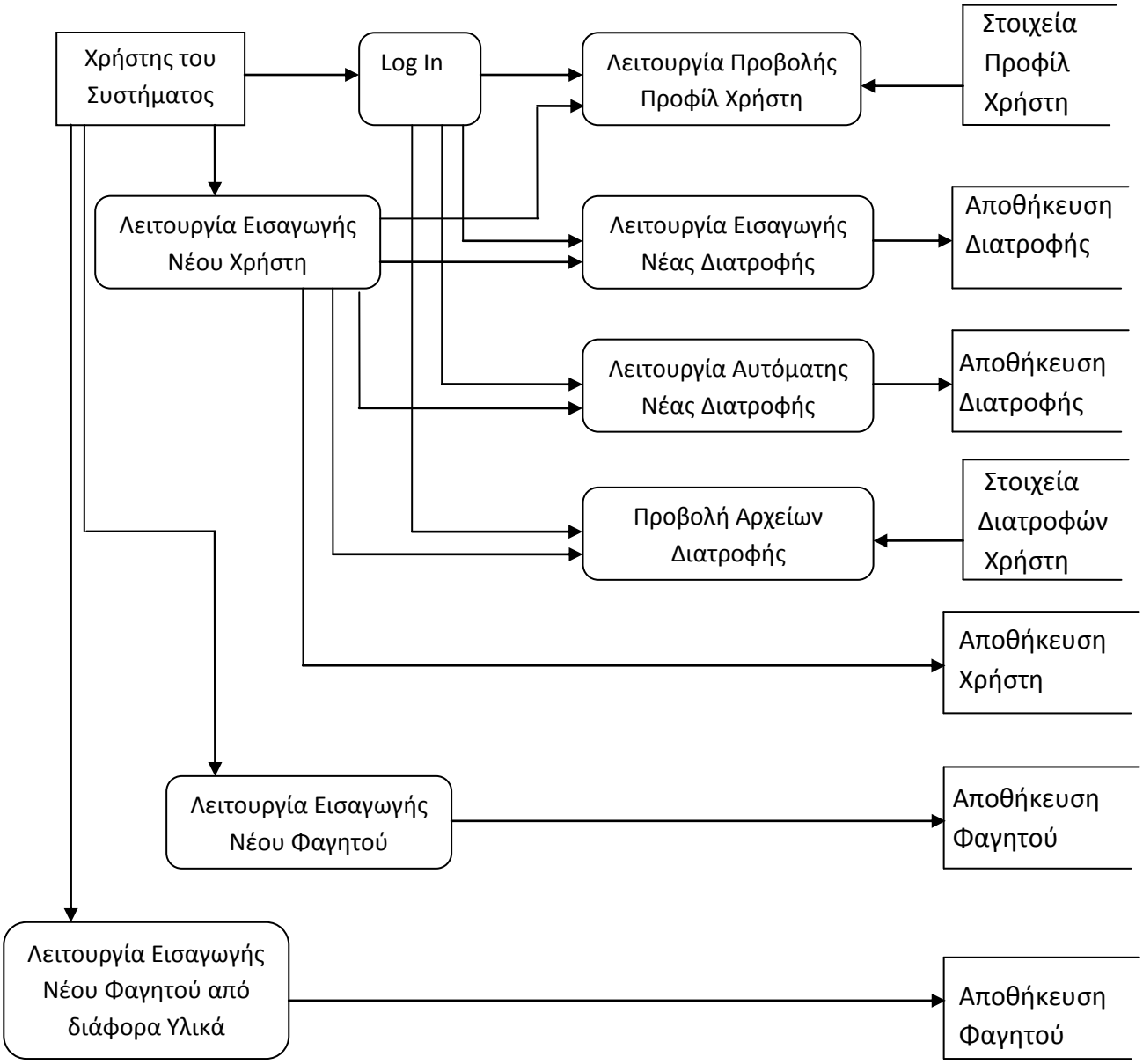
Το βέλος αναπαριστά την ροή των δεδομένων από και προς τις διαδικασίες, εξωτερικούς πράκτορες και αποθηκευτικές μονάδες.

Το ανοικτό ορθογώνιο αντιπροσωπεύει τις αποθηκευτικές μονάδες από τις οποίες ανακτώνται και αποθηκεύονται τα δεδομένα. Συνήθως υλοποιούνται ως αρχεία ή βάσεις δεδομένων.

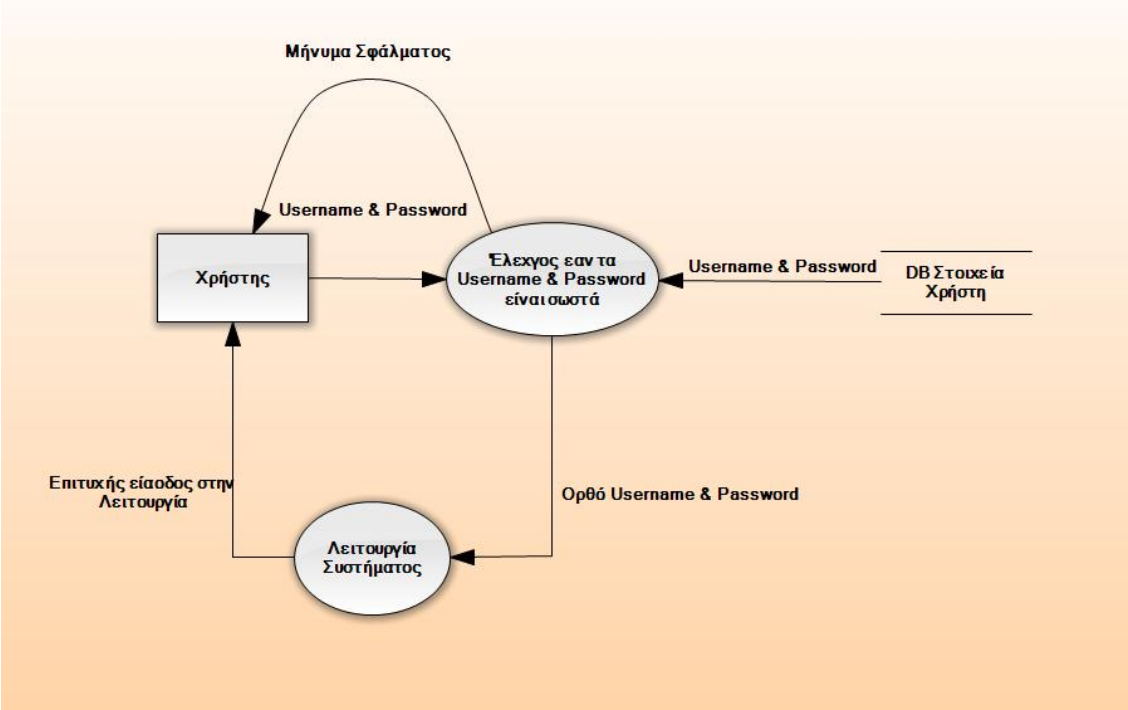
4.2.1.1 DFD για το σύστημα - Επίπεδο 0



4.2.1.2 DFD για τον χρήστη του συστήματος - Επίπεδο 1



4.2.1.3 DFD για την λειτουργία LOG IN του χρήστη στο σύστημα - Επίπεδο 2



4.2.1.3.1 Εκλέπτυνση Ροών Δεδομένων:

Username & Password: Τα στοιχεία που χρειάζονται για την είσοδο του χρήστη στο σύστημα.

Μήνυμα Σφάλματος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εισάγει λανθασμένα στοιχεία.

Ορθό Username & Password: Ο χρήστης έχει καταχωρήσει ορθό Username & Password.

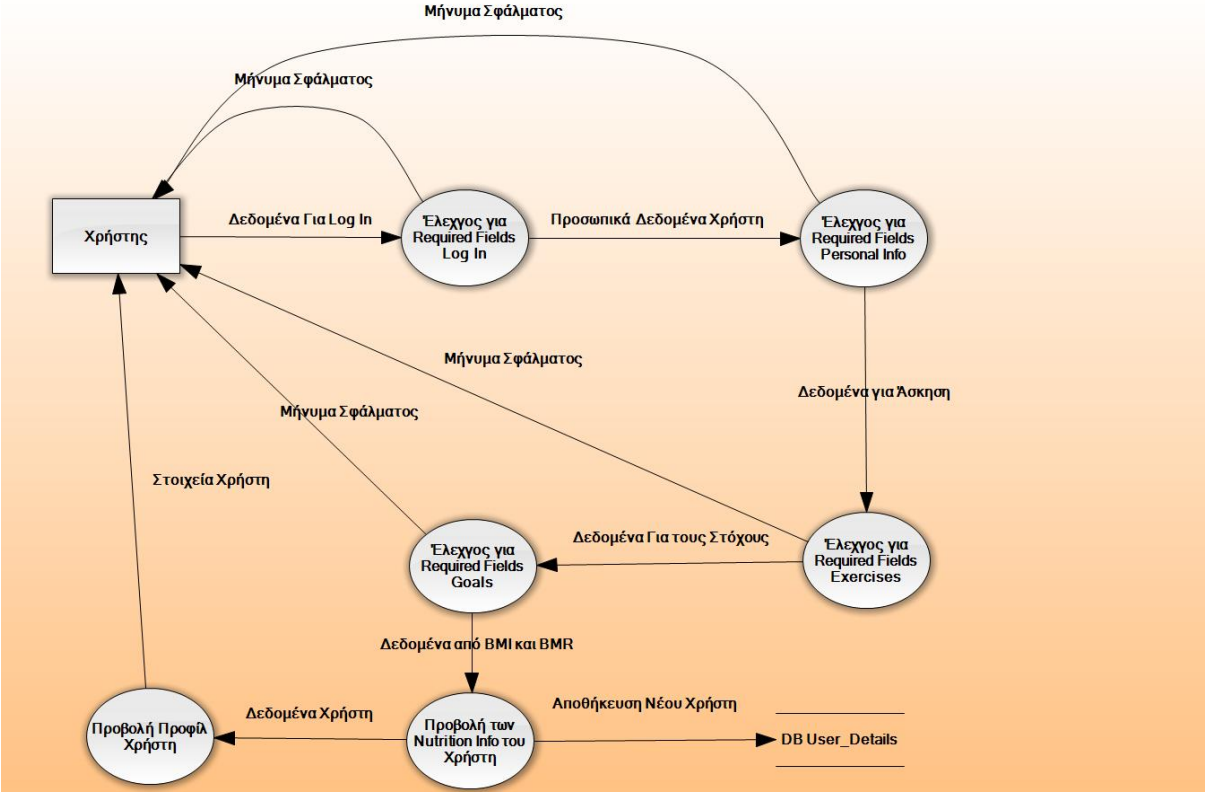
Επιτυχείς Είσοδος στην Λειτουργία: Το συνθηματικό που εισήγαγε ο χρήστης είναι ορθό και μπορεί να έχει πρόσβαση στις Λειτουργίες του συστήματος.

4.2.1.3.2 Προσδιορισμός λογικής διαδικασιών:

Έλεγχος εαν τα Username & Password είναι σωστά: Έλεγχος εάν τα Username & Password βρίσκονται στην ίδια εγγραφή στην βάση δεδομένων.

Λειτουργία Συστήματος: Οποιαδήποτε Λειτουργία μπορεί να εκτελέσει το σύστημα και χρειάζεται να γίνει πρώτα αναγνώριση του χρήστη.

4.2.1.4 DFD για την λειτουργία Δημιουργία Νέου Χρήστη - Επίπεδο 2



4.2.1.4.1 Εκλέπτυνση Ροών Δεδομένων:

Δεδομένα για Log In: Τα στοιχεία που χρειάζονται για την είσοδο του χρήστη στο σύστημα.

Μήνυμα Σφάλματος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εισάγει λανθασμένα στοιχεία.

Προσωπικά Δεδομένα Χρήστη: Τα προσωπικά στοιχεία του χρήστη που πρέπει να εισάγει για να δημιουργηθεί το προφίλ του όπως όνομα, ηλικία, ύψος, βάρος.

Δεδομένα για Άσκηση: Εισαγωγή δεδομένων για το επίπεδο της άσκησης που κάνει καθημερινά.

Δεδομένα Για τους Στόχους: Εισαγωγή των στόχων του χρήστη στο σύστημα.

Δεδομένα από BMI και BMR: Υπολογιζόμενα στοιχεία τα οποία βγαίνουν από τα υπόλοιπα στοιχεία του χρήστη.

Δεδομένα Χρήστη: Όλα τα στοιχεία του χρήστη (Υπολογιζόμενα και εισαγόμενα από τον ίδιο τον χρήστη).

Αποθήκευση Νέου Χρήστη: Αποθήκευση όλων των στοιχείων του χρήστη (Υπολογιζόμενων και εισαγόμενων από τον ίδιο τον χρήστη) στην βάση δεδομένων.

Στοιχεία Χρήστη: Προβολή όλων των στοιχείων του χρήστη (Υπολογιζόμενων και εισαγόμενων από τον ίδιο τον χρήστη).

4.2.1.4.2 Προσδιορισμός λογικής διαδικασιών:

Έλεγχος για Required Fields Log In: Έλεγχος εάν τα απαραίτητα στοιχεία του Log In (Username, Password) έχουν εισαχθεί από τον χρήστη.

Έλεγχος για Required Fields Personal Info: Έλεγχος εάν τα απαραίτητα στοιχεία του Personal Info (όνομα, επώνυμο, ηλικία, ύψος, βάρος) έχουν εισαχθεί από τον χρήστη.

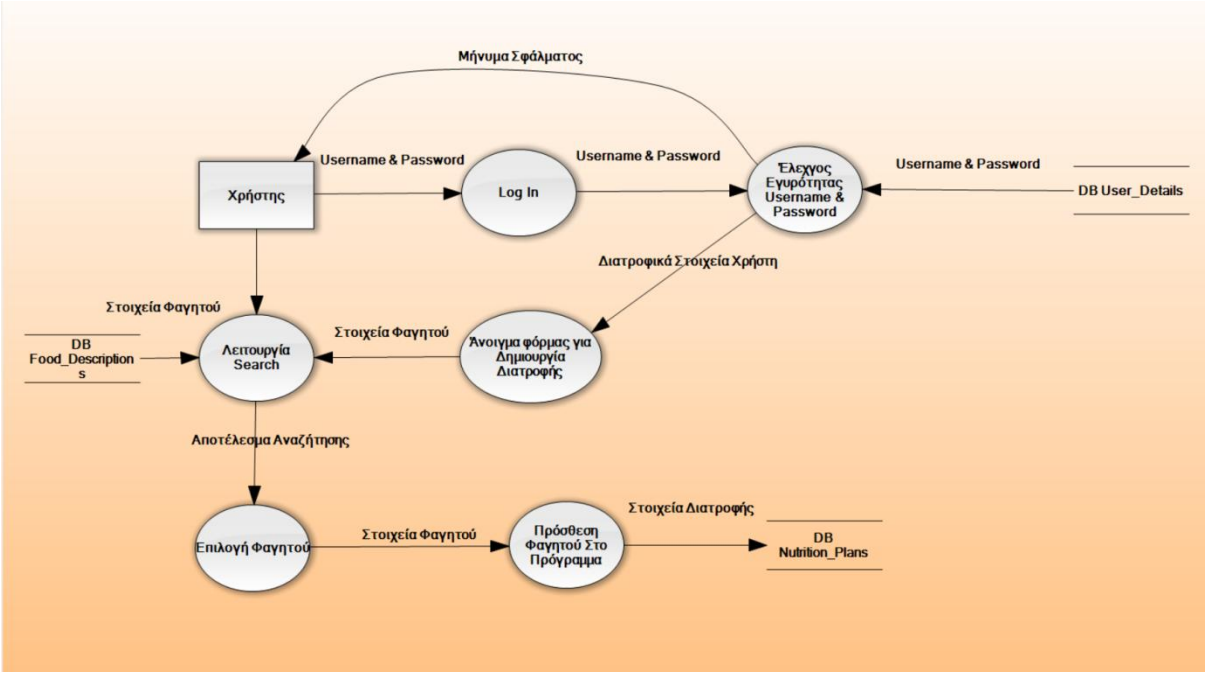
Έλεγχος για Required Fields Exercises: Έλεγχος εάν τα απαραίτητα στοιχεία του Exercises (το πόσο ενεργός είναι οι χρήστης) έχουν εισαχθεί από τον χρήστη.

Έλεγχος για Required Fields Goals: Έλεγχος εάν τα απαραίτητα στοιχεία του Goals έχουν εισαχθεί από τον χρήστη. Αυτό περιλαμβάνει το αν θέλει να χασει, να βάλει ή να διατηρήσει το βάρος του και τα κιλά που επιθυμεί να φτάσει.

Προβολή των Nutrition Info του Χρήστη: Προβολή των πληροφοριών που έχουν υπολογιστεί από τα δεδομένα που έδωσε ο χρήστης στο σύστημα.

Προβολή Προφίλ Χρήστη: Η προβολή του πλήρες προφίλ του χρήστη με όλα του τα στοιχεία.

4.2.1.5 DFD για την λειτουργία Δημιουργία Διατροφής - Επίπεδο 2



4.2.1.5.1 Εκλέπτυνση Ροών Δεδομένων:

Username & Password: Τα στοιχεία που χρειάζονται για την είσοδο του χρήστη στο σύστημα.

Μήνυμα Σφάλματος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εισάγει λανθασμένα στοιχεία.

Διατροφικά Στοιχεία Χρήστη: Τα διατροφικά στοιχεία του χρήστη τα οποία βλέπει όταν ανοίγει την φόρμα δημιουργίας διατροφής. Πρόκειται για τα υπολογιζόμενα δεδομένα τα οποία υπολογίζονται κατά την δημιουργία του προφίλ του.

Στοιχεία Φαγητού: Δεδομένα που εισάγει για την αναζήτηση φαγητού στην βάση. Συνήθως το όνομα του φαγητού.

Αποτέλεσμα Αναζήτησης: Όλα τα φαγητά που βρέθηκαν με το όνομα που εισήγαγε ο χρήστης.

Στοιχεία Διατροφής: Το σύνολο των φαγητών που επιλέχθηκαν να είναι στο πρόγραμμα διατροφής και όλα τα στοιχεία τους τα οποία αποθηκεύνται στην βάση δεδομένων.

4.2.1.5.2 Προσδιορισμός λογικής διαδικασιών:

Log In: Η εξοσιοδοτημένη είσοδος του χρήστη στο σύστημα.

Έλεγχος εγκυρότητας Username & Password: Έλεγχος εάν τα Username & Password βρίσκονται στην ίδια εγγραφή στην βάση δεδομένων.

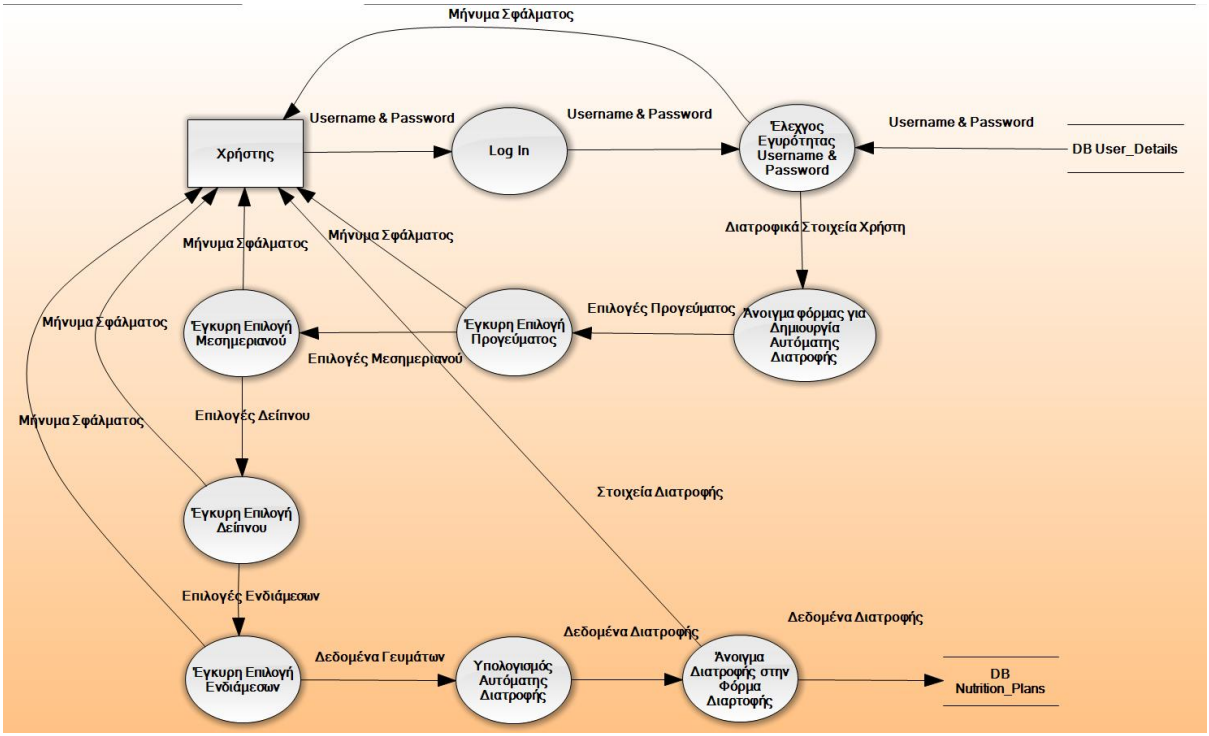
Άνοιγμα Φόρμας Για Δημιουργία Διατροφής: Το άνοιγμα της φόρμας για Δημιουργία Διατροφής, στην οποία θα αναγράφονται τα επιτρεπόμενα όρια των θρεπτικών συστατικών τα οποία δικαιούται να παίρνει ο χρήστης σε μια μέρα.

Λειτουργία Search: Η αναζήτηση του φαγητού που θέλει ο χρήστης να συμπεριλάβει στην διατροφή του.

Επιλογή Φαγητού: Η επιλογή του φαγητού από την λίστα με τα φαγητά που βρέθηκαν από την αναζήτηση.

Πρόσθεση του φαγητού στο Πρόγραμμα: Πρόσθεση συγκρεκριμένης ποσότητας φαγητού στην διατροφή.

4.2.1.6 DFD για την λειτουργία Δημιουργία Αυτόματης Διατροφής - Επίπεδο 2



4.2.1.6.1 Εκλέπτυνση Ροών Δεδομένων:

Username & Password: Τα στοιχεία που χρειάζονται για την είσοδο του χρήστη στο σύστημα.

Μήνυμα Σφάλματος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εισάγει λανθασμένα στοιχεία.

Διατροφικά Στοιχεία Χρήστη: Τα διατροφικά στοιχεία του χρήστη τα οποία βλέπει όταν ανοίγει την φόρμα δημιουργίας διατροφής. Πρόκειται για τα υπολογιζόμενα δεδομένα τα οποία υπολογίζονται κατά την δημιουργία του προφίλ του.

Επιλογές Προγεύματος: Οι επιλογές που θα κάνει ο χρήστης για το τι επιθυμεί να είναι το πρόγευμα του.

Επιλογές Μεσημεριανού: Οι επιλογές που θα κάνει ο χρήστης για το τι επιθυμεί να είναι το μεσημεριανό του.

Επιλογές Δείπνου: Οι επιλογές που θα κάνει ο χρήστης για το τι επιθυμεί να είναι το δείπνο του.

Επιλογές Ενδιάμεσων: Οι επιλογές που θα κάνει ο χρήστης για το τι επιθυμεί να είναι τα ενδιάμεσα του.

Υπολογισμός Αυτόματης Διατροφής: Το τρέξιμο του προγράματος ικανοποίησης περιορισμών για εύρεση της βέλτιστης διατοφής για ικανοποίηση των ημερήσιων αναγκών του χρήστη. Πρόκειται για εύρεση των κατάλληλων τροφών σε κάθε γεύμα και της κατάλληλης ποσότητας της κάθε τροφής ουτως ώστε τα θρεπτικά συστατικά να είναι μέσα στα επιτρεπόμενα όρια.

Άνοιγμα της Διατροφής στην Φόρμα Διατροφής: Η εμφάνιση της διατροφής που υπολογίστηκε στον χρήστη μέσα στην φόρμα για τις διατροφές.Εδώ ο χρήστης μπορεί να επιλέξει εαν θέλει να αποθηκεύσει την διατροφή.

4.2.1.6.2 Προσδιορισμός λογικής διαδικασιών:

Log In: Η εξοσιοδοτημένη είσοδος του χρήστη στο σύστημα.

Έλεγχος εγκυρότητας Username & Password: Έλεγχος εάν τα Username & Password βρίσκονται στην ίδια εγγραφή στην βάση δεδομένων.

Άνοιγμα Φόρμας Για Δημιουργία Αυτόματης Διατροφής: Το άνοιγμα της φόρμας για Δημιουργία Διατροφής, από την οποία ο χρήστης πρέπει να επιλέξει το είδος του φαγητού που επιτυμεί να έχει σε κάθε γεύμα.

Έγκυρη Επιλογή Προγεύματος: Ο χρήστης πρέπει να επιλέξει τροφές από τς 4 ομάδες τροφίμων. Δημητριακά/Υδατάνθρακες, Γαλακτοκομικά, Πρωτείνες και Φρούτα.

Έγκυρη Επιλογή Μεσημεριανού: Ο χρήστης πρέπει να επιλέξει τροφές από διάφορα είδη. Μπορεί να επιλέξει Εθνικό φαγητό, Φαγητό εστιατορίου ή να φτιάξει το δικό του υγιεινό γεύμα επιλέγοντας τροφές από τις 5 βασικές ομάδες:Πρωτείνες, Υδατάνθρακες, Γαλακτοκομικά και Λαχανικά και όσπρια.

Έγκυρη Επιλογή Δείπνου: Ο χρήστης πρέπει να επιλέξει τροφές από διάφορα είδη. Μπορεί να επιλέξει Εθνικό φαγητό, Φαγητό εστιατορίου ή να φτιάξει το δικό του υγιεινό γεύμα επιλέγοντας τροφές από τις 5 βασικές ομάδες:Πρωτείνες, Υδατάνθρακες, Γαλακτοκομικά και Λαχανικά και όσπρια.

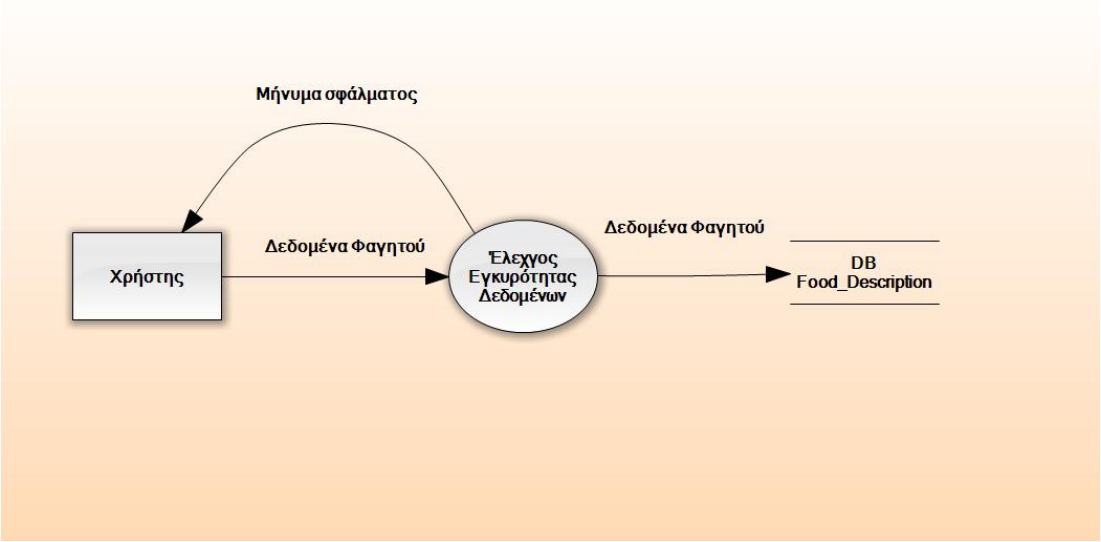
Έγκυρη Επιλογή Ενδιάμεσων: Ο χρήστης πρέπει να επιλέξει από διάφορα είδη ενδιάμεσων όπως φρουτα, snack bars, ξηρούς καρπούς, ροφήματα, ή γλυκά και κέικ.

Λειτουργία Search: Η αναζήτηση του φαγητού που θέλει ο χρήστης να συμπεριλάβει στην διατροφή του.

Επιλογή Φαγητού: Η επιλογή του φαγητού από την λίστα με τα φαγητά που βρέθηκαν από την αναζήτηση.

Πρόσθεση του φαγητού στο Πρόγραμμα: Πρόσθεση συγκρεκριμένης ποσότητας φαγητού στην διατροφή.

4.2.1.7 DFD για την λειτουργία Εισαγωγή Νέου Φαγητού - Επίπεδο 2



4.2.1.7.1 Εκλέπτυνση Ροών Δεδομένων:

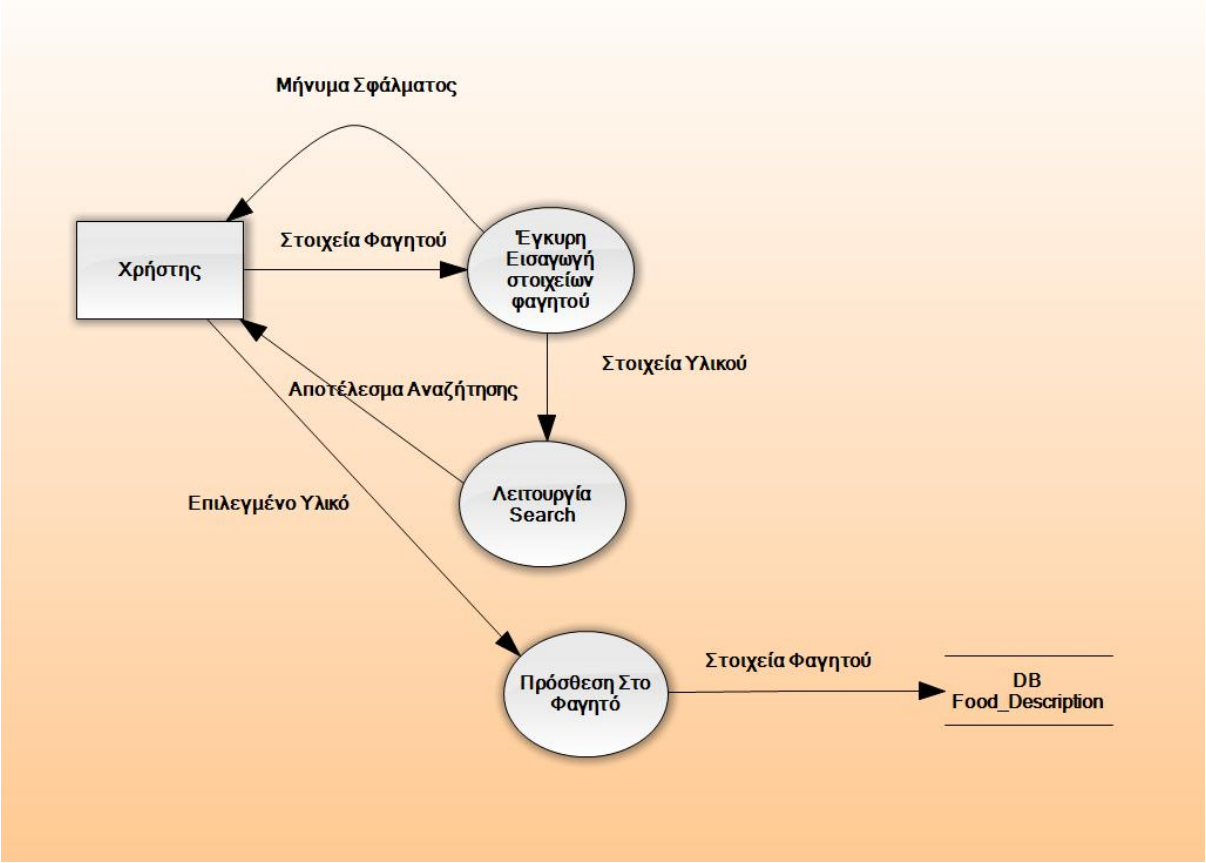
Δεδομένα Φαγητού: Τα στοιχεία που χρειάζονται για την είσοδο του φαγητού στο σύστημα όπως αυτά αναγράφονται στην συσκευασία του.

Μήνυμα Σφάλματος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εισάγει λανθασμένα στοιχεία.

4.2.1.7.2 Προσδιορισμός λογικής διαδικασιών:

Έλεγχος εγκυρότητας Δεδομένων: Έλεγχος εάν τα στοιχεία του φαγητού συμπληρώθηκαν όλα.

4.2.1.8 DFD για την λειτουργία Εισαγωγή Νέου Φαγητού από Διάφορα Υλικά - Επίπεδο 2



4.2.1.8.1 Εκλέπτυνση Ροών Δεδομένων:

Στοιχεία Φαγητού: Τα στοιχεία που χρειάζονται για την είσοδο του φαγητού στο σύστημα όπως τίτλος και αριθμός μερίδων και τα υλικά από τα οποία αποτείται.

Μήνυμα Σφάλματος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εισάγει λανθασμένα στοιχεία.

Στοιχεία Υλικού: Τα στοιχεία που πρέπει να εισάγει ο χρήστης για να αναζητήσει το υλικό που επιθυμεί από την βάση.

Αποτέλεσμα αναζήτησης: Όλα τα υλικά που βρέθηκαν στην βάση που ταιριάζουν με την περιγραφή που δώθηκε.

Επιλεγμένο Υλικό: Το υλικό που επιλέγει από την λίστα με τα υλικά που βρέθηκαν.

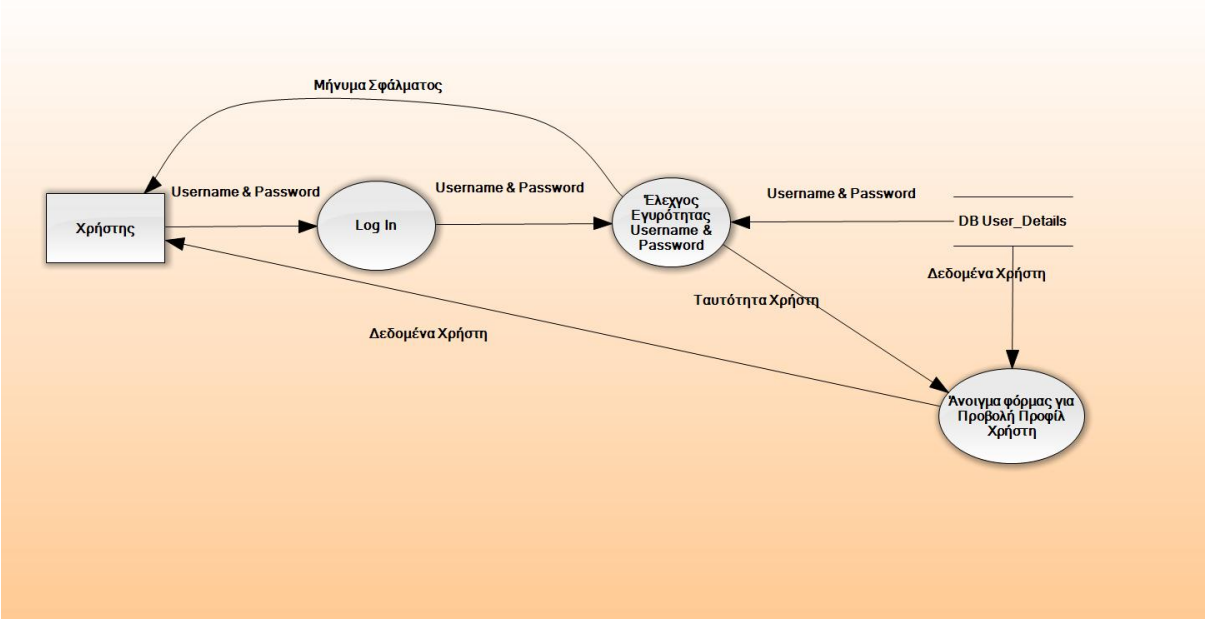
4.2.1.8.2 Προσδιορισμός λογικής διαδικασιών:

Έγκυρη Εισαγωγή Στοιχείων Φαγητού: Έλεγχος εάν τα στοιχεία του φαγητού (Όνομα καο αριθμός μερίδων) συμπληρώθηκαν όλα.

Λειτουργία Search: Η αναζήτηση του υλικού που θέλει ο χρήστης να συμπεριλάβει στο φαγητό του.

Πρόσθεση στο Φαγητό: Πρόσθεση του υλικού στο φαγητό και υπολογισμός των ανάλογων θερμίδων σε όλο το φαγητό, ανα μερίδα και ανά 100 grams.

4.2.1.9 DFD για την λειτουργία Προβολή Προφίλ Χρήστη- Επίπεδο 2



4.2.1.9.1 Εκλέπτυνση Ροών Δεδομένων:

Username & Password: Τα στοιχεία που χρειάζονται για την είσοδο του χρήστη στο σύστημα.

Μήνυμα Σφάλματος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εισάγει λανθασμένα στοιχεία.

Δεδομένα Χρήστη: Τα στοιχεία του χρήστη τα οποία υπάρχουν καταχωρημένα στην βάση .

Ταυτότητα Χρήστη: Η ταυτότητα που δώθηκε στον χρήστη κατά την δημιουργία του προφίλ του.

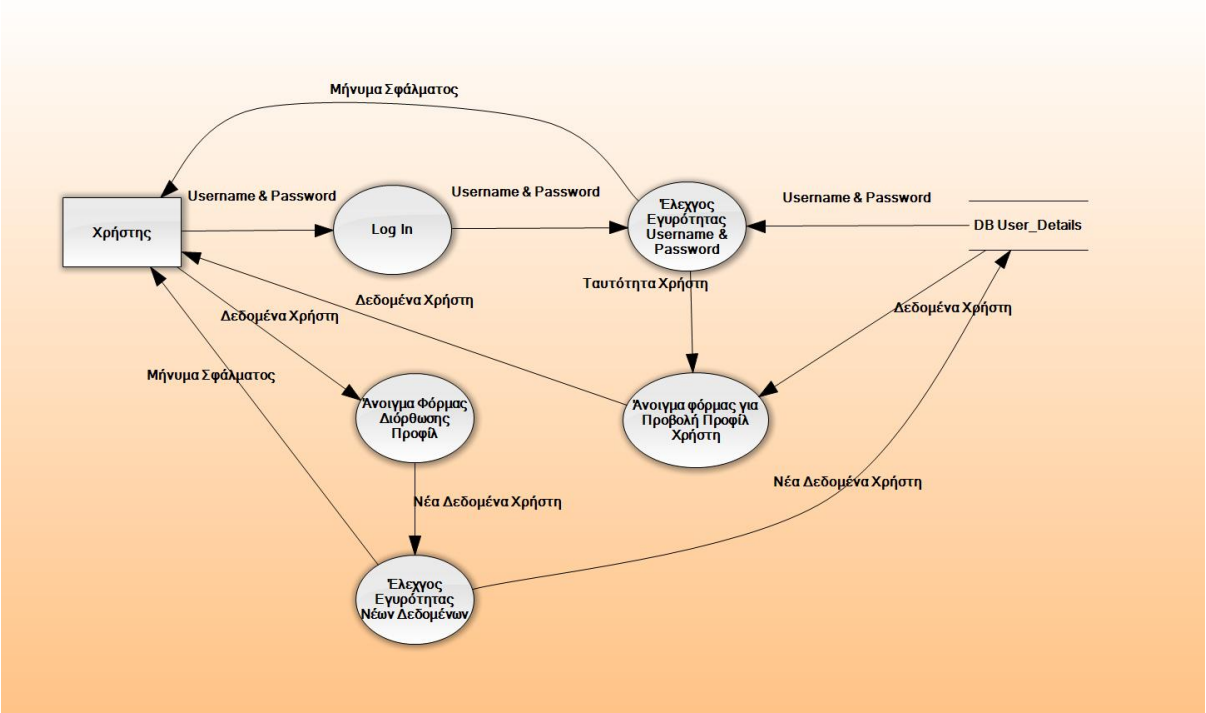
4.2.1.9.2 Προσδιορισμός λογικής διαδικασιών:

Log In: Η εξοσιοδοτημένη είσοδος του χρήστη στο σύστημα.

Έλεγχος εγκυρότητας Username & Password: Έλεγχος εάν τα Username & Password βρίσκονται στην ίδια εγγραφή στην βάση δεδομένων.

Άνοιγμα Φόρμας Για Προβολή Προφίλ Χρήστη: Το άνοιγμα της φόρμας για προβολή του Προφίλ του χρήστη στην οποία εμφανίζονται όλα τα στοιχεία του χρήστη.

4.2.1.10 DFD για την λειτουργία Διόρθωση Προφίλ Χρήστη- Επίπεδο 2



4.2.1.10.1 Εκλέπτυνση Ροών Δεδομένων:

Username & Password: Τα στοιχεία που χρειάζονται για την είσοδο του χρήστη στο σύστημα.

Μήνυμα Σφάλματος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εισάγει λανθασμένα στοιχεία.

Δεδομένα Χρήστη: Τα στοιχεία του χρήστη τα οποία υπάρχουν καταχωρημένα στην βάση .

Ταυτότητα Χρήστη: Η ταυτότητα που δώθηκε στον χρήστη κατά την δημιουργία του προφίλ του.

Νέα δεδομένα Χρήστη: Τα νέα δεδομένα που εισάγει ο χρήστης στην θέση των υφιστάμενων, με τα οποία θέλει να τα αντικαταστήσει.

4.2.1.10.2 Προσδιορισμός λογικής διαδικασιών:

Log In: Η εξοσιοδοτημένη είσοδος του χρήστη στο σύστημα.

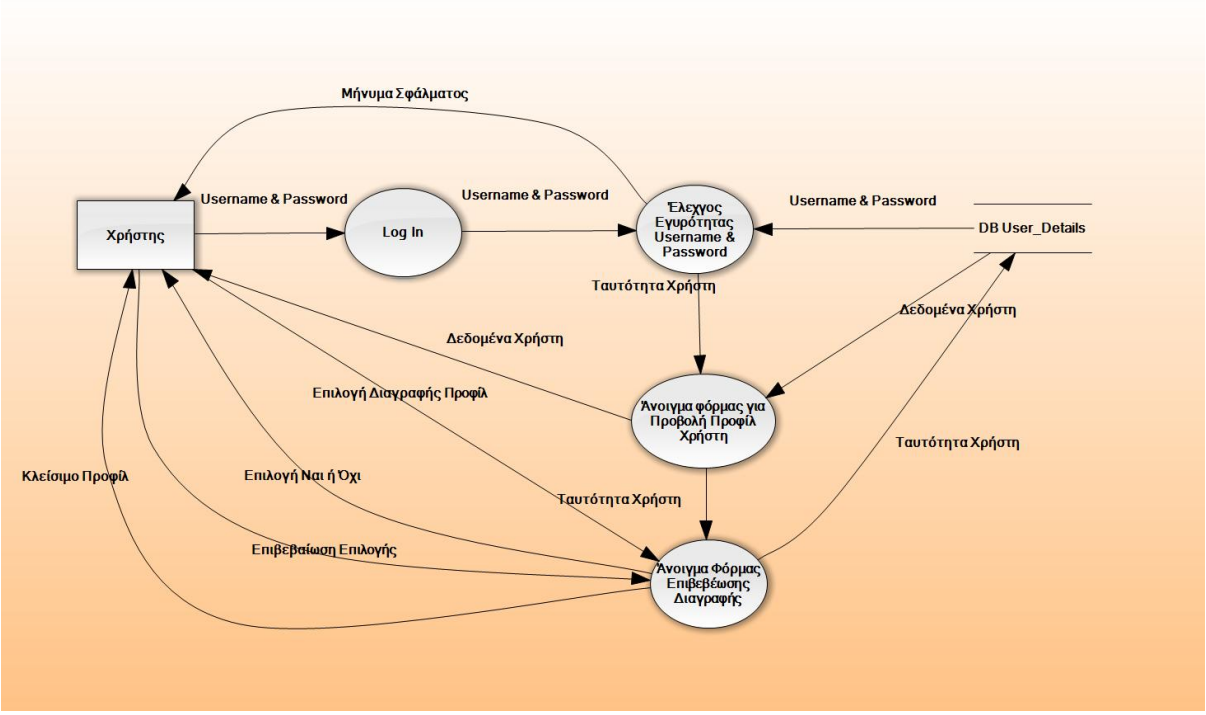
Έλεγχος εγκυρότητας Username & Password: Έλεγχος εάν τα Username & Password βρίσκονται στην ίδια εγγραφή στην βάση δεδομένων.

Άνοιγμα Φόρμας Για Προβολή Προφίλ Χρήστη: Το άνοιγμα της φόρμας για προβολή του Προφίλ του χρήστη στην οποία εμφανίζονται όλα τα στοιχεία του χρήστη.

Άνοιγμα Φόρμας Για Διόρθωση Προφίλ Χρήστη: Το άνοιγμα της φόρμας για διόρθωση του Προφίλ του χρήστη στην οποία εμφανίζονται όλα τα στοιχεία του χρήστη συμπληρωμένα και τα οποία ο χρήστης μπορεί να αλλάξει.

Έλεγχος εγκυρότητας Νέων Δεδομένων: Έλεγχος έαν όλα τα πεδία έχουν συμπληρωθεί για να πάρουν τιμές τα απαραίτητα πεδια(required fields).

4.2.1.11 DFD για την λειτουργία Διαγραφή Προφίλ Χρήστη- Επίπεδο 2



4.2.1.11.1 Εκλέπτυνση Ροών Δεδομένων:

Username & Password: Τα στοιχεία που χρειάζονται για την είσοδο του χρήστη στο σύστημα.

Μήνυμα Σφάλματος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εισάγει λανθασμένα στοιχεία.

Δεδομένα Χρήστη: Τα στοιχεία του χρήστη τα οποία υπάρχουν καταχωρημένα στην βάση .

Ταυτότητα Χρήστη: Η ταυτότητα που δώθηκε στον χρήστη κατά την δημιουργία του προφίλ του.

Επιλογή Διαγραφής Προφίλ: Ο χρήστης επιλέγει να διαγράψει το προφίλ του.

Επιλογή Ναι ή Όχι: Η εμφάνιση του μηνύματος ότι ο χρήστης είναι σίγουρος εαν θέλει να διαγράψει το προφίλ του ή όχι.

Επιβεβαίωση Επιλογής: Η επιλογή του χρήστη εαν θα διαγραφεί το προφίλ του.

Κλείσιμο Προφίλ: Εαν διαγράφηκε το προφίλ, τότε κλείνει η φόρμα του.

4.2.1.11.2 Προσδιορισμός λογικής διαδικασιών:

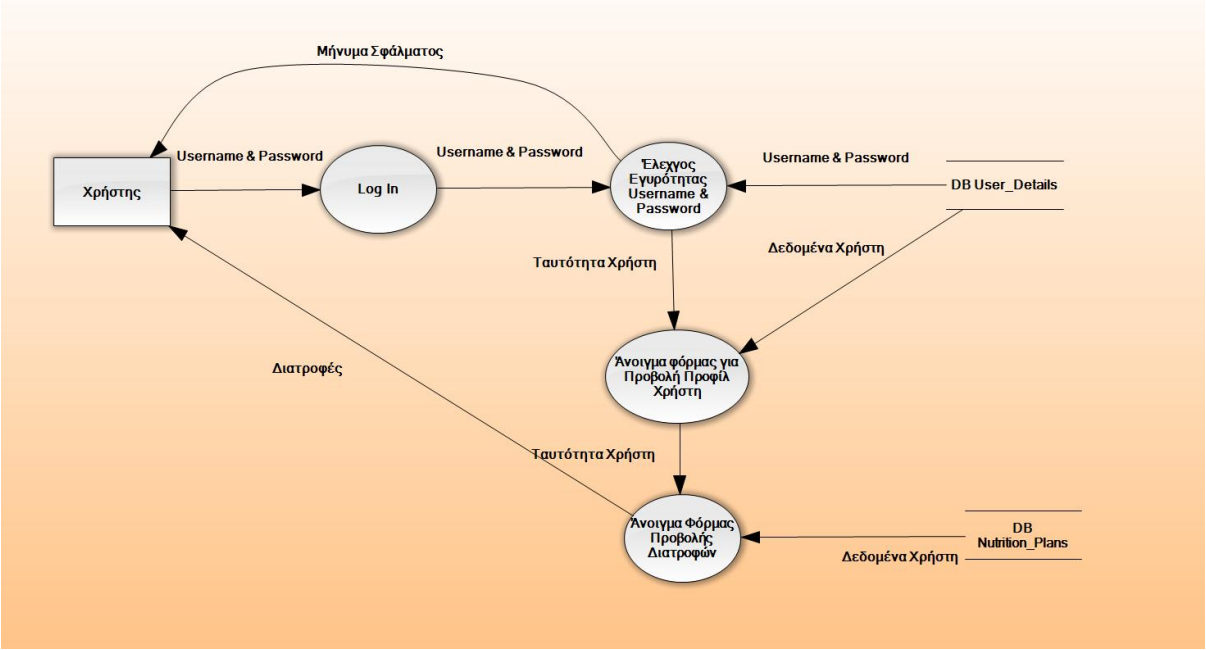
Log In: Η εξοσιοδοτημένη είσοδος του χρήστη στο σύστημα.

Έλεγχος εγκυρότητας Username & Password: Έλεγχος εάν τα Username & Password βρίσκονται στην ίδια εγγραφή στην βάση δεδομένων.

Άνοιγμα Φόρμας Για Προβολή Προφίλ Χρήστη: Το άνοιγμα της φόρμας για προβολή του Προφίλ του χρήστη στην οποία εμφανίζονται όλα τα στοιχεία του χρήστη.

Άνοιγμα Φόρμας Για Επιβεβαίωση Διαγραφής Χρήστη: Το άνοιγμα της φόρμας επιβεβαίωσης διαγραφής σε περίπτωση που ο χρήστης αλλάξει γνώμη και δεν θέλει να διαγράψει το προφίλ του.

4.2.1.12 DFD για την λειτουργία Προβολή Διατροφών Χρήστη- Επίπεδο 2



4.2.1.12.1 Εκλέπτυνση Ροών Δεδομένων:

Username & Password: Τα στοιχεία που χρειάζονται για την είσοδο του χρήστη στο σύστημα.

Μήνυμα Σφάλματος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εισάγει λανθασμένα στοιχεία.

Δεδομένα Χρήστη: Τα στοιχεία του χρήστη τα οποία υπάρχουν καταχωρημένα στην βάση σχετικά με όλες τις διατροφές που έχει δημιουργήσει.

Ταυτότητα Χρήστη: Η ταυτότητα που δώθηκε στον χρήστη κατά την δημιουργία του προφίλ του.

Διατροφές: Όλες οι διατροφές που έχει δημιουργήσει ο συγκεκριμένος χρήστης.

4.2.1.12.2 Προσδιορισμός λογικής διαδικασιών:

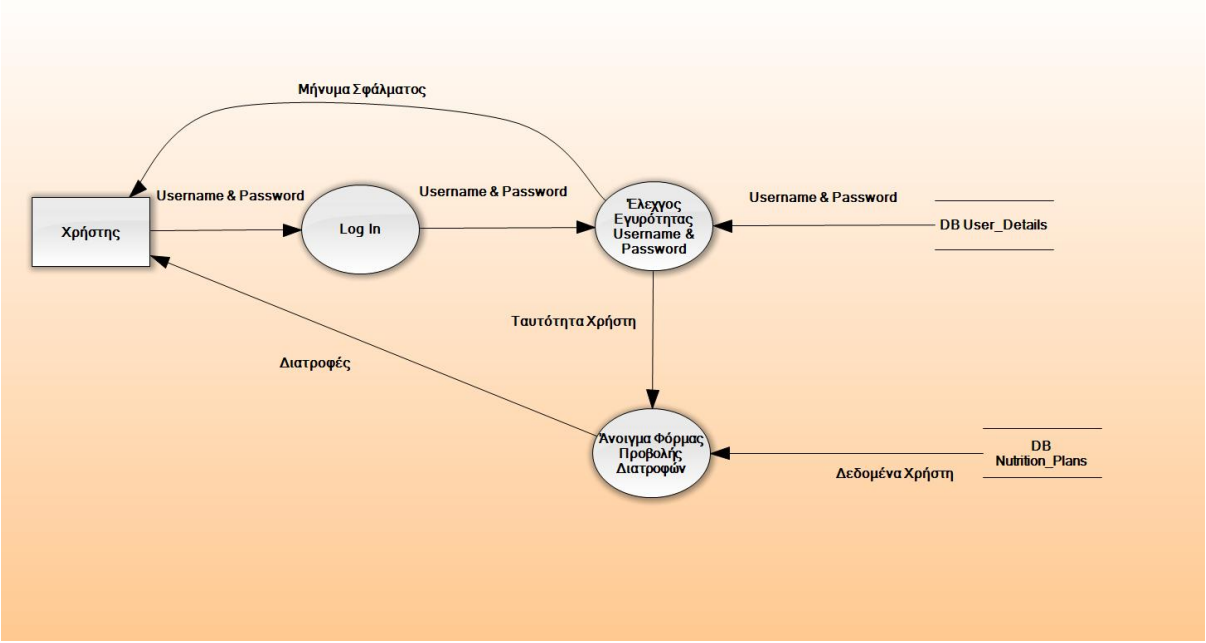
Log In: Η εξοσιοδοτημένη είσοδος του χρήστη στο σύστημα.

Έλεγχος εγκυρότητας Username & Password: Έλεγχος εάν τα Username & Password βρίσκονται στην ίδια εγγραφή στην βάση δεδομένων.

Άνοιγμα Φόρμας Για Προβολή Προφίλ Χρήστη: Το άνοιγμα της φόρμας για προβολή του Προφίλ του χρήστη στην οποία εμφανίζονται όλα τα στοιχεία του χρήστη.

Άνοιγμα Φόρμας Για Προβολή Διατροφών: Το άνοιγμα της φόρμας για προβολή των διατροφών του συγκεκριμένου χρήστη.

4.2.1.13 DFD για την λειτουργία Προβολή Διατροφών Χρήστη (2)- Επίπεδο 2



4.2.1.13.1 Εκλέπτυνση Ροών Δεδομένων:

Username & Password: Τα στοιχεία που χρειάζονται για την είσοδο του χρήστη στο σύστημα.

Μήνυμα Σφάλματος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εισάγει λανθασμένα στοιχεία.

Δεδομένα Χρήστη: Τα στοιχεία του χρήστη τα οποία υπάρχουν καταχωρημένα στην βάση σχετικά με όλες τις διατροφές που έχει δημιουργήσει.

Ταυτότητα Χρήστη: Η ταυτότητα που δώθηκε στον χρήστη κατά την δημιουργία του προφίλ του.

Διατροφές: Όλες οι διατροφές που έχει δημιουργήσει ο συγκεκριμένος χρήστης.

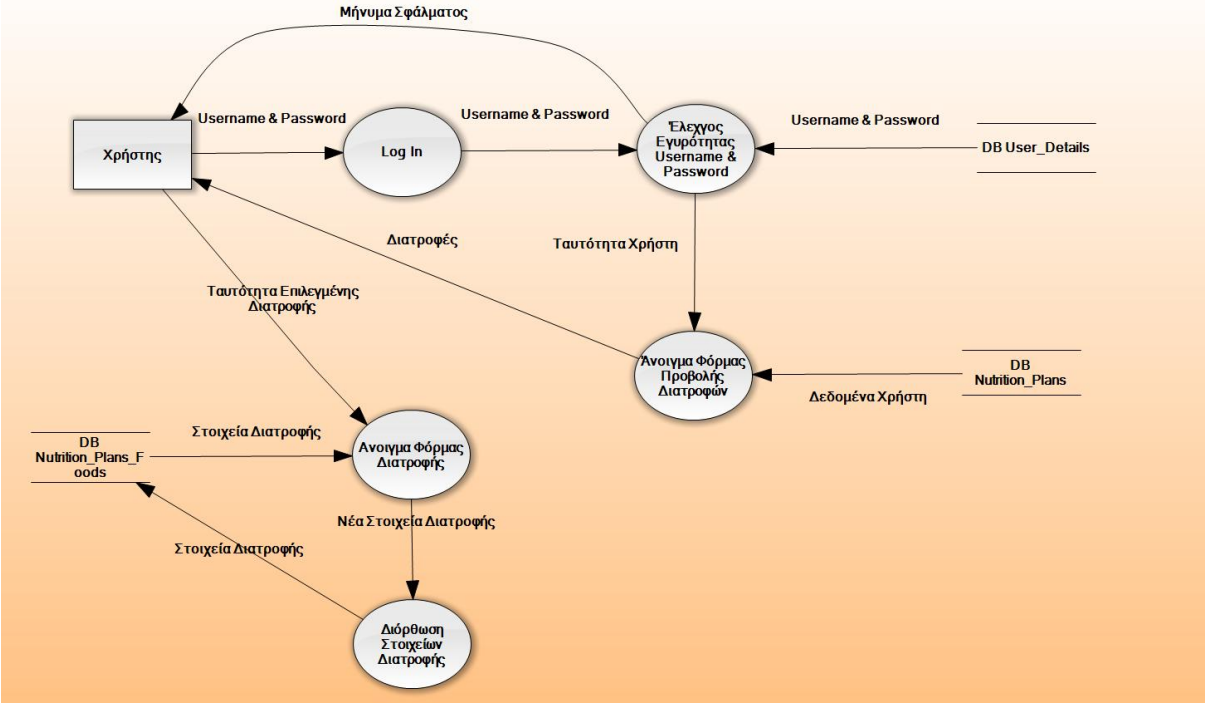
4.2.1.13.2 Προσδιορισμός λογικής διαδικασιών:

Log In: Η εξοσιοδοτημένη είσοδος του χρήστη στο σύστημα.

Έλεγχος εγκυρότητας Username & Password: Έλεγχος εάν τα Username & Password βρίσκονται στην ίδια εγγραφή στην βάση δεδομένων.

Άνοιγμα Φόρμας Για Προβολή Διατροφών: Το άνοιγμα της φόρμας για προβολή των διατροφών του συγκεκριμένου χρήστη.

4.2.1.14 DFD για την λειτουργία Διόρθωση Διατροφής Χρήστη- Επίπεδο 2



4.2.1.14.1 Εκλέπτυνση Ροών Δεδομένων:

Username & Password: Τα στοιχεία που χρειάζονται για την είσοδο του χρήστη στο σύστημα.

Μήνυμα Σφάλματος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εισάγει λανθασμένα στοιχεία.

Δεδομένα Χρήστη: Τα στοιχεία του χρήστη τα οποία υπάρχουν καταχωρημένα στην βάση σχετικά με όλες τις διατροφές που έχει δημιουργήσει.

Ταυτότητα Χρήστη: Η ταυτότητα που δώθηκε στον χρήστη κατά την δημιουργία του προφίλ του.

Διατροφές: Όλες οι διατροφές που έχει δημιουργήσει ο συγκεκριμένος χρήστης.

Ταυτότητα Επιλεγμένης Διατροφής: Η ταυτότητα της επιλεγμένης διατροφής του χρήστη την οποία επιθυμεί να διορθώσει.

Στοιχεία Διατροφής: Όλα τα στοιχεία της αποθηκευμένης διατροφής από την βάση δεδομένων. Περιλαμβάνει το όνομα της και τα φαγητά από τα οποία αποτελείται.

Νέα Στοιχεία Διατροφής: Τα νέα στοιχεία της διατροφής όπως αυτά προκύπτουν από τις διορθώσεις που κάνει ο χρήστης.

4.2.1.14.2 Προσδιορισμός λογικής διαδικασιών:

Log In: Η εξοσιοδοτημένη είσοδος του χρήστη στο σύστημα.

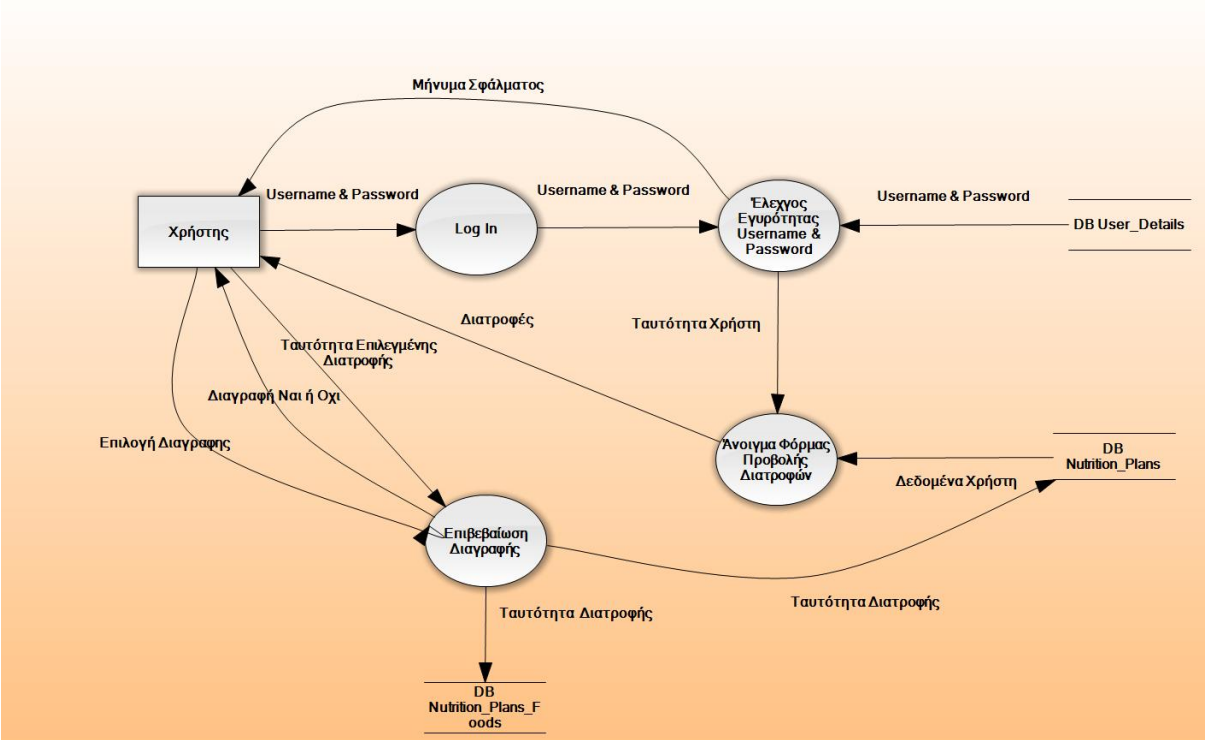
Έλεγχος εγκυρότητας Username & Password: Έλεγχος εάν τα Username & Password βρίσκονται στην ίδια εγγραφή στην βάση δεδομένων.

Άνοιγμα Φόρμας Για Προβολή Διατροφών: Το άνοιγμα της φόρμας για προβολή των διατροφών του συγκεκριμένου χρήστη.

Άνοιγμα Φόρμας Διατροφής: Το άνοιγμα της φόρμας για προβολή συγκεκριμένης διατροφής του συγκεκριμένου χρήστη. Θα εμφανίζονται όλα τα γεύματα όπως τα διάλεξε ο χρήστης όταν την πρωτοδημιούργησε.

Διόρθωση Στοιχείων διατροφής: Ο χρήστης μπορεί να αφαιρέσει και να προσθέσει άλλα φάγητά για να δημιουργήσει έτσι μια ανανεωμένη διατροφή.

4.2.1.15 DFD για την λειτουργία Διαγραφή Διατροφής Χρήστη- Επίπεδο 2



4.2.1.15.1 Εκλέπτυνση Ροών Δεδομένων:

Username & Password: Τα στοιχεία που χρειάζονται για την είσοδο του χρήστη στο σύστημα.

Μήνυμα Σφάλματος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εισάγει λανθασμένα στοιχεία.

Δεδομένα Χρήστη: Τα στοιχεία του χρήστη τα οποία υπάρχουν καταχωρημένα στην βάση σχετικά με όλες τις διατροφές που έχει δημιουργήσει.

Ταυτότητα Χρήστη: Η ταυτότητα που δώθηκε στον χρήστη κατά την δημιουργία του προφίλ του.

Διατροφές: Όλες οι διατροφές που έχει δημιουργήσει ο συγκεκριμένος χρήστης.

Ταυτότητα Επιλεγμένης Διατροφής: Η ταυτότητα της επιλεγμένης διατροφής του χρήστη την οποία επιθυμεί να διορθώσει.

Διαγραφή Ναι ή Οχι: Εμφάνιση του μηνύματος στον χρήστη εαν είναι σίγουρος ότι θέλει να διαγράψει την διατροφή.

Επιλογή Διαγραφής: Η επιλογή του χρήστη. Εαν θέλει να διαγράψει ή όχι την διατροφή.

4.2.1.15.2 Προσδιορισμός λογικής διαδικασιών:

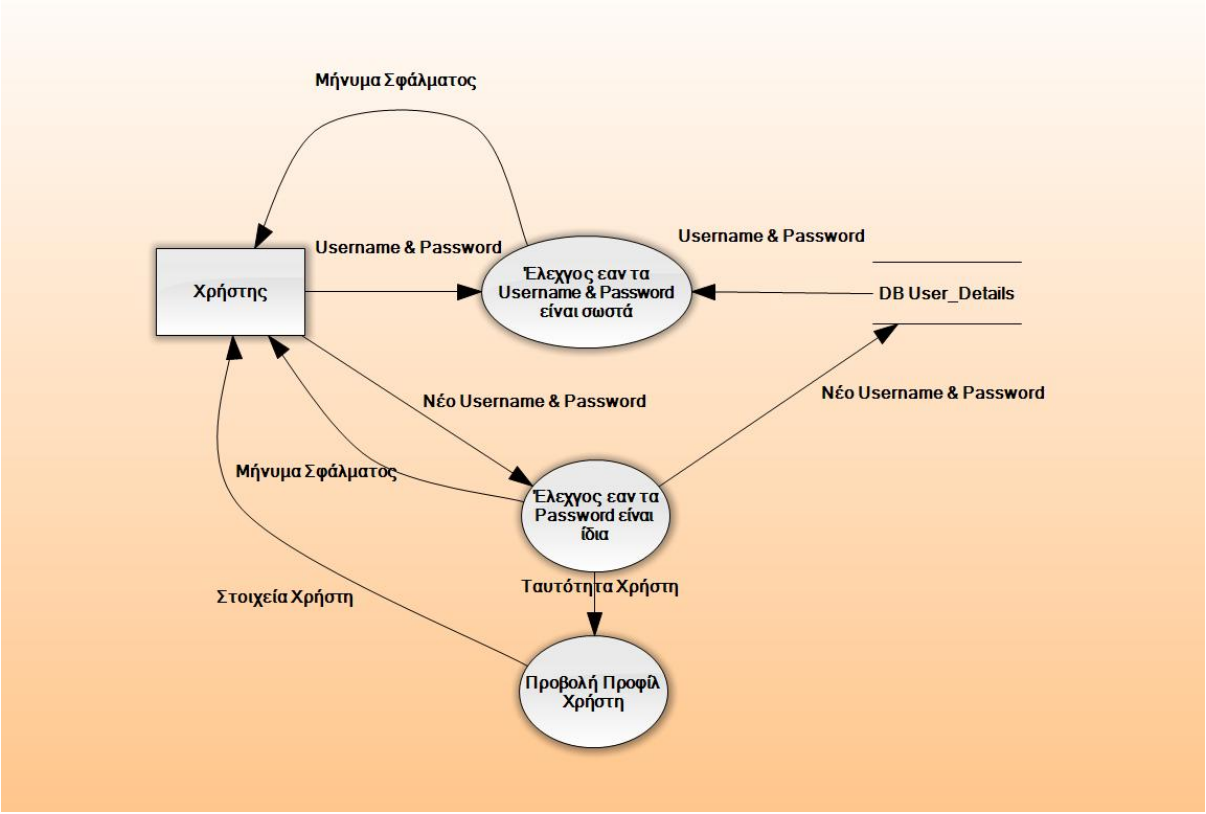
Log In: Η εξοσιοδοτημένη είσοδος του χρήστη στο σύστημα.

Έλεγχος εγκυρότητας Username & Password: Έλεγχος εάν τα Username & Password βρίσκονται στην ίδια εγγραφή στην βάση δεδομένων.

Άνοιγμα Φόρμας Για Προβολή Διατροφών: Το άνοιγμα της φόρμας για προβολή των διατροφών του συγκεκριμένου χρήστη.

Επιβεβαίωση Διαγραφής: Το άνοιγμα της φόρμας επιβεβαίωσης διαγραφής σε περίπτωση που ο χρήστης αλλάξει γνώμη και δεν θέλει να διαγράψει την διατροφή του.

4.2.1.16 DFD για την λειτουργία Αλλαγή Password Χρήστη- Επίπεδο 2



4.2.1.16.1 Εκλέπτυνση Ροών Δεδομένων:

Username & Password: Τα στοιχεία που χρειάζονται για την είσοδο του χρήστη στο σύστημα.

Μήνυμα Σφάλματος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εισάγει λανθασμένα στοιχεία.

Στοιχεία Χρήστη: Τα στοιχεία του χρήστη τα οποία υπάρχουν καταχωρημένα στην βάση σχετικά με όλες τις διατροφές που έχει δημιουργήσει.

Ταυτότητα Χρήστη: Η ταυτότητα που δώθηκε στον χρήστη κατά την δημιουργία του προφίλ του.

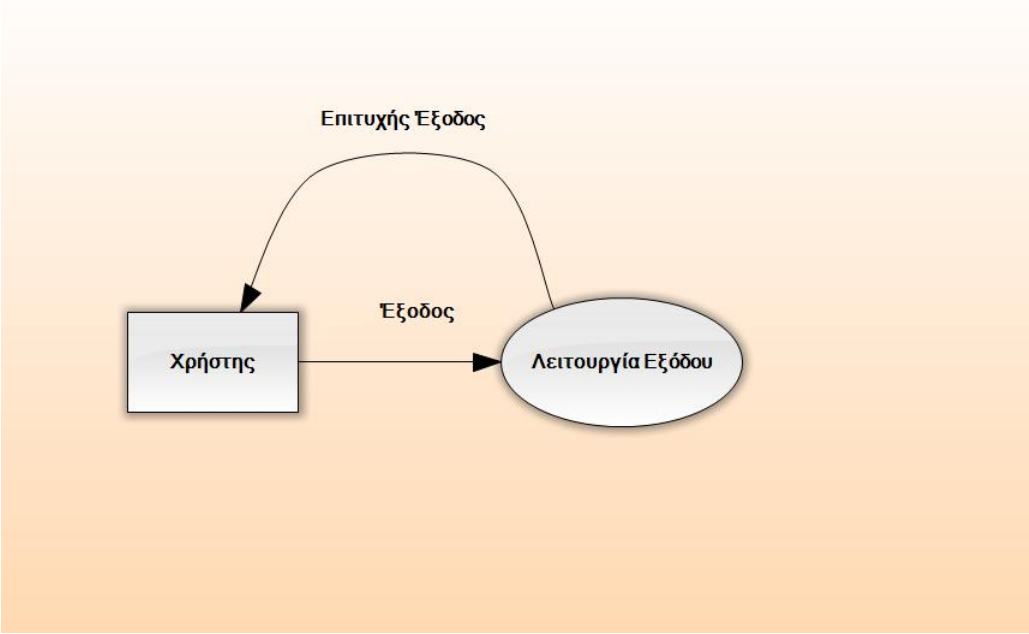
Νέο Username & Password: Τα νέα στοιχεία που χρειάζονται για την είσοδο του χρήστη στο σύστημα.

4.2.1.16.2 Προσδιορισμός λογικής διαδικασιών:

Έλεγχος εαν τα Username & Password είναι σωστά: Έλεγχος εάν τα Username & Password βρίσκονται στην ίδια εγγραφή στην βάση δεδομένων.
Έλεχγος εαν τα Password είναι τα ίδια: Ο χρήστης εισάγει δύο φορές το Password του και ελέχγουμε εαν είναι τα ίδια.

Προβολή Προφίλ Χρήστη: Η προβολή του πλήρες προφίλ του χρήστη με όλα του τα στοιχεία.

4.2.1.17 DFD για την λειτουργία Log Out Χρήστη- Επίπεδο 2



4.2.1.17.1 Εκλέπτυνση Ροών Δεδομένων:

Έξοδος: Η επιλογή του χρήστη για έξοδο από το σύστημα
Επιτυχής Έξοδος: Το μήνυμα που εμφανίζεται στον χρήστη εαν εξέλθει επιτυχώς από το σύστημα.

4.2.1.17.2 Προσδιορισμός λογικής διαδικασιών:

Λειτουργία Εξόδου: Η έξοδος του χρήστη από το σύστημα.

4.2.2 Προσδιορισμός της αποθήκευσης Δεδομένων

4.2.2.1 Λειτουργία: Log In

A/A	Πεδίο	Μορφή
1	Username	Συμβολοσειρά
2	Password	Συμβολοσειρά
3	User_ID	Integer

4.2.2.2 Λειτουργία: Δημιουργία Νέου Χρήστη

A/A	Πεδίο	Μορφή
1	Username	Συμβολοσειρά
2	Password	Συμβολοσειρά
3	Όνομα	Συμβολοσειρά
4	Επώνυμο	Συμβολοσειρά
5	Ηλικία	Integer
6	Βάρος	Decimal
7	Ύψος	Decimal
8	Φύλο	Binary
9	Επίπεδο Άσκησης	Συμβολοσειρά
10	Στόχος	Decimal
11	Επιθυμητό Βάρος	Decimal
12	BMI	Decimal
13	BMR	Decimal
14	Ημερήσιες Θερμίδες	Decimal
15	User_ID	Integer

4.2.2.3 Λειτουργία: Δημιουργία Νέας Διατροφής

A/A	Πεδίο	Μορφή
1	Ταυτότητα Διατροφής	Integer
2	Username	Συμβολοσειρά
3	Password	Συμβολοσειρά
4	User_ID	Integer
5	Ημερομηνία	Date
6	Φαγητό	Συμβολοσειρά
7	Ποσότητα	Decimal
8	Θερμίδες	Decimal
9	Υδατάνθρακες	Decimal
10	Πρωτείνες	Decimal

11	Λίπη	Decimal
12	Γεύμα	Συμβολοσειρά

4.2.2.4 Λειτουργία: Δημιουργία Αυτόματης Διατροφής

Α/Α	Πεδίο	Μορφή
1	Ταυτότητα Διατροφής	Integer
2	Username	Συμβολοσειρά
3	Password	Συμβολοσειρά
4	User_ID	Integer
5	Ημερομηνία	Date
6	Φαγητό	Συμβολοσειρά
7	Ποσότητα	Decimal
8	Θερμίδες	Decimal
9	Υδατάνθρακες	Decimal
10	Πρωτείνες	Decimal
11	Λίπη	Decimal
11	Γεύμα	Συμβολοσειρά

4.2.2.5 Λειτουργία: Εισαγωγή Νέου Φαγητού

Α/Α	Πεδίο	Μορφή
1	Ταυτότητα Φαγητού	Integer
2	Όνομα Φαγητού	Συμβολοσειρά
3	Περιγραφή Φαγητού	Συμβολοσειρά
4	Ομάδα Φαγητού	Συμβολοσειρά
5	Θερμίδες	Decimal
6	Υδατάνθρακες	Decimal
7	Λίπη	Decimal
8	Πρωτείνες	Decimal
9	Φυτικές	Decimal
10	Νάτριο	Decimal

4.2.2.6 Λειτουργία: Εισαγωγή Νέου Φαγητού από Διάφορα Υλικά

Α/Α	Πεδίο	Μορφή
1	Ταυτότητα Φαγητού	Integer
2	Όνομα Φαγητού	Συμβολοσειρά
3	Περιγραφή Φαγητού	Συμβολοσειρά
4	Ομάδα Φαγητού	Συμβολοσειρά
5	Θερμίδες	Decimal
6	Υδατάνθρακες	Decimal
7	Λίπη	Decimal
8	Πρωτείνες	Decimal

4.2.2.7 Λειτουργία: Προβολή Προφίλ Χρήστη

Α/Α	Πεδίο	Μορφή
1	Username	Συμβολοσειρά
2	Password	Συμβολοσειρά
3	User_ID	Integer

4.2.2.8 Λειτουργία: Διόρθωση Προφίλ Χρήστη

Α/Α	Πεδίο	Μορφή
1	Username	Συμβολοσειρά
2	Password	Συμβολοσειρά
3	Όνομα	Συμβολοσειρά
4	Επώνυμο	Συμβολοσειρά
5	Ηλικία	Integer
6	Βάρος	Decimal
7	Ύψος	Decimal
8	Φύλο	Binary
9	Επίπεδο Άσκησης	Συμβολοσειρά
10	Στόχος	Decimal
11	Επιθυμητό Βάρος	Decimal
12	BMI	Decimal
13	BMR	Decimal
14	Ημερήσιες Θερμίδες	Decimal
15	User_ID	Integer

4.2.2.9 Λειτουργία: Διαγραφή Προφίλ Χρήστη

Α/Α	Πεδίο	Μορφή
1	Username	Συμβολοσειρά
2	Password	Συμβολοσειρά
3	User_ID	Integer

4.2.2.10 Λειτουργία: Προβολή Διατροφών Χρήστη

Α/Α	Πεδίο	Μορφή
1	Username	Συμβολοσειρά
2	Password	Συμβολοσειρά
3	User_ID	Integer
4	Nutrition_IDs	Πίνακας Integer

4.2.2.11 Λειτουργία: Διόρθωση Διατροφής Χρήστη

Α/Α	Πεδίο	Μορφή
1	Ταυτότητα Διατροφής	Integer
2	Username	Συμβολοσειρά
3	Password	Συμβολοσειρά
4	User_ID	Integer

5	Ημερομηνία	Date
6	Φαγητό	Συμβολοσειρά
7	Ποσότητα	Decimal
8	Θερμίδες	Decimal
9	Υδατάνθρακες	Decimal
10	Πρωτείνες	Decimal
11	Λίπη	Decimal
12	Γεύμα	Συμβολοσειρά

4.2.2.12 Λειτουργία: Διαγραφή Διατροφής Χρήστη

A/A	Πεδίο	Μορφή
1	Username	Συμβολοσειρά
2	Password	Συμβολοσειρά
3	User_ID	Integer
4	Nutrition_ID	Integer

4.2.2.13 Λειτουργία: Αλλαγή Password Χρήστη

A/A	Πεδίο	Μορφή
1	Username	Συμβολοσειρά
2	Password	Συμβολοσειρά
3	User_ID	Integer
4	Νέο Password	Συμβολοσειρά

4.2.3.Ορισμός Φυσικών Πόρων

4.2.3.1 Πόροι για Χρήστες

Όνομα:	Πρόσβαση Χρήστη
Οργάνωση:	Indexed
Μέσο Αποθήκευσης:	Βάση Δεδομένων
Εγγραφές:	Ταυτότητα, Username, Password

Όνομα:	Στοιχεία Χρήστη
Οργάνωση:	Indexed
Μέσο Αποθήκευσης:	Βάση Δεδομένων
Εγγραφές:	Ταυτότητα, Όνομα, Επίθετο, Φύλο, Ηλικία, Βάρος, Ύψος, Επίπεδο Άσκησης, Επιθυμητό Βάρος, Στόχος, BMR, BMI, Εισηγούμενες Θερμίδες

4.2.3.2 Πόροι για Φαγητά

Όνομα:	Πληροφορίες Φαγητών
Οργάνωση:	Indexed
Μέσο Αποθήκευσης:	Βάση Δεδομένων
Εγγραφές:	Ταυτότητα Φαγητού, Ομάδα Φαγητών, Όνομα Φαγητού, Περιγραφή Φαγητού

Όνομα:	Θρεπτικά Στοιχεία Φαγητών
Οργάνωση:	Indexed
Μέσο Αποθήκευσης:	Βάση Δεδομένων
Εγγραφές:	Ταυτότητα Φαγητού, Ταυτότητα Θρεπτικού Στοιχείου, Τιμή Θρεπτικού Στοιχείου.

Όνομα:	Πληροφορίες για Μερτήσεις Φαγητών
Οργάνωση:	Indexed
Μέσο Αποθήκευσης:	Βάση Δεδομένων
Εγγραφές:	Ταυτότητα Φαγητού, Ταυτότητα Μέτρησης, Ποσότητα, Όνομα Μέρτησης, Βάρος Μέρτησης

4.2.3.3 Πόροι για Διατροφές

Όνομα:	Διατροφές Χρηστών
Οργάνωση:	Indexed
Μέσο Αποθήκευσης:	Βάση Δεδομένων
Εγγραφές:	Ταυτότητα Χρήστη, Ταυτότητα Διατροφής

Όνομα:	Φαγητά Διατροφών
Οργάνωση:	Indexed
Μέσο Αποθήκευσης:	Βάση Δεδομένων
Εγγραφές:	Ταυτότητα Διατροφής, Ημερομηνία, Φαγητό, Ποσότητα, Γεύμα, Θερμίδες, Υδατάνθρακες, Πρωτείνες, Λίπη

4.2.4 Ορισμός Προδιαγραφών για Input / Output

4.2.4.1 Input / Output για Χρήστες

Πρόσβαση Χρήστη

Είσοδος	Τιμές Πεδίου Εισόδου	Έξοδος
Username	Σειρά από 30 γράμματα του αγγλικού αλφαβήτου και αριθμοί	Το User_ID το οπόιο θα χρησιμοποιηθεί για εκτέλεση άλλων MySQL queries.
Password	Σειρά από 30 γράμματα του αγγλικού αλφαβήτου και αριθμοί	
User_ID	Ακέραιος Αριθμός	

Στοιχεία Χρήστη

Είσοδος	Τιμές Πεδίου Εισόδου	Έξοδος
Όνομα	Σειρά από 30 γράμματα του αγγλικού αλφαβήτου και αριθμοί	Οι πληροφορίες του χρήστη οι οποίες είναι αποθηκευμένες στην βάση και οι οποίες εξάγονται μέσω των MySQL queries στην κάθε λειτουργία του συστήματος.
Επώνυμο	Σειρά από 30 γράμματα του αγγλικού αλφαβήτου και αριθμοί	
Ηλικία	Ακέραιος Αριθμός	
Βάρος	Αριθμός με 2 Δεκαδικά Ψηφία	
Ύψος	Αριθμός με 2 Δεκαδικά Ψηφία	
Φύλο	Χαρακτήρας (M/F)	
Επίπεδο Άσκησης	Σειρά από 30 γράμματα του αγγλικού αλφαβήτου και αριθμοί	
Στόχος	Σειρά από 30 γράμματα του αγγλικού αλφαβήτου και αριθμοί	
Επιθυμητό Βάρος	Αριθμός με 2 Δεκαδικά Ψηφία	
BMI	Αριθμός με 2 Δεκαδικά Ψηφία	
BMR	Αριθμός με 2 Δεκαδικά Ψηφία	
Ημερήσιες Θερμίδες	Αριθμός με 2 Δεκαδικά Ψηφία	
User_ID	Ακέραιος Αριθμός	

4.2.4.2 Input / Output για Φαγητά

Για τα δεδομένα των φαγητών το σύστημα θα κάνει χρήση πινάκων από την βάση Δεδομένων του Υπουργείου Γεωργίας και Υγείας των ΗΠΑ (USDA). Η βάση αυτή αποτελέι την βασική πηγή δεδομένων για την σύνθεση των φαγητών αφού αποτελεί το θεμέλιο για πολλές άλλες βάσεις δεδομένων που σχεδιάστηκαν για την σύνθεση φαγητών.

Οι πίνακες που θα χρησιμοποιήσω είναι οι εξής:

Πληροφορίες Φαγητών

Είσοδος	Τιμές Πεδίου Εισόδου	Έξοδος
Ταυτότητα Φαγητού	Ακέραιος Αριθμός	Το ID του φαγητού και γενικές πληροφορίες οι οποίες είναι αποθηκευμένες στην βάση και οι οποίες εξάγονται μέσω των MySQL queries στην ανάλογη λειτουργία του συστήματος.
Ομάδα Φαγητών	Ακέραιος Αριθμός	
Όνομα Φαγητού	Σειρά από 30 γράμματα του αγγλικού αλφαβήτου και αριθμοί	
Περιγραφή Φαγητού	Σειρά από 100 γράμματα του αγγλικού αλφαβήτου και αριθμοί	

Θρεπτικά Στοιχεία Φαγητών

Είσοδος	Τιμές Πεδίου Εισόδου	Έξοδος
Ταυτότητα Φαγητού	Ακέραιος Αριθμός	Οι τιμές του θρεπτικών συστατικών ενός φαγητού οι οποίες είναι αποθηκευμένες στην βάση και οι οποίες εξάγονται μέσω των MySQL queries στην ανάλογη λειτουργία του συστήματος.
Ταυτότητα Θρεπτικού Στοιχείου	Ακέραιος Αριθμός	
Τιμή Θρεπτικού Στοιχείου	Αριθμός με 2 Δεκαδικά Ψηφία	

Πληροφορίες για Μερτήσεις Φαγητών

Είσοδος	Τιμές Πεδίου Εισόδου	Έξοδος
Ταυτότητα Φαγητού	Ακέραιος Αριθμός	Οι τιμές του βάρους των μερτήσεων για συγκεκριμένο φαγητό οι οποίες είναι αποθηκευμένες στην βάση και οι οποίες εξάγονται μέσω των MySQL queries στην ανάλογη λειτουργία του συστήματος.
Ταυτότητα Μέτρησης	Ακέραιος Αριθμός	
Ποσότητα	Σειρά από 30 γράμματα του αγγλικού αλφαβήτου και αριθμοί	
Όνομα Μέρτησης		
Βάρος Μέρτησης	Σειρά από 100 γράμματα του αγγλικού αλφαβήτου και αριθμοί	

4.2.4.3 Input / Output για Διατροφές

Διατροφές Χρηστών

Είσοδος	Τιμές Πεδίου Εισόδου	Έξοδος
Ταυτότητα Διατροφής	Ακέραιος Αριθμός	Το ID της διατροφής το οπόιο θα χρησιμοποιηθεί για εκτέλεση άλλων MySQL queries.
Ταυτότητα Χρήστη	Ακέραιος Αριθμός	

Φαγητά Διατροφών

Είσοδος	Τιμές Πεδίου Εισόδου	Έξοδος
Ταυτότητα Διατροφής	Ακέραιος Αριθμός	Τα φαγητά της κάθε διατροφής τα οποία είναι αποθηκευμένες στην βάση και τα οποία εξάγονται μέσω των MySQL queries στην ανάλογη λειτουργία του συστήματος.
User_ID	Ακέραιος Αριθμός	
Ημερομηνία	Ημερομηνία	
Φαγητό	Σειρά από 30 γράμματα του αγγλικού αλφαβήτου και αριθμοί	
Ποσότητα	Αριθμός με 2 Δεκαδικά Ψηφία	
Θερμίδες	Αριθμός με 2 Δεκαδικά Ψηφία	
Υδατάνθρακες	Αριθμός με 2 Δεκαδικά Ψηφία	
Πρωτείνες	Αριθμός με 2 Δεκαδικά Ψηφία	
Λίπη	Αριθμός με 2 Δεκαδικά Ψηφία	
Γεύμα	Σειρά από 30 γράμματα του αγγλικού αλφαβήτου και αριθμοί	

4.2.5 Εκτιμήσεις για το μέγεθος των δεδομένων Input / Output

Τα δεδομένα θα αποθηκεύονται όλα στην βάση δεδομένων. Το μέγεθος της βάσης αρχικά, είναι περίπου 57MB.

4.2.6 Απαιτήσεις σε Hardware

Δεν υπάρχουν ιδιαίτερες απαιτήσεις σε Hardware αφού το σύστημα θα τρέχει στον υπολογιστή του χρήστη.

4.2.6.1 Αποθηκευτικές Μονάδες

Όλα τα δεδομένα του συστήματος θα αποθηκεύονται στον υπολογιστή του χρήστη. Έτσι πρέπει να υπάρχει πάντα χώρος ελεύθερος.

4.2.6.2 Συσκευές Εισόδου Δεδομένων

Για την είσοδο των δεδομένων ο χρήστης θα χρησιμοποιεί το ποντίκι και το πληκτρολόγιο του υπολογιστή του.

4.2.6.3 Συσκευές Εξόδου Δεδομένων

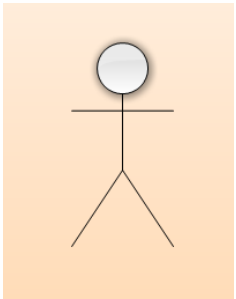
Όλες οι πληροφορίες των λειτουργιών του συστήματος θα παρουσιάζονται στην οθόνη του υπολογιστή.

4.3 Αντικειμενοστραφής Ανάλυση

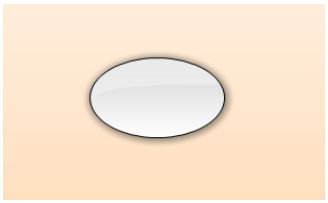
4.3.1 Δημιουργία Διαγράμματος Use Case

Το Use Case Diagram είναι μια γραφική αναπαράσταση της λειτουργίας του συστήματος η οποία αποτελείται από τις λειτουργικότητες του συστήματος (use cases) και τα άτομα ή μέρη του υλικού που αλληλεπιδρά με το σύστημα(Actors). Υπάρχουν δύο είδη σχέσεων που συνδέουν τα μέρη του διαγράμματος: η include και η extend. Η χρήση της include γίνεται όταν ένα use case χρησιμοποιεί ένα άλλο το οποίο τρέχει ταυτόχρονα μαζί του ενώ της extend όταν το use case επεκτείνει την λειτουργία του σε ένα άλλο το οποίο δεν είναι απαραίτητο να τρέξει.

Τα σύμβολα που θα χρησιμοποιηθούν στο διάγραμμα είναι τα εξής:



Actor ή Χρήστης του συστήματος

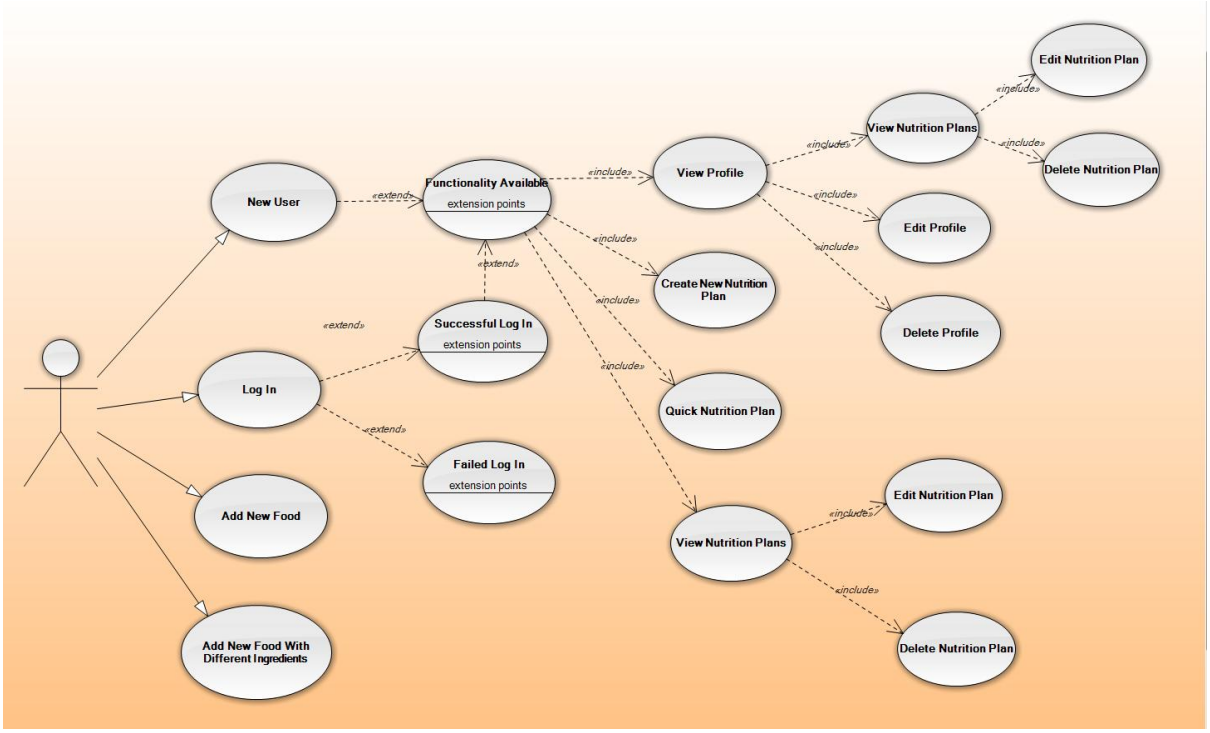


Use Case

Σύνδεση μεταξύ Use Case

Σύνδεση μεταξύ Use Case και Actors

4.3.1.1 Το διάγραμμα Use Case για το συγκεκριμένο σύστημα έχει ώς εξής:

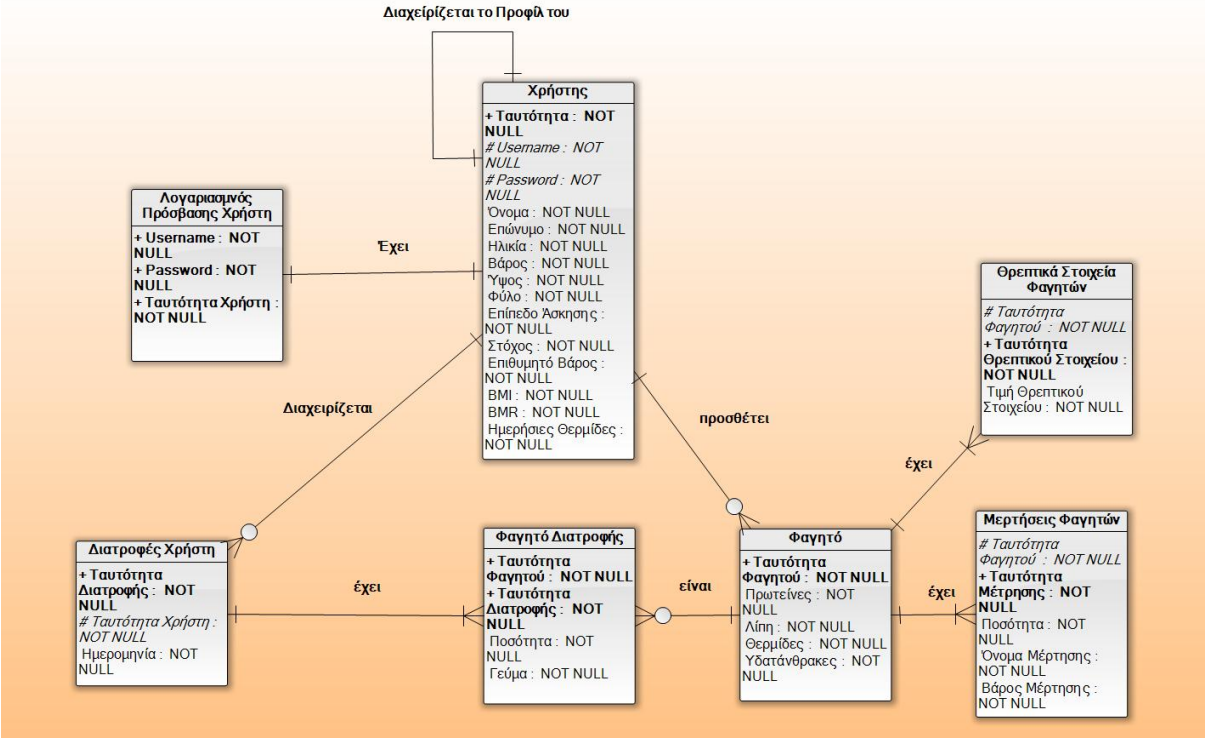


Ο χρήστης του συστήματος μπορεί να δημιουργήσει ένα νέο προφίλ, να προσθέσει ένα φαγητό στην βάση, ή να κάνει Login για να χρησιμοποιήσει τις διατροφικές λειτουργίες του συστήματος. Στην προσπάθεια του να κάνει Login μπορεί να κάνει καποιο λάθος και να μην γίνει αποδεκτή η εισδοχή του. Εαν συμβεί αυτό θα ενημερωθεί ο χρήστης για να εισάγει ξανά τα στοιχεία του. Όταν γίνει επιτυχείς το Login τότε μπορεί να δεί το προφίλ του, να δημιουργήσει νέα διατροφή αυτόματα ή "χειροκίνητα". Μπορεί επίσης να δεί ποιες διατροφές έχει δημιουργήσει και πότε. Μπορεί να επιλέξει μια διατροφή και να την διορθώσει ή και να την διαγράψει όπως μπορεί να κάνει και με το προφίλ του.

4.3.2 Δημιουργία Entity Relationship Diagram (ERD)

Το διάγραμμα ERD είναι μια γραφική αναπαράσταση δεδομένων. Πρόκειται για μια μέθοδο μοντελοποίησης μιας βάσης δεδομένων για δημιουργία του μοντέλου των δεδομένων ενός συστήματος.

Το ERD διάγραμμα για το σύστημα είναι το εξής:



Κεφάλαιο 5

Σχεδιασμός Συστήματος

5.1 Εισαγωγή
5.2 Αντικειμενοστραφής Σχεδίαση

5.1 Εισαγωγή

Στην φάση της σχεδίασης γίνεται μια πιο αναλυτική περιγραφή των όσων συζητήθηκαν στις φάσεις των απαιτήσεων και των προδιαγραφών για να γίνει η φάση της υλοποίησης πολύ πιο εύκολη.
Συγκεριμένα συζητούνται θέματα όπως η αρχιτεκτονική του συστήματος, οι οντότητες που το αποτελούν και τα δεδομένα που χρειάζεται το σύστημα για να ικανοποιεί τις απαιτήσεις.
Η φάση της σχεδίασης αποσκοπεί επίσης στην καλή συντήρηση αφού ένα καλά σχεδιασμένο σύστημα θα μπορεί να συντηρηθεί πολύ πιο εύκολα.
Σε αυτό το στάδιο θα δημιουργηθούν τα διαγράμματα συνεργασίας, τα ακολουθιακά διαγράμματα και το class diagram του συστήματος.

5.2 Αντικειμενοστραφής Σχεδίαση

5.2.1 Διαγράμματα Συνεργασίας και Ακολουθιακά Διαγράμματα

Τα διαγράμματα συνεργασίας και τα ακολουθιακά διαγράμματα είναι διαγράμματα τα οποία παρουσιάζουν το πως ανταλλάζονται τα δεδομένα μεταξύ των αντικειμένων του συστήματος για κάθε λειτουργία. Τα διαγράμματα συνεργασίας δίνουν έμφαση στις σχέσεις μεταξύ των αντικειμένων ενώ τα ακολουθιακά διαγράμματα δίνουν έμφαση στην ακριβή χρονολογική ακολουθία των μηνυμάτων. Είναι πιο χρήσιμα δηλαδή για την σειρά εμφάνισης των γεγονότων μιας λειτουργίας.

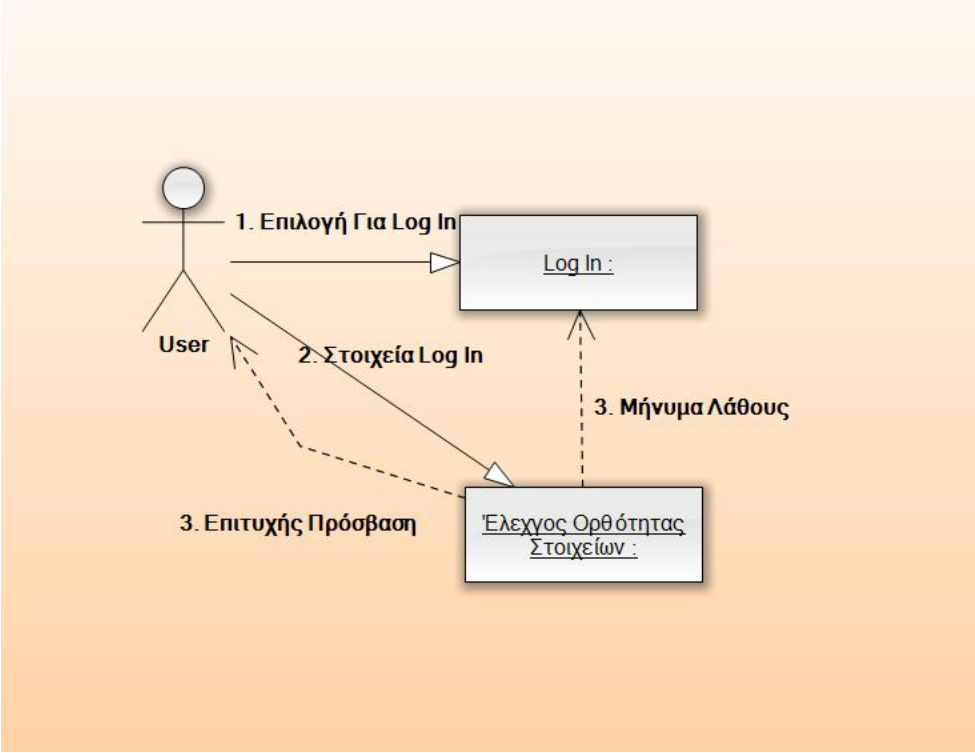
Παρακάτω παρουσιάζονται τα διαγράμματα συνεργασίας και τα ακολουθιακά διαγράμματα κάθε λειτουργίας του συστήματος.

5.2.1.1 Λειτουργία: Log In

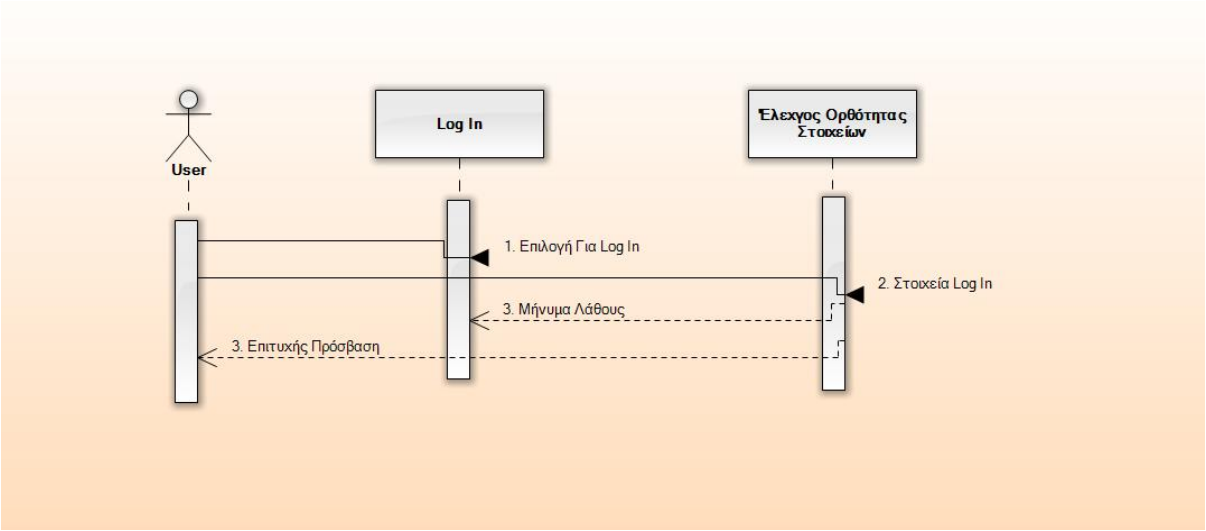
Κατά την λειτουργία του Log In γίνονται τα εξής βήματα:

1. Επιλογή του χρήστη για Log In ή για κάποια άλλη λειτουργία (Προβολή Προφίλ, Δημιουργία Διατροφής, Χρήση Αυτόματου Εργαλείου Δημιουργίας Διατροφής, Προβολή Διατροφών)
2. Εισαγωγή Username και Password
3. Επιβεβαίωση Στοιχείων Χρήστη.
4. Είσοδος στο σύστημα ή στην λειτουργία

Διάγραμμα Συνεργασίας



Ακολουθιακό Διάγραμμα

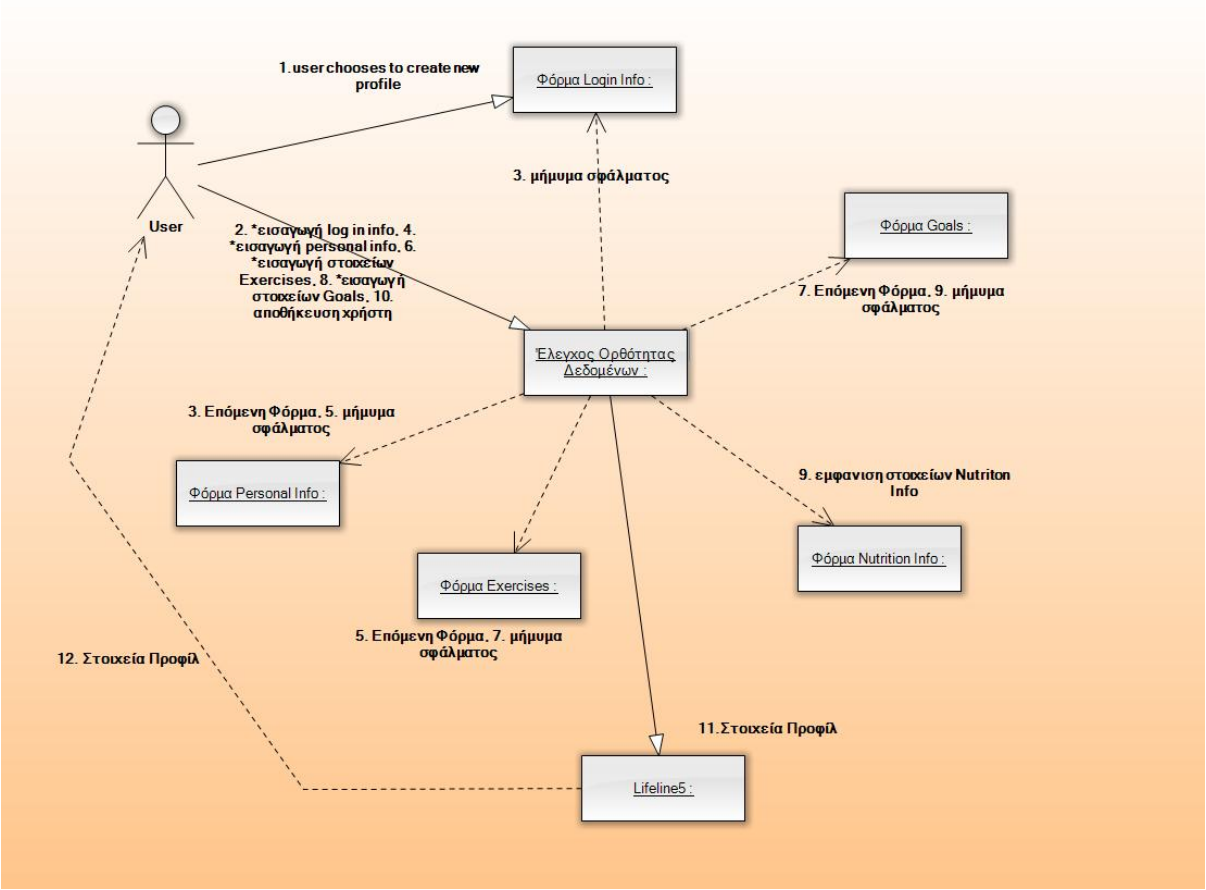


5.2.1.2 Λειτουργία: Δημιουργία Νέου Χρήστη

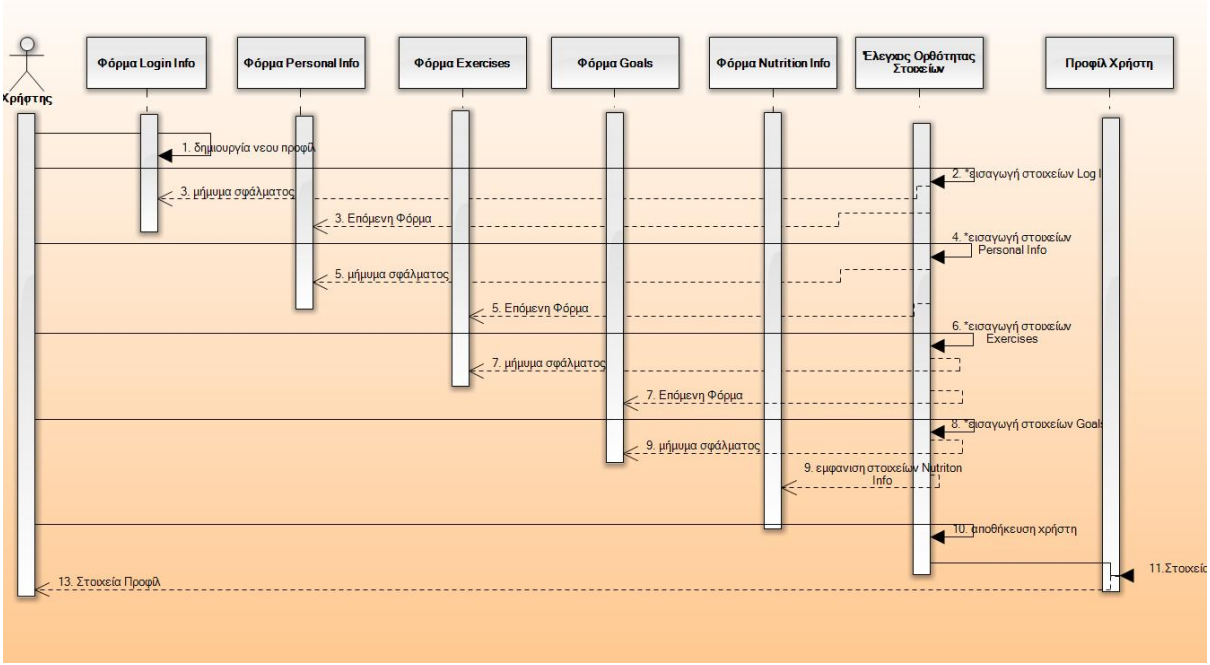
Κατά την λειτουργία της Δημιουργίας Νέου Χρήστη γίνονται τα εξής βήματα:

1. Επιλογή του χρήστη για δημιουργία νέου Προφίλ Χρήστη.
2. Εισαγωγή των στοιχείων για Log In (Username, Password και Confirm Password)
3. Έλεγχος εαν τα στοιχεία δεν υπάρχουν ήδη στην βάση.
4. Ξανά εισαγωγή εαν ο έλεγχος επισημάνει λάθη
5. Εισαγωγή Προσωπικών Στοιχείων (Όνομα, Επίθετο, Ηλικία, Φύλο, Βάρος, Ύψος)
6. Έλεγχος εαν συμπληρώθηκαν ορθά
7. Ξανά εισαγωγή εαν ο έλεγχος επισημάνει λάθη
8. Εισαγωγή Επιπέδου άσκησης
9. Εισαγωγή Στόχου και επιθυμητού Βάρους
10. Έλεγχος εαν συμπληρώθηκαν ορθά
11. Ξανά εισαγωγή εαν ο έλεγχος επισημάνει λάθη
12. Υπολογισμός BMR, BMI και ημερήσιων Θερμίδων
13. Επιλογή για αποθήκευση νέου Χρήστη
14. Προβολή Προφίλ Χρήστη

Διάγραμμα Συνεργασίας



Ακολουθιακό Διάγραμμα



5.2.1.3 Λειτουργία: Δημιουργία Διατροφής

Υπάρχουν 2 τρόποι να γίνει αυτή η λειτουργία. Το ακουλουθιακό διάγραμμα και το διάγραμμα συνεργασίας που ακολουθεί περιγράφει τον πρώτο από τους δύο τρόπους. Ο δεύτερος τρόπος γίνεται με όμοιο τρόπο.

Κατά την λειτουργία της Δημιουργίας Διατροφής γίνονται τα εξής βήματα:

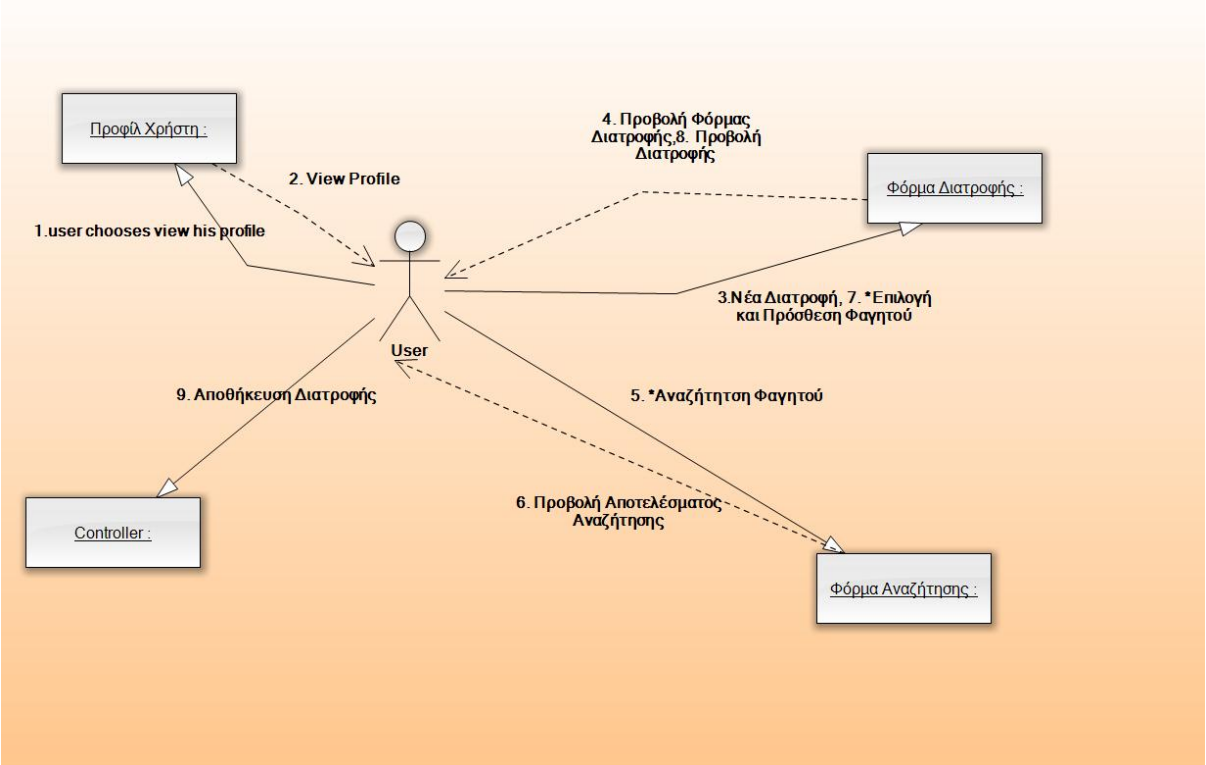
Α Τρόπος:

1. Επιλογή του χρήστη για Προβολής του Προφίλ του.
2. Προβολή Προφίλ
3. Επιλογή του χρήστη για δημιουργία νέας Διατροφής
4. Αναζήτηση Φαγητών για γεύματα
5. Επιλογή Φαγητών Ποσότητας και Γεύμα στο οποίο θέλουν να προσθέσουν το φαγητό
6. Πρόσθεση Επιλεγμενου Φαγητού στο Πρόγραμμα
7. Επανάληψη των βημάτων 2-4 όσες φορές επιθυμεί ο χρήστης
8. Αποθήκευση Διατροφής

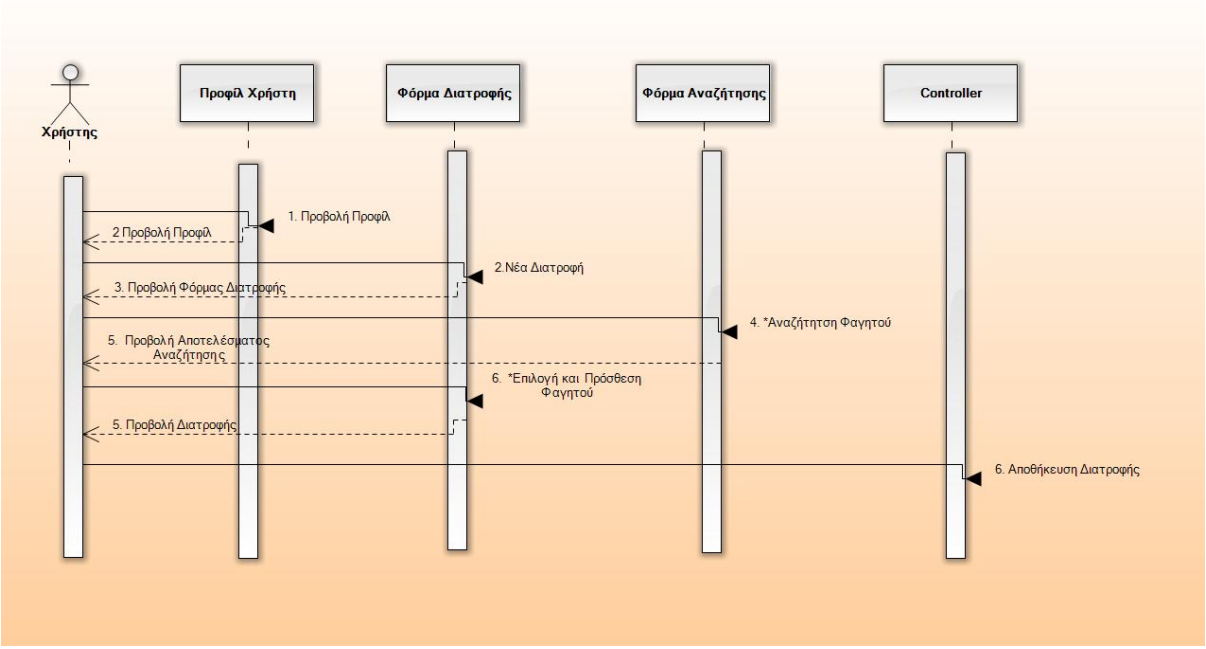
Β Τρόπος:

1. Επιλογή του χρήστη για δημιουργία νέας Διατροφής
2. Αναζήτηση Φαγητών για γεύματα
3. Επιλογή Φαγητών Ποσότητας και Γεύμα στο οποίο θέλουν να προσθέσουν το φαγητό
4. Πρόσθεση Επιλεγμένου Φαγητού στο Πρόγραμμα
5. Επανάληψη των βημάτων 2-4 όσες φορές επιθυμεί ο χρήστης
6. Αποθήκευση Διατροφής

Διάγραμμα Συνεργασίας



Ακολουθιακό Διάγραμμα



5.2.1.4 Λειτουργία: Δημιουργία Αυτόματης Διατροφής

Υπάρχουν 2 τρόποι να γίνει αυτή η λειτουργία. Το ακουλουθιακό διάγραμμα και το διάγραμμα συνεργασίας που ακολουθεί περιγράφει τον πρώτο από τους δύο τρόπους. Ο δεύτερος τρόπος γίνεται με όμοιο τρόπο.

Κατά την λειτουργία της Δημιουργίας Αυτόματης Διατροφής γίνονται τα εξής βήματα:

Α Τρόπος:

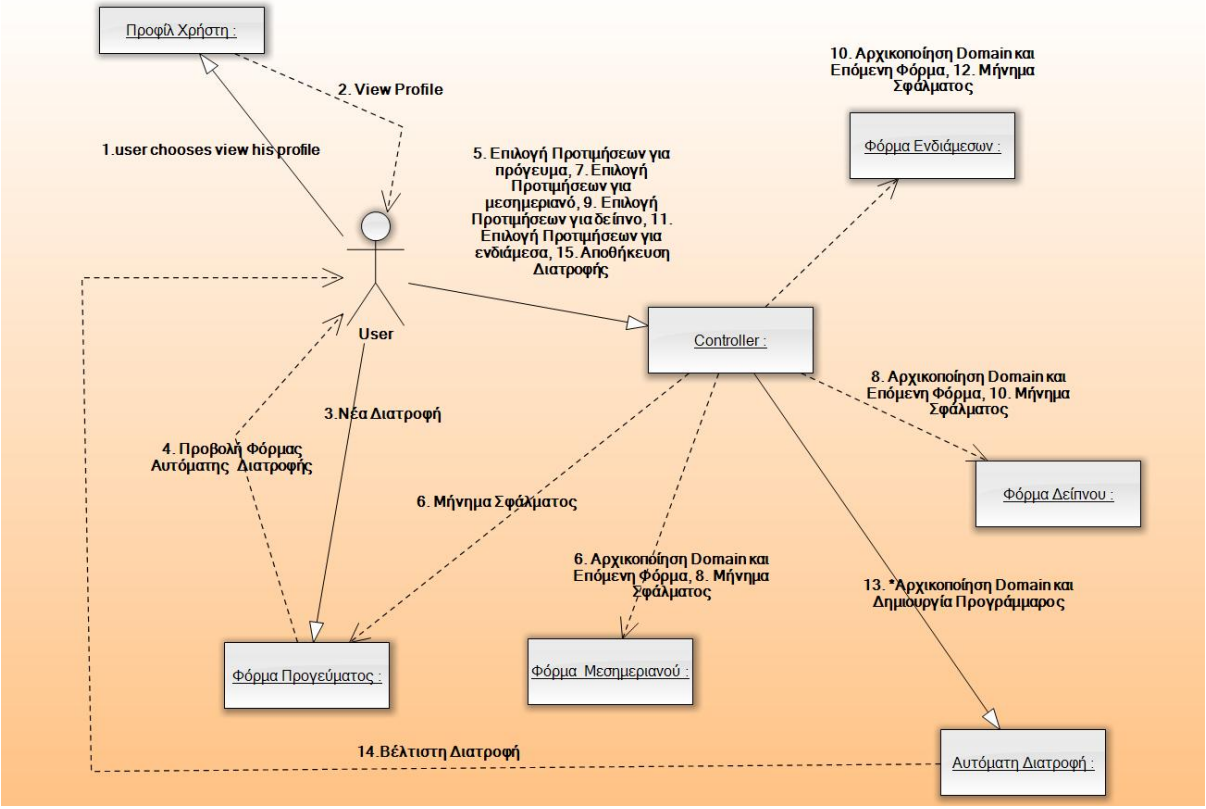
1. Επιλογή του χρήστη για Προβολής του Προφίλ του.
2. Προβολή Προφίλ
3. Επιλογή του χρήστη για δημιουργία νέας Αυτόματης Διατροφής
4. Επιλογή Προτιμήσεων για πρόγευμα
5. Έλεγχος για έγκυρες επιλογές
6. Αρχικοποίηση Domain για πρόγευμα
7. Επιλογή Προτιμήσεων για μεσημεριανό
8. Έλεγχος για έγκυρες επιλογές
9. Αρχικοποίηση Domain για μεσημεριανό
10. Επιλογή Προτιμήσεων για δείπνο
11. Έλεγχος για έγκυρες επιλογές
12. Αρχικοποίηση Domain για δείπνο
13. Επιλογή Προτιμήσεων για ενδιάμεσα
14. Έλεγχος για έγκυρες επιλογές
15. Αρχικοποίηση Domain για ενδιάμεσα
16. Δημιουργία Προγράμματος Ικανοποίησης Περιορισμών
17. Υπολογισμός Βέλτιστης Διατροφής
18. Έαν δεν υπάρχει λύση Ξανά-αρχικοποιούνται τα Domain με διαφορετικές τιμές
19. Επανάληψη βημάτων 15-16 μέχρι να βρεθεί λύση.

Β Τρόπος:

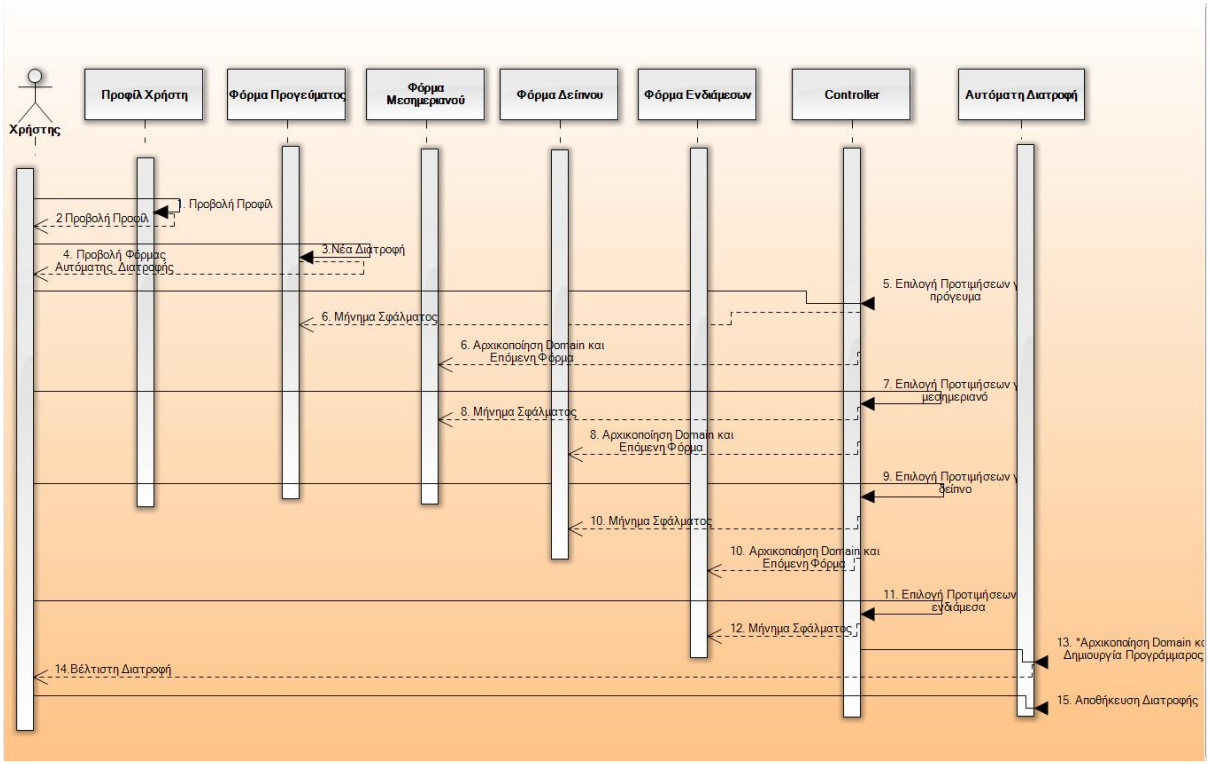
1. Επιλογή του χρήστη για δημιουργία νέας Αυτόματης Διατροφής
2. Επιλογή Προτιμήσεων για πρόγευμα
3. Έλεγχος για έγκυρες επιλογές
4. Αρχικοποίηση Domain για πρόγευμα

5. Επιλογή Προτιμήσεων για μεσημεριανό
6. Έλεγχος για έγκυρες επιλογές
7. Αρχικοποίηση Domain για μεσημεριανό
8. Επιλογή Προτιμήσεων για δείπνο
9. Έλεγχος για έγκυρες επιλογές
10. Αρχικοποίηση Domain για δείπνο
11. Επιλογή Προτιμήσεων για ενδιάμεσα
12. Έλεγχος για έγκυρες επιλογές
13. Αρχικοποίηση Domain για ενδιάμεσα
14. Δημιουργία Προγράμματος Ικανοποίησης Περιορισμών
15. Υπολογισμός Βέλτιστης Διατροφής
16. Εάν δεν υπάρχει λύση Ξανά-αρχικοποιούνται τα Domain με διαφορετικές τιμές
17. Επανάληψη βημάτων 15-16 μέχρι να βρεθεί λύση.

Διάγραμμα Συνεργασίας



Ακολουθιακό Διάγραμμα

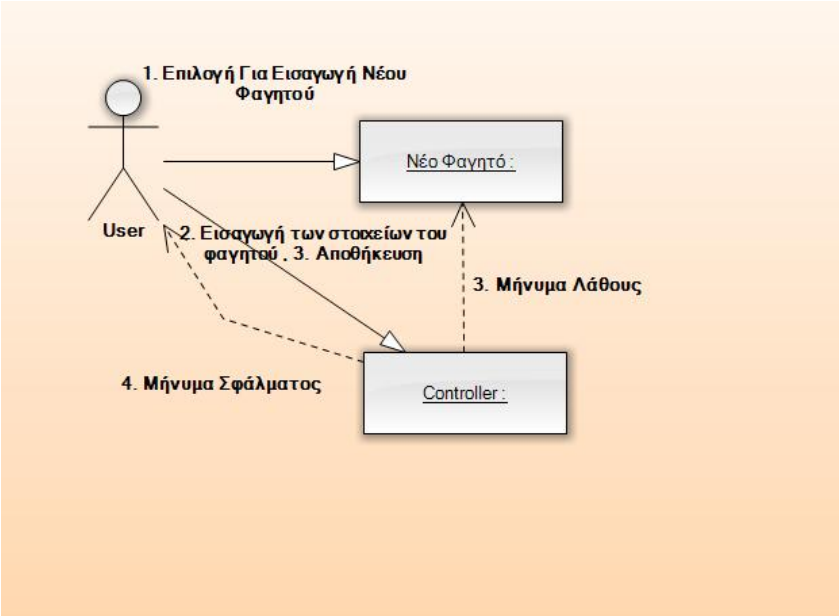


5.2.1.5 Λειτουργία: Εισαγωγή Νέου Φαγητού

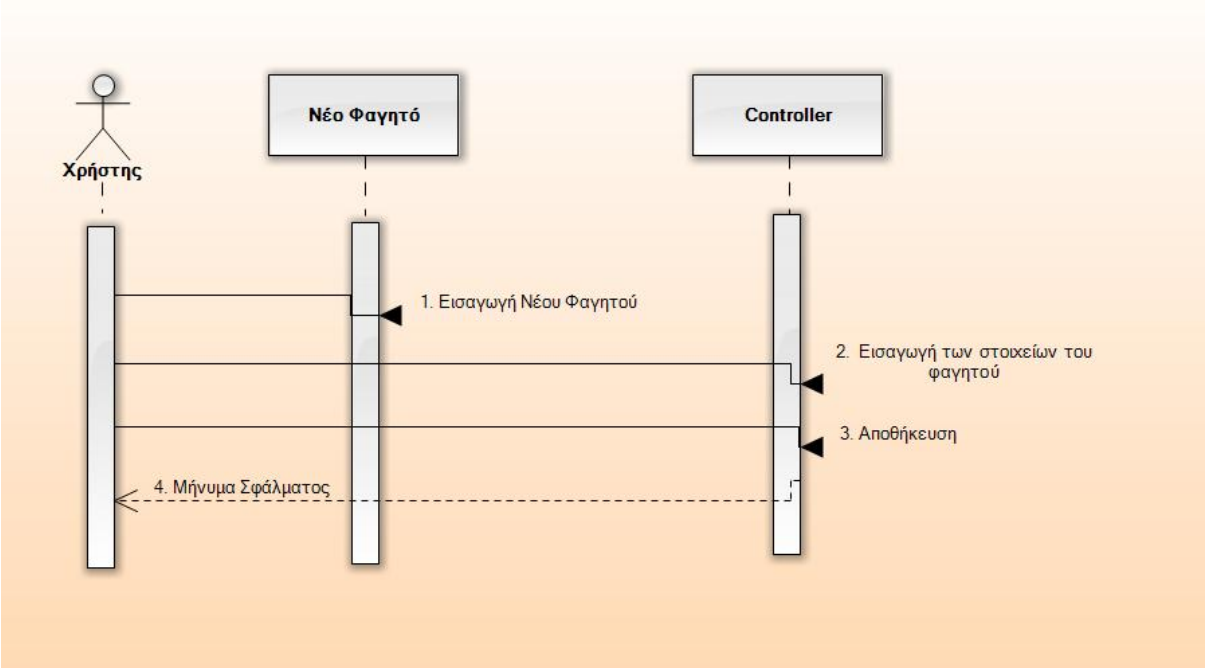
Κατά την λειτουργία της Εισαγωγής Νέου Φαγητού γίνονται τα εξής βήματα:

1. Επιλογή του χρήστη για Εισαγωγή Νέου Φαγητού.
2. Εισαγωγή των στοιχείων του φαγητού (Όνομα, Περιγραφή, και θρεπτική αξία στα 100 gr)
3. Επιλογή για αποθήκευση φαγητού στην βάση
4. Έλεχγος εάν τα στοιχεία δεν υπάρχουν ήδη στην βάση.
5. Αποθήκευση

Διάγραμμα Συνεργασίας



Ακολουθιακό Διάγραμμα



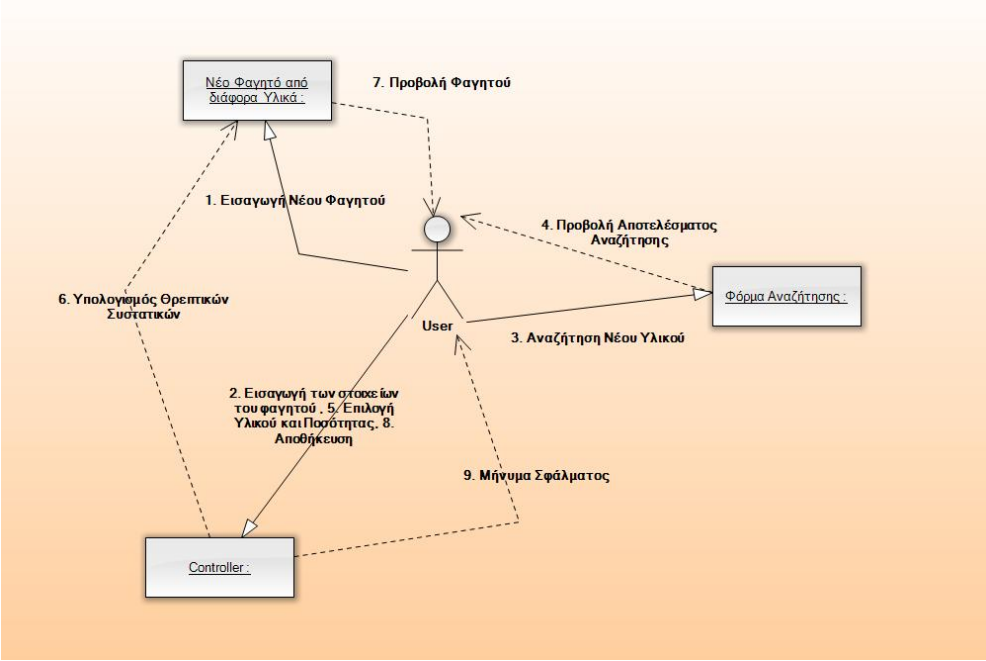
5.2.1.6 Λειτουργία: Εισαγωγή Νέου Φαγητού από Διάφορα Υλικά

Κατά την λειτουργία της Εισαγωγής Νέου Φαγητού από Διάφορα Υλικά γίνονται τα εξής βήματα:

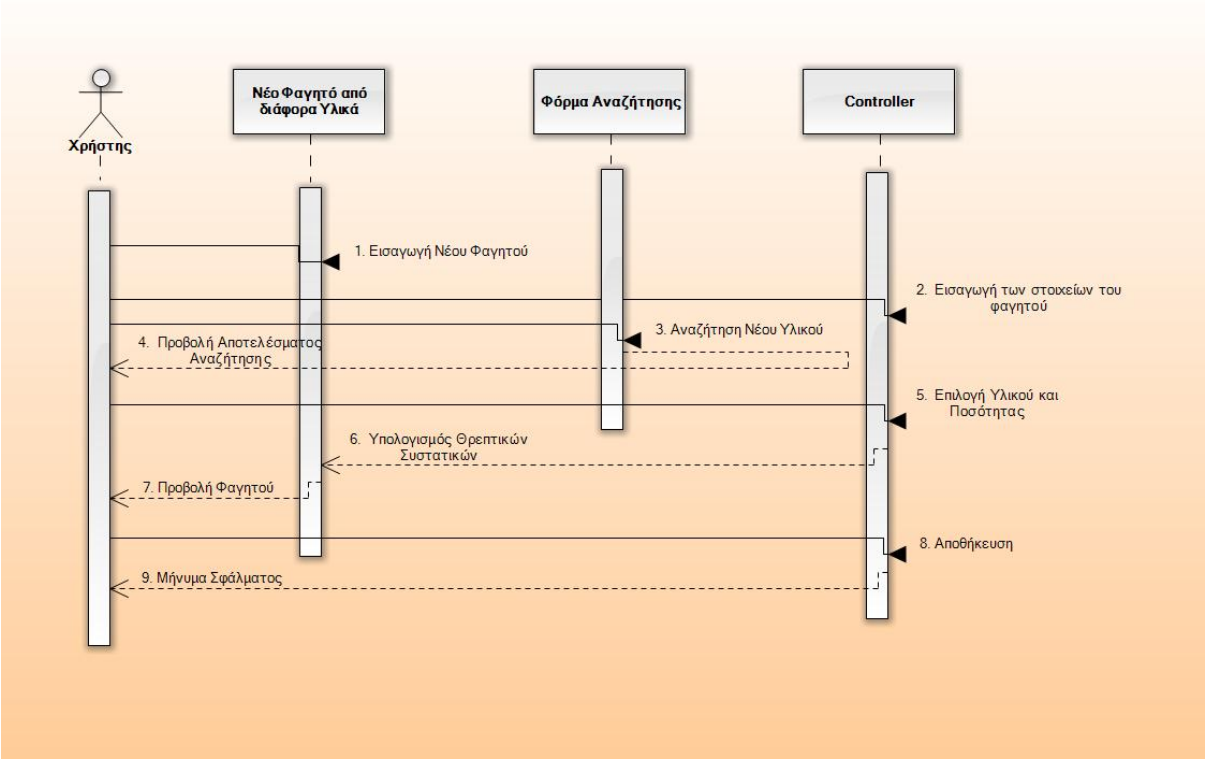
1. Επιλογή του χρήστη για Εισαγωγή Νέου Φαγητού από Διάφορα Υλικά.
2. Εισαγωγή των στοιχείων του φαγητού (Όνομα, Περιγραφή, Αριθμό Μερίδων)

3. Αναζήτηση Υλικών για Φαγητό
4. Επιλογή Υλικού και Ποσότητας του Υλικού που θέλουν να προσθέσουν στο φαγητό
5. Πρόσθεση Επιλεγμένου Υλικού στην λίστα με τα Υλικά του Φαγητού
6. Υπολογισμός Θρεπτικής Αξίας Φαγητού ανά μερίδα, 100 gr και ολόκληρου του φαγητού.
7. Επανάληψη των βημάτων 3-6 όσες φορές επιθυμεί ο χρήστης
8. Επιλογή για αποθήκευση φαγητού στην βάση
9. Έλεγχος εαν τα στοιχεία δεν υπάρχουν ήδη στην βάση.
10. Αποθήκευση

Διάγραμμα Συνεργασίας



Ακολουθιακό Διάγραμμα

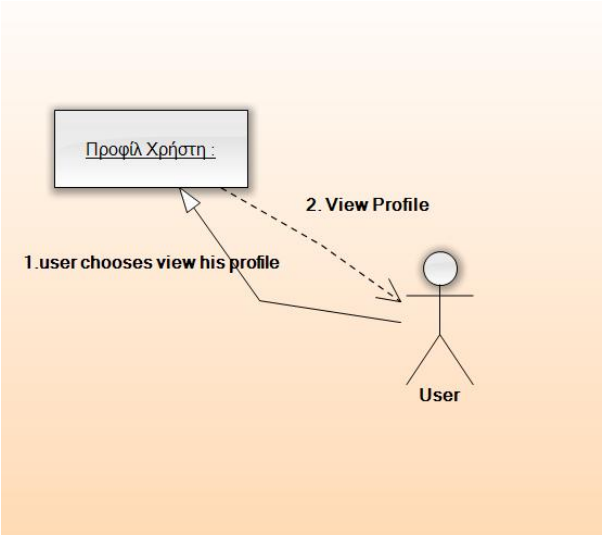


5.2.1.7 Λειτουργία: Προβολή Προφίλ Χρήστη

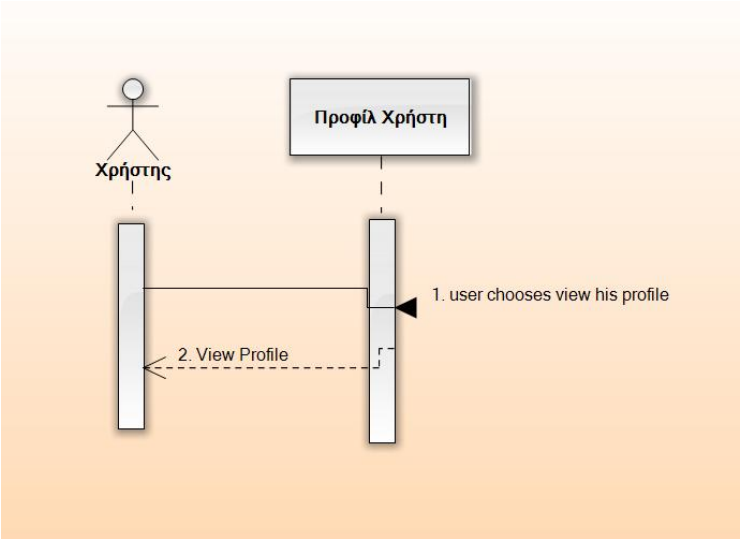
Κατά την λειτουργία της Προβολής του Προφίλ του Χρήστη γίνονται τα εξής βήματα:

1. Επιλογή του χρήστη για Προβολής του Προφίλ του.
2. Προβολή Προφίλ

Διάγραμμα Συνεργασίας



Ακολουθιακό Διάγραμμα

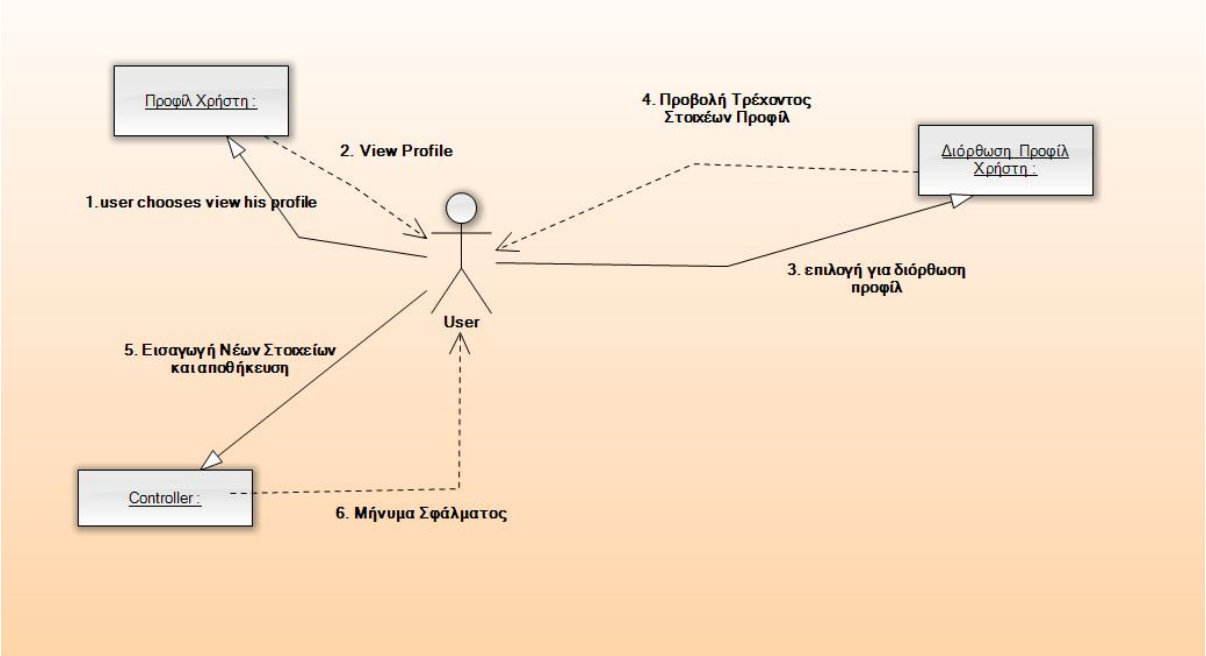


5.2.1.8 Λειτουργία: Διόρθωση Προφίλ Χρήστη

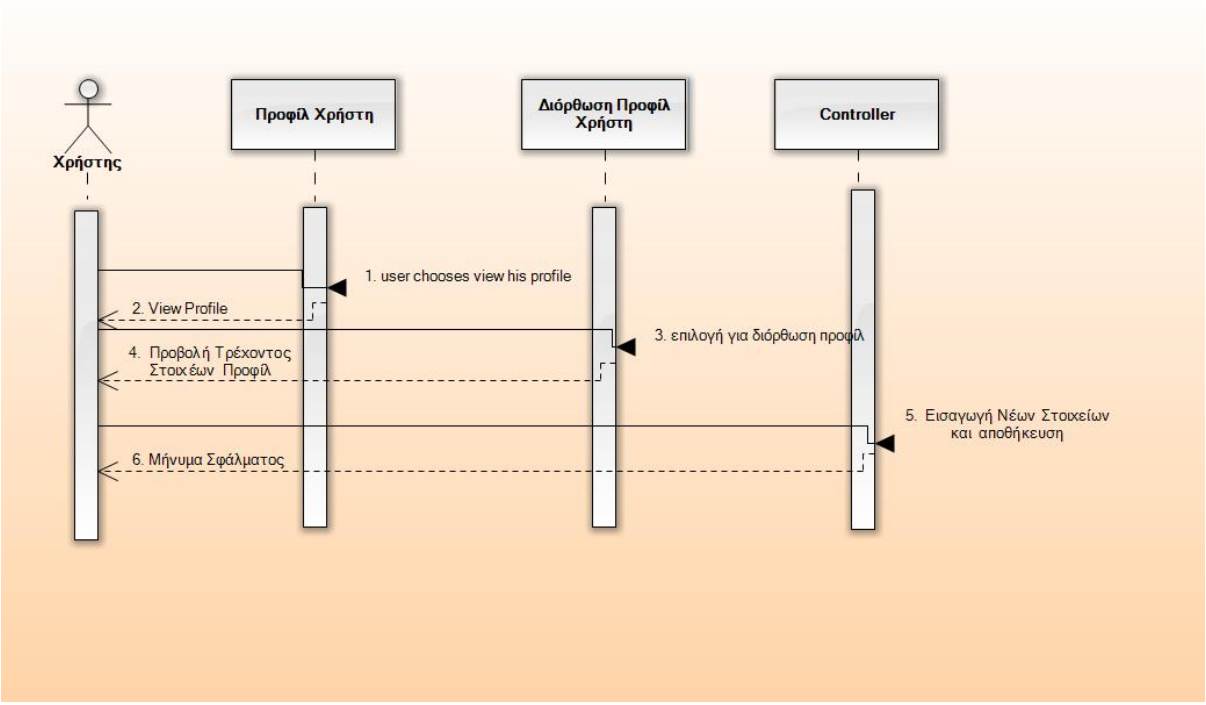
Κατά την λειτουργία της Διόρθωσης του Προφίλ του Χρήστη γίνονται τα εξής βήματα:

1. Επιλογή του χρήστη για Προβολή του Προφίλ του.
2. Προβολή Προφίλ
3. Επιλογή για Διόρθωση Προφίλ
4. Εισαγωγή Νέων Στοιχείων στην θέση των παλιών
5. Επιλογή για αποθήκευση αλλαγών
6. Έλεγχος εαν τα νέα στοιχεία έχουν συμπληρωθεί σωστά
7. Αποθήκευση ανανεωμένου προφίλ Χρήστη

Διάγραμμα Συνεργασίας



Ακολουθιακό Διάγραμμα



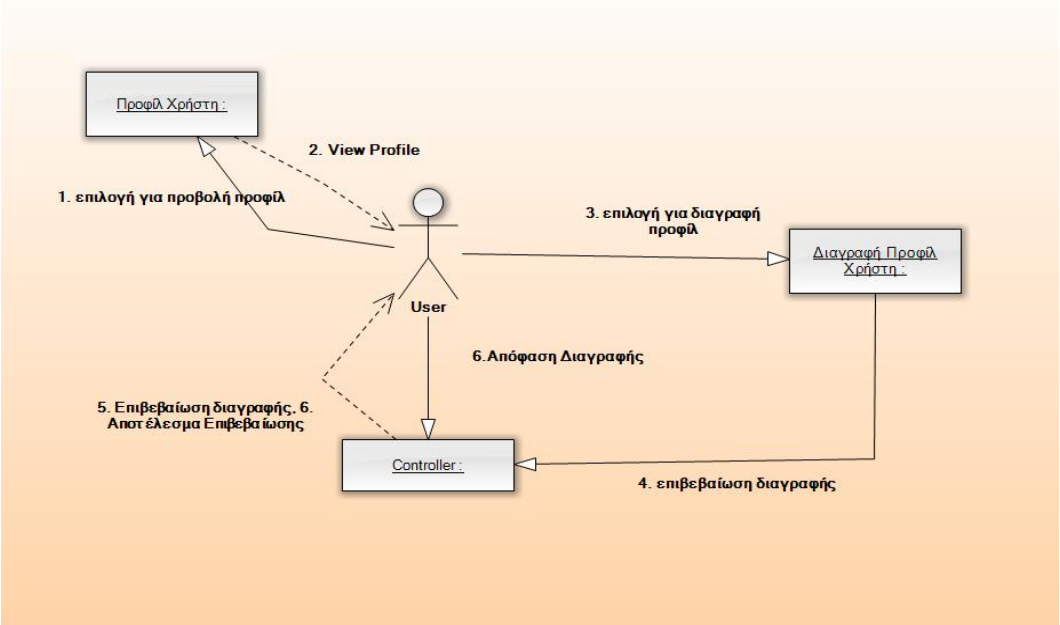
5.2.1.9 Λειτουργία: Διαγραφή Προφίλ Χρήστη

Κατά την λειτουργία της Διαγραφής του Προφίλ του Χρήστη γίνονται τα εξής βήματα:

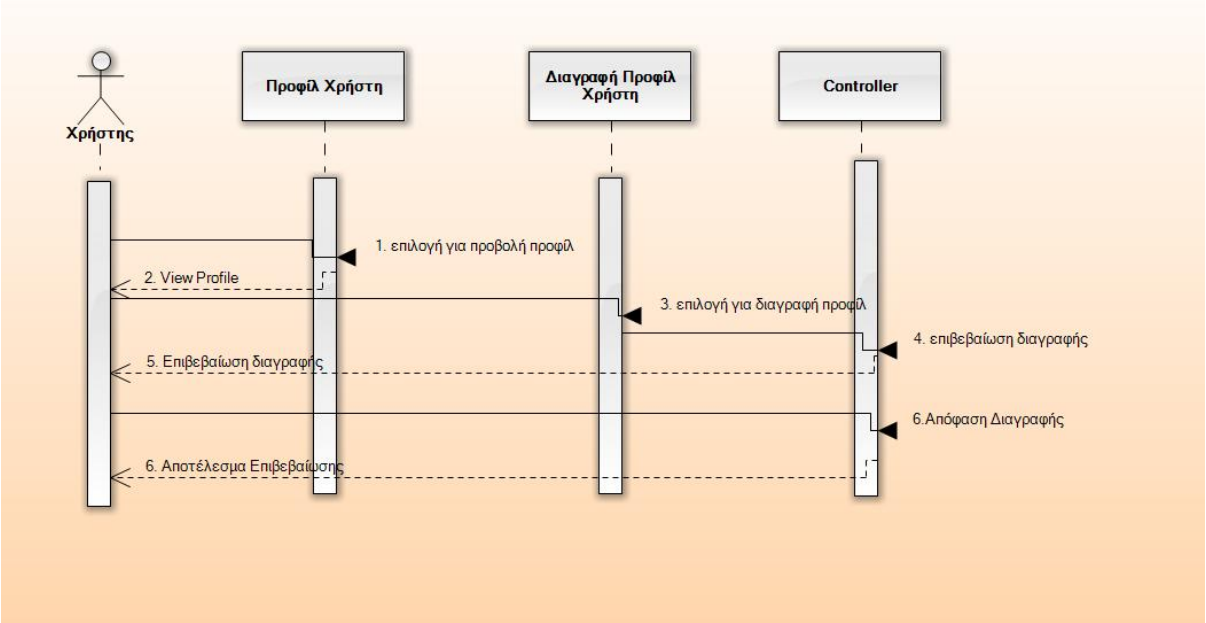
1. Επιλογή του χρήστη για Προβολή του Προφίλ του.

2. Προβολή Προφίλ
3. Επιλογή για Διαγραφή Προφίλ
4. Επιβεβαίωση διαγραφής Προφίλ
5. Διαγραφή Προφίλ

Διάγραμμα Συνεργασίας



Ακολουθιακό Διάγραμμα



5.2.1.10 Λειτουργία: Προβολή Διατροφών Χρήστη

Υπάρχουν 2 τρόποι να γίνει αυτή η λειτουργία. Το ακουλουθιακό διάγραμμα και το διάγραμμα συνεργασίας που ακολουθεί περιγράφει τον πρώτο από τους δύο τρόπους. Ο δεύτερος τρόπος γίνεται με όμοιο τρόπο.

Κατά την λειτουργία της Προβολής των Διατροφών του Χρήστη γίνονται τα εξής βήματα:

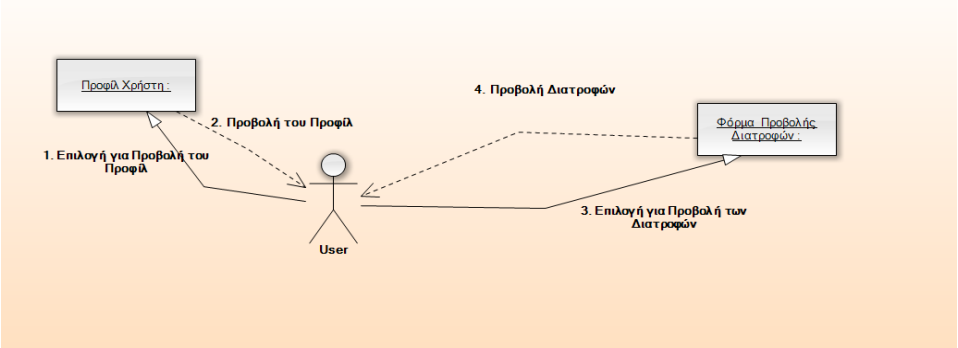
Α Τρόπος:

1. Επιλογή του χρήστη για Προβολή του Προφίλ του.
2. Προβολή Προφίλ
3. Επιλογή για Προβολή των Διατροφών του
4. Εμφάνιση Διατροφών Χρήστη

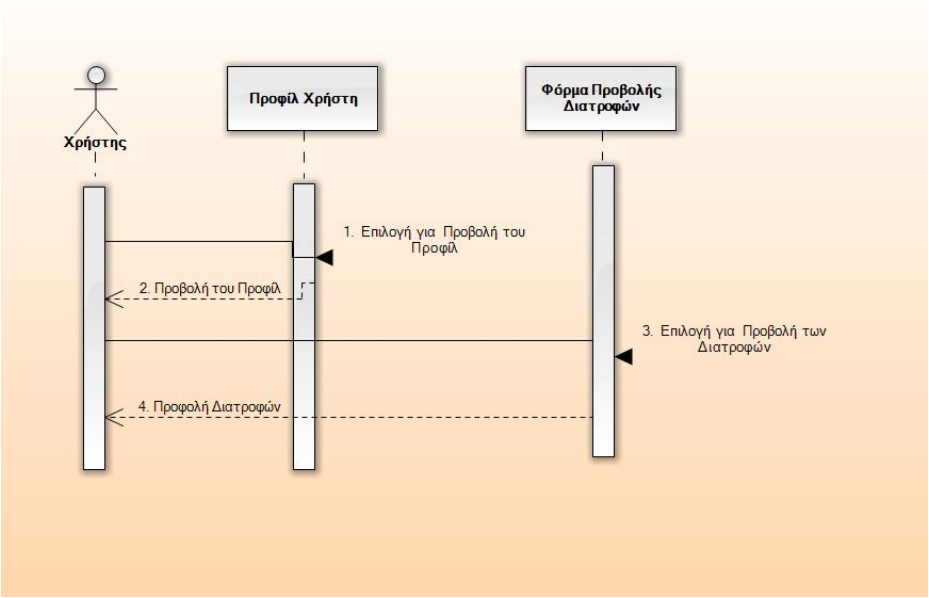
Β Τρόπος:

1. Επιλογή του χρήστη για Προβολή των Διατροφών του
2. Εμφάνιση Διατροφών Χρήστη

Διάγραμμα Συνεργασίας



Ακολουθιακό Διάγραμμα



5.2.1.11 Λειτουργία: Διόρθωση Διατροφής Χρήστη

Υπάρχουν 2 τρόποι να γίνει αυτή η λειτουργία. Το ακουλουθιακό διάγραμμα και το διάγραμμα συνεργασίας που ακολουθεί περιγράφει τον πρώτο από τους δύο τρόπους. Ο δεύτερος τρόπος γίνεται με όμοιο τρόπο.

Κατά την λειτουργία της Διόρθωσης μιας Διατροφής του Χρήστη γίνονται τα εξής βήματα:

Α Τρόπος:

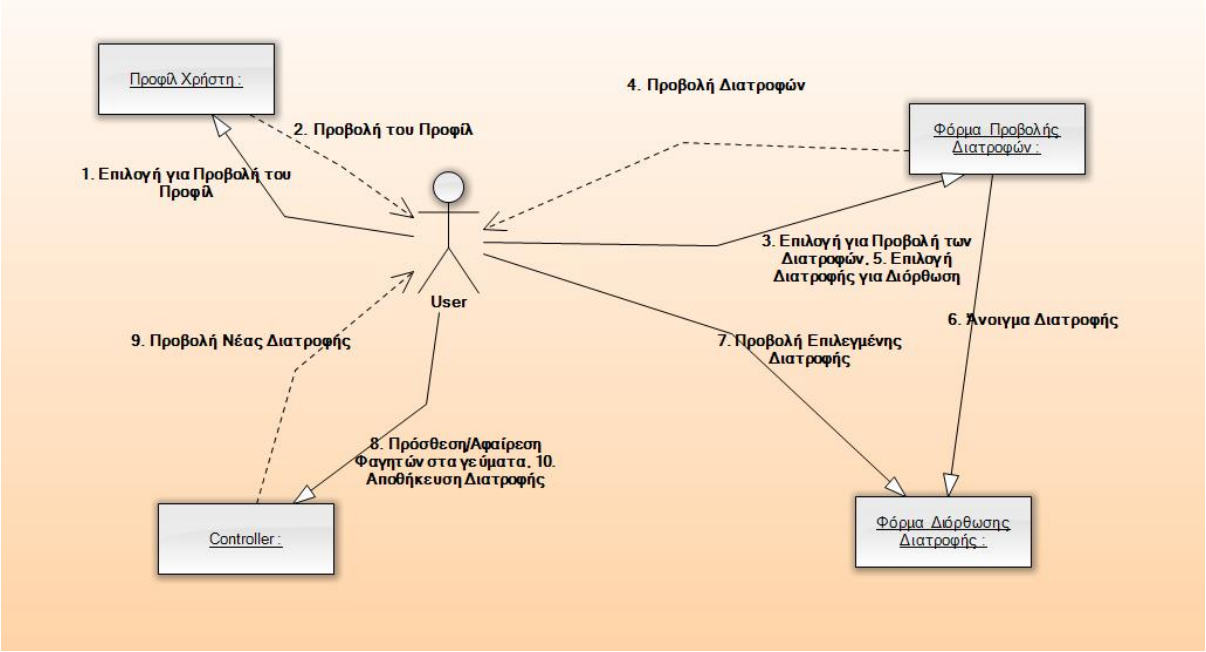
1. Επιλογή του χρήστη για Προβολή του Προφίλ του.
2. Προβολή Προφίλ
3. Επιλογή για Προβολή των Διατροφών του Χρήστη
4. Εμφάνιση Διατροφών Χρήστη
5. Επιλογή και άνοιγμα Διατροφής για Διόρθωση
6. Πρόσθεση/Αφαίρεση Φαγητών στα γεύματα
7. Αποθήκευση Διατροφής

Β Τρόπος:

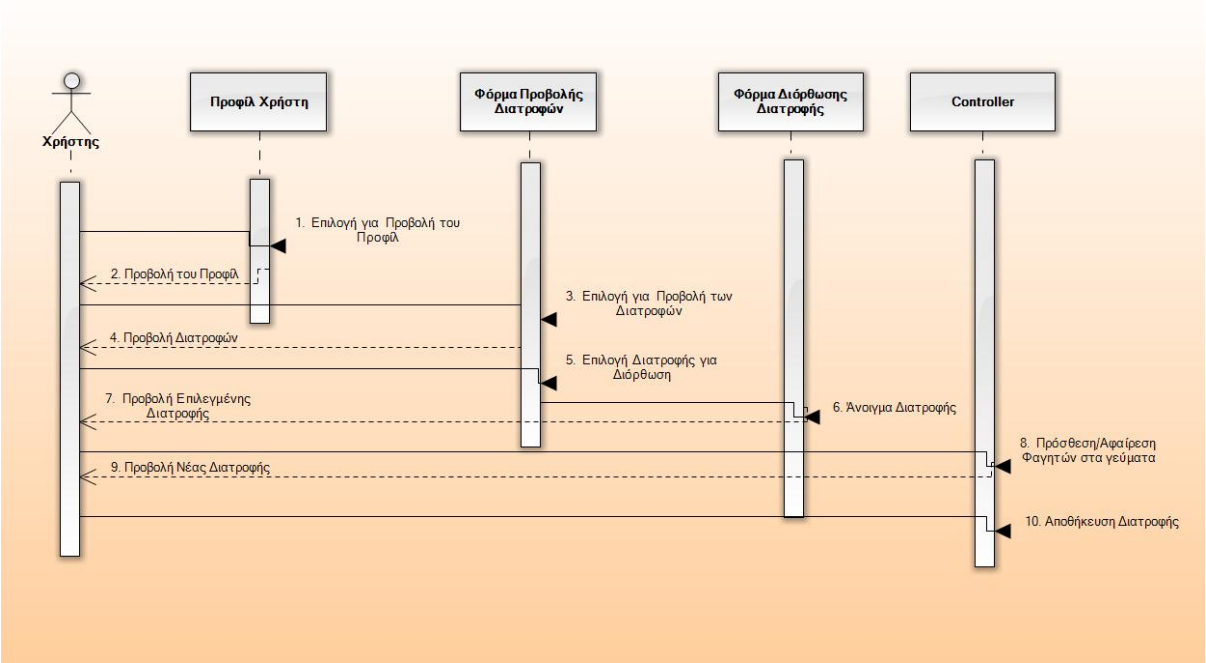
1. Επιλογή για Προβολή των Διατροφών του Χρήστη
2. Εμφάνιση Διατροφών Χρήστη
3. Επιλογή και άνοιγμα Διατροφής για Διόρθωση

4. Πρόσθεση/Αφαίρεση Φαγητών στα γεύματα
5. Αποθήκευση Διατροφής

Διάγραμμα Συνεργασίας



Ακολουθιακό Διάγραμμα



5.2.1.12 Λειτουργία: Διαγραφή Διατροφής Χρήστη

Υπάρχουν 2 τρόποι να γίνει αυτή η λειτουργία. Το ακουλουθιακό διάγραμμα και το διάγραμμα συνεργασίας που ακολουθεί περιγράφει τον πρώτο από τους δύο τρόπους. Ο δεύτερος τρόπος γίνεται με όμοιο τρόπο.

Κατά την λειτουργία της Διαγραφής μιας Διατροφής του Χρήστη γίνονται τα εξής βήματα:

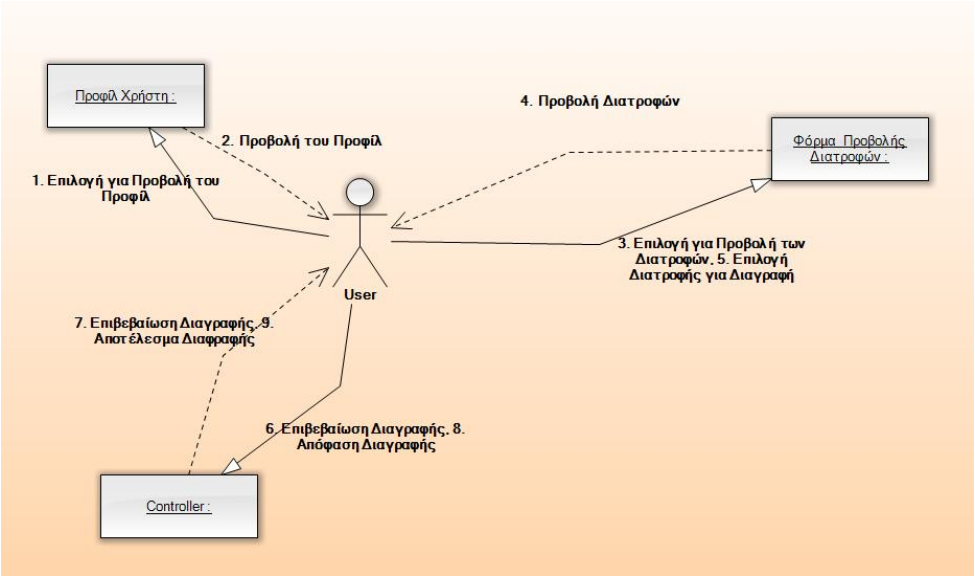
Α Τρόπος:

1. Επιλογή του χρήστη για Προβολή του Προφίλ του.
2. Προβολή Προφίλ
3. Επιλογή για Προβολή των Διατροφών του Χρήστη
4. Εμφάνιση Διατροφών Χρήστη
5. Επιλογή Διατροφής για διαγραφή
6. Επιβεβαίωση Διαγραφής
7. Διαγραφή Διατροφής

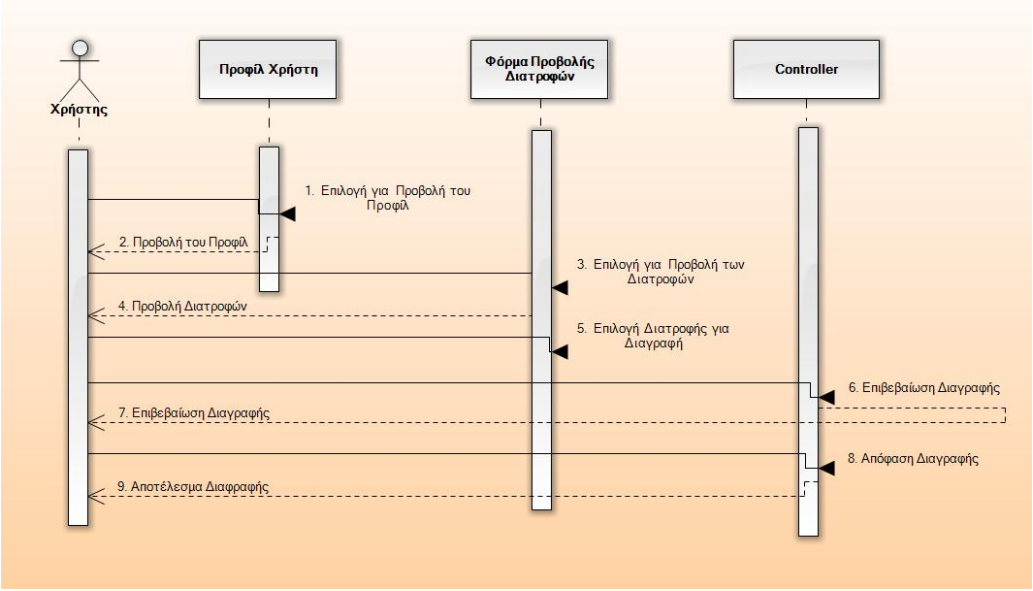
Β Τρόπος:

1. Επιλογή για Προβολή των Διατροφών του Χρήστη
2. Εμφάνιση Διατροφών Χρήστη
3. Επιλογή Διατροφής για διαγραφή
4. Επιβεβαίωση Διαγραφής
5. Διαγραφή Διατροφής

Διάγραμμα Συνεργασίας



Ακολουθιακό Διάγραμμα



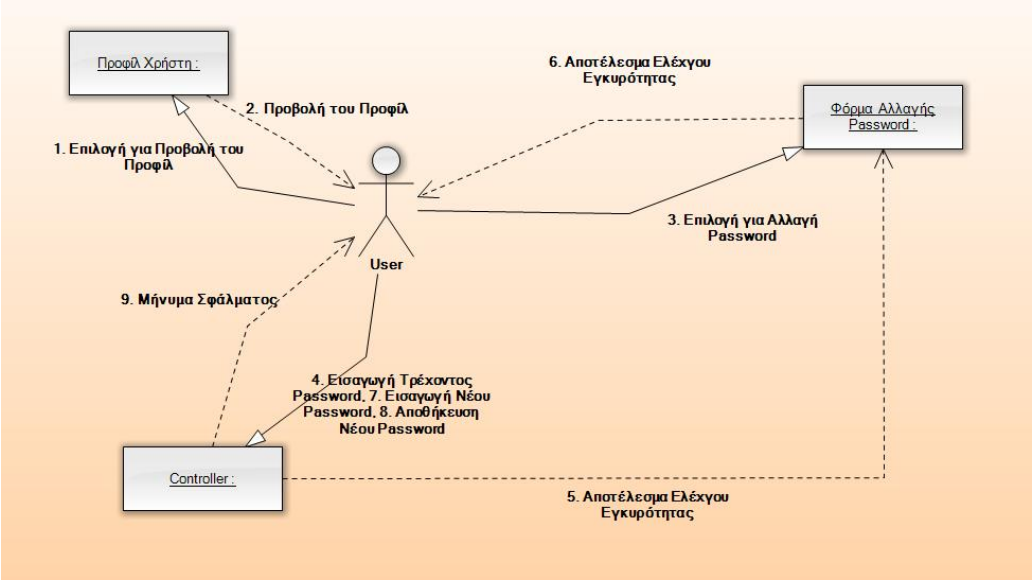
5.2.1.13 Λειτουργία: Αλλαγή Password Χρήστη

Κατά την λειτουργία της Διαγραφής μιας Διατροφής του Χρήστη γίνονται τα εξής βήματα:

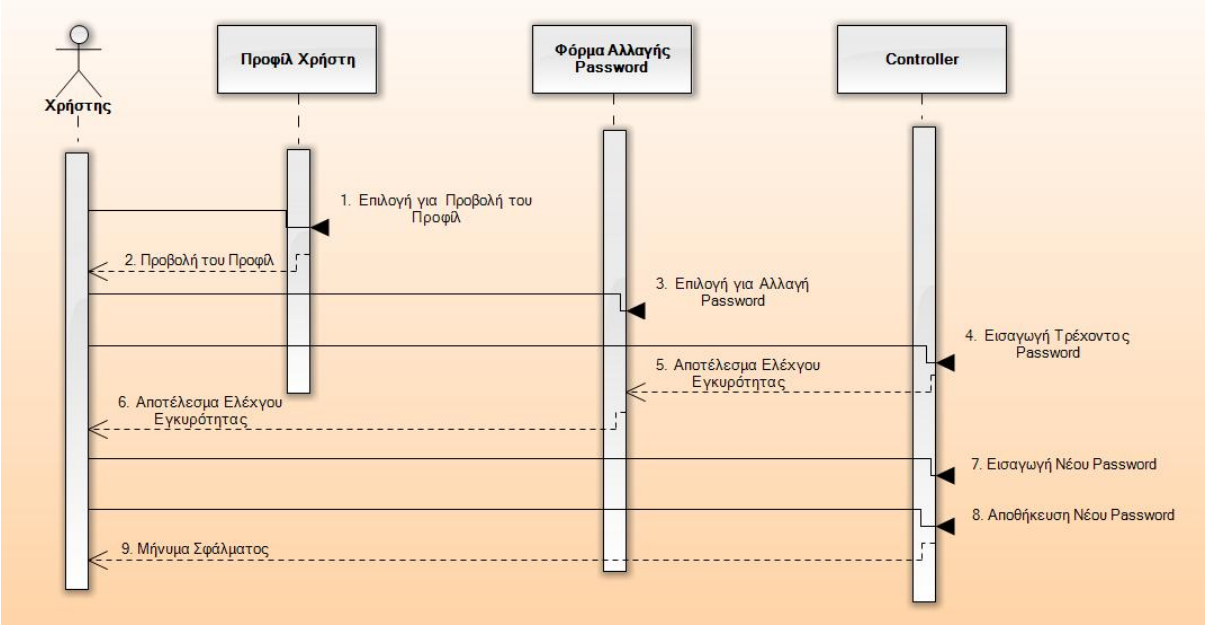
1. Επιλογή του χρήστη για Προβολή του Προφίλ του.
2. Προβολή Προφίλ
3. Επιλογή για Αλλαγή Password του Χρήστη
4. Εισαγωγή Τρέχοντος Password
5. Έλεγχος εάν είναι όντως έγκυρο

6. Εισαγωγή Νέου Password και επιβεβαίωση του
7. Αποθήκευση Νέου Password.

Διάγραμμα Συνεργασίας



Ακολουθιακό Διάγραμμα

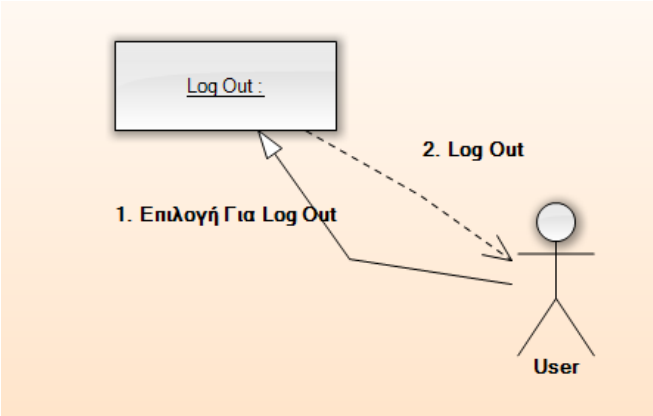


5.2.1.14 Λειτουργία: Log Out

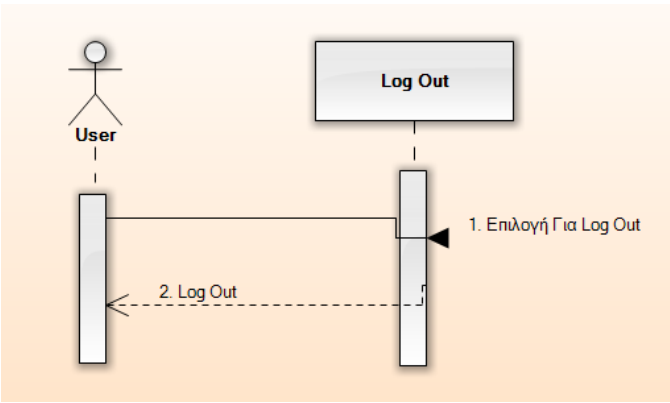
Κατά την λειτουργία Log Out γίνονται τα εξής βήματα:

1. Επιλογή του χρήστη για Log Out.
2. Log Out

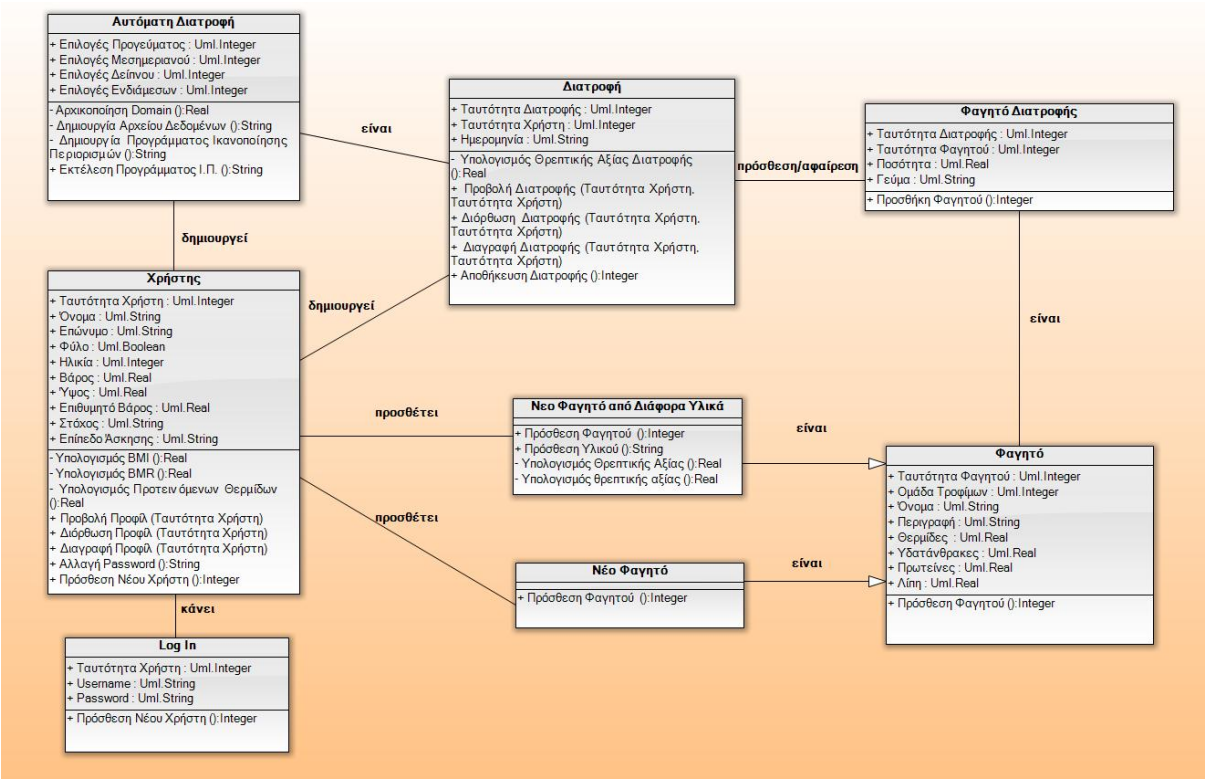
Διάγραμμα Συνεργασίας



Ακολουθιακό Διάγραμμα



5.2.2 Class Diagram



Κεφάλαιο 6

Υλοποίηση Συστήματος

6.1 Εισαγωγή
6.2 Περιγραφή Εργαλείων Ανάπτυξης
6.3 Περιγραφή Γλωσσών Προγραμματισμού
6.4 Περιγραφή Βάσης Δεδομένων
6.5 Προγραμματισμός Ικανοποίησης Περιορισμών
6.6 Δυσκολίες Υλοποίησης και Εύρεση Λύσης

6.1 Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται η περιγραφή της φάσης της υλοποίησης όπως αυτή προέκυψε από τις φάσεις των απαιτήσεων, προδιαγραφών και της σχεδίασης. Θα περιγραφούν τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του κάθε κομματιού του συστήματος, των γλωσσών προγραμματισμού, της βάσης δεδομένων και την ανάπτυξη του προγράμματος ικανοποίησης περιορισμών. Επίσης θα επισημανθούν ποια σημεία ήταν δύσκολα και πως επιλύθηκαν.

6.2 Περιγραφή Εργαλείων Ανάπτυξης

Η επιλογή των εργαλείων ανάπτυξης του συστήματος παίζει καθοριστικό ρόλο στον κύκλο ζωής ενός συστήματος. Όσο πιο εύχρηστο το εργαλείο τόσο πιο εύκολή θα είναι η υλοποίηση του συστήματος καθώς επίσης και η συντήρηση του σε μετέπειτα σταδιο.
Για την ανάπτυξη του συστήματος χρησιμοποιήθηκε συνδυασμός εργαλείων τα οποία περιγράφονται αναλυτικά πιο κάτω.

6.2.1 NetBeans IDE

Το εργαλείο ανάπτυξης NetBeans είναι ένα open source πρόγραμμα που υποστηρίζει πολλές γλώσσες προγραμματισμού για την ανάπτυξη γραφικού περιβάλλοντος (GUI). Πρόκειται για ένα εργαλείο αρκετά εξελιγμένο και εύκολο στην χρήση που το κάνει κατάλληλο για ανάπτυξη οποιουδήποτε είδους εφαρμογής (επιτραπέζιων, επιχειρηματικών, web και mobile applications).
Μερικά από τα χαρακτηριστικά του είναι η αυτόματη αναγνώριση των λαθών, η αυτόματη συμπλήρωση του κώδικα και η εύκολη ανάπτυξη interface μέσω του drag and drop γραφικού του εργαλείου. Επίσης η εργαλειοθήκη του είναι πολύ καλά εξοπλισμένη με διάφορα components όπως buttons, radiobuttons, lists, textboxes τα οποία ο χρήστης μπορεί να εισάγει στις φόρμες και στην συνέχεια να τους δώσει κάποια λειτουργικότητα.

6.2.2 MySQL Workbench

Το MySQL Workbench είναι το εργαλείο για μοντελοποίηση, ανάπτυξη και διαχείρηση βάσεων δεδομένων στην γλώσσα MySQL. Προσφέρει πολλαπλές συνδέσεις MySQL, μοντελοποίηση των δεδομένων με την δημιουργία σχήματος βάσης, δημιουργία πινάκων και εκτέλεση των MySQL queries. Είναι ένα εργαλείο που παρέχει προστασία στα δεδομένα και αποδοτική χρήση αφού είναι αρκετά γρήγορο.

6.2.3 Software Ideas Modeler

Πρόκειται για το εργαλείο δημιουργίας των σχεδιαγραμμάτων κατά τον κύκλο ανάπτυξης του λογισμικού. Υποστηρίζει την δημιουργία διαγραμμάτων UML 2.2, BPMN και άλλων. Είναι δωρεάν λογισμικό που προσφέρει εύκολη και γρήγορη σχεδίαση και μπορεί να χρησιμοποιηθεί για την σχεδίαση 16 διαφορετικών είδων διαγραμμάτων.

6.3 Περιγραφή Γλωσσών Προγραμματισμού

Όπως και η επιλογή των εργαλείων ανάπτυξης, η επιλογή των γλωσσών προγραμματισμού είναι εξίσου σημαντική στην υλοποίηση ενός συστήματος. Πιο κάτω περιγράφονται οι γλώσσες προγραμματισμου που επέλεξα για την υλοποίηση της εφαρμογής.

6.3.1 Java

Πρόκειται για μια απλή, αντικειμενοστραφή γλώσσα προγραμματισμού που βασίζεται στην δημιουργία κλάσεων και είναι σχεδιασμένη για να έχει όσο το δυνατόν λιγότερες εκτελεστικές εξαρτήσεις γίνεται. Υποστηρίζει την ανάπτυξη γραφικών διεπιφανιών και έχει πολλές βιβλιοθήκες που επεκτείνουν ακόμη περισσότερο τις δυνατότητες της. Είναι η πιο διαδεδομένη γλώσσα προγραμματισμού.

6.3.2 MiniZinc

Η MiniZinc είναι μια μεσαίου επιπέδου γλώσσα που σχεδιάστηκε για την επίλυση προβλήμάτων ικανοποίησης περιορισμών. Είναι αρκετά ψηλού επιπέδου για μοντελοποίηση προβλημάτων εύκολα και αρκετά χαμηλού επιπέδου για να μπορεί να χρησιμοποιηθεί σε υπάρχοντες επιλυτές (solvers). Χρησιμοποιείται για προβλήματα βελτιστοποίησης και απόφασης για ακέραιους και πραγματικούς αριθμούς. Είναι ένα υποσύνολο της γλώσσας Zinc και χρησιμοποιεί την FlatZinc για την επίλυση των προβλημάτων. Αυτό γίνεται με την μετατροπή του μοντέλου της MiniZinc σε μοντέλο FlatZinc. Την επέλεξα γιατί είναι απλή στην εγκατάσταση και την χρήση.
Ένα πρόγραμμα σε MiniZinc αποτελείται από ένα ή δύο αρχεία. Υπάρχει δηλαδή το αρχείο με την επέκταση ".mzn" το οποίο περιέχει το πρόγραμμα και μπορεί να υπάρχει και ένα αρχείο δεδομένων με την επέκταση ".dzn" το οποίο να περιέχει τα δεδομένα του προγράμματος.

6.4 Περιγραφή Βάσης Δεδομένων

Το σύστημα χειρίζεται δύο ειδών δεδομένα. Τα δεδομένα που αφορούν τον χρήστη και τα δεδομένα που αφορούν τα φαγητά. Τα δεδομένα για τον χρήστη είναι τα προσωπικά του στοιχεία και οι διατροφές που δημιουργεί και εισάγονται από τον ίδιο τον χρήστη κατά την διάρκεια της εκτέλεσης των λειτουργιών του συστήματος. Τα δεδομένα σχετικά με τα φαγητά όμως πηγάζουν από την βάση δεδομένων του Υπουργείου Γεωργίας και Υγείας των Η.Π.Α. (USDA) και η οποία είναι διαθέσιμη για ελεύθερη χρήση. Στα δεδομένα για τα φαγητά έρχεται να προσθέσει ο χρήστης κατά την λειτουργία "Προσθεση Νέου Φαγητού" και "Πρόσθεση Νέου Φαγητού από Διάφορα Υλικά". Έτσι γίνεται μια επέκταση της βάσης δεδομένων USDA για την εξυπηρέτηση των σκοπών του συστήματος αυτού. Πιο κάτω δίνονται οι πίνακες που αφορούν τις δύο κατηγορίες δεδομένων.

6.4.1 Πίνακες Βάσης USDA

Στην βάση USDA υπάρχουν οι πίνακες:

- Food Description
- Nutrient Data
- Weight
- Footnote
- Food Group Description
- LanguaL Factor
- LanguaL Factors Description
- Nutrient Definition
- Source Code
- Data Derivation Description
- Sources of Data
- Sources of Data Link

Οι περισσότεροι πίνακες δεν χρησιμοποιήθηκαν για αυτό και δεν θα περιγραφούν στα πλαίσια της διπλωματικής αυτής εργασίας. Επίσης κάποια από τα πεδία των πινάκων δεν

χρειάστηκαν. Πιο κάτω ακολουθούν οι πίνακες που χρησιμοποιήθηκαν και οι περιγραφές των δεδομένων που κρατούν. Επίσης αναγράφονται ποια πεδία χρησιμοποιήθηκαν από τον κάθε πίνακα και για ποιες λειτουργίες.

6.4.1.1 Πίνακας Food Description:

Όνομα Στήλης	Τύπος	Περιγραφή
NDB_No*	Συμβολοσειρά	Αριθμός που προσδιορίζει μοναδικά ένα φαγητό
FdGrp_Cd	Συμβολοσειρά	Αριθμός που προσδιορίζει την ομάδα τροφίμων του φαγητού
Long_Desc	Συμβολοσειρά	Περιγραφή του φαγητού
Shrt_Desc	Συμβολοσειρά	Περιγραφή του φαγητού με την χρήση συντομογραφιών
ComName	Συμβολοσειρά	Το κοινό όνομα του φαγητού
ManufacName	Συμβολοσειρά	Το όνομα του κατασκευαστή
Survey	Συμβολοσειρά	Υπάρχει πλήρης αναφορά για τα θρεπτικά συστατικά ή όχι
Ref_desc	Συμβολοσειρά	Περιγραφή των μη βρώσιμων μερών ενός φαγητού (π.χ. των οστών)
Refuse	Δεκαδικός	Ποσοστό απορριμάτων
SciName	Συμβολοσειρά	Επιστημονικό όνομα φαγητού
N_Factor	Δεκαδικός	Συντελεστής για τη μετατροπή του αζώτου σε πρωτεΐνη
Pro_Factor	Δεκαδικός	Συντελεστής για τον υπολογισμό των θερμίδων από πρωτεΐνη
Fat_Factor	Δεκαδικός	Συντελεστής για τον υπολογισμό των θερμίδων από λίπος
CHO_Factor	Δεκαδικός	Συντελεστής για τον υπολογισμό των θερμίδων από υδατάνθρακες

*Primary Key για τον Πίνακα Food Description.

Ο πίνακας αυτός χρησιμοποιείται κατά τις λειτουργίες της αυτόματης δημιουργίας διατροφών, αναζήτησης φαγητών για πρόσθεση στο ημερήσιο διαιτολόγιο κατά την λειτουργία της "χειροκίνητης" δημιουργίας διατροφής, και πρόσθεσης νέου φαγητού στην βάση.

Τα πεδία που χρησιμοποιούνται είναι το NDB_No, FdGrp_Cd, Long_Desc, Shrt_Desc και το ComName.

6.4.1.2 Πίνακας Food Group Description:

Όνομα Στήλης	Τύπος	Περιγραφή
FdGrp_Cd*	Συμβολοσειρά	Κωδικός που προσδιορίζει μια ομάδα τροφίμων.
FdGrp_Desc	Συμβολοσειρά	Όνομα ομάδας τροφίμων

*Primary Key για τον Πίνακα Food Group Description.

Ο πίνακας αυτός χρησιμοποιείται κατά την λειτουργία της πρόσθεσης νέου φαγητού στην βάση τόσο ενός συγκεκριμένου προϊόντος όσο και ενός φαγητού από διάφορα υλικά. Χρησιμοποιούνται όλα τα πεδία.

6.4.1.3 Πίνακας Nutrient Data:

Όνομα Στήλης	Τύπος	Περιγραφή
NDB_No*	Συμβολοσειρά	Αριθμός που προσδιορίζει μοναδικά ένα φαγητό
Nutr_No*	Συμβολοσειρά	Μοναδικός αριθμός για θρεπτικό συστατικό
Nutr_Val	Δεκαδικός	Θρεπτική Αξία στα 100 γρ του βρώσιμου μέρους
Num_Data_Pts	Δεκαδικός	Ο αριθμός των αναλύσεων για υπολογισμό των θρεπτικών συστατικών
Std_Error	Δεκαδικός	Τυπικό σφάλμα του μέσου υπολογισμού
Src_Cd	Συμβολοσειρά	Κωδικός για τύπο δεδομένων
Deriv_Cd	Συμβολοσειρά	Κωδικός που προσδιορίζει το πως υπολογίστηκε η τιμή
Ref_NDB_No	Συμβολοσειρά	Αριθμός του στοιχείου που χρησιμοπείται για ελληπείς τιμές
Add_Nutr_Mark	Συμβολοσειρά	Υποδεικνύει μια βιταμίνη ή ορυκτό που προσθέτηκε για εμπλουτισμό
Num_Studies	Ακέραιος	Αριθμός Μελέτων
Min	Δεκαδικός	Ελάχιστη Τιμή
Max	Δεκαδικός	Μέγιστη τιμή
DF	Ακέραιος	Βαθμός ελευθερίας
Low_EB	Δεκαδικός	Κατώτατο 95% όριο λάθους
Up_EB	Δεκαδικός	Ανώτατο 95% όριο λάθους
Stat_cmt	Συμβολοσειρά	Στατιστικά Σχόλια
AddMod_Date	Συμβολοσειρά	Ημερομηνία Προσθήσης ή Ανανέωσης Τιμής
CC	Συμβολοσειρά	Κωδικός που υποδηλώνει εμπιστοσύνη της ποιότητας των δεδομένων

*Primary Key για τον Πίνακα Nutrient Data.

Ο πίνακας αυτός χρησιμοποιείται κατά τις λειτουργίες της αυτόματης δημιουργίας διατροφών, αναζήτησης φαγητών για πρόσθεση στο ημερήσιο διαιτολόγιο κατά την λειτουργία της "χειροκίνητης" δημιουργίας διατροφής, και πρόσθεσης νέου φαγητού στην βάση.

Τα πεδία που χρησιμοποιούνται είναι το NDB_No, Nutr_No και το Nutr_Val.

6.4.1.4 Πίνακας Nutrient Definition:

Όνομα Στήλης	Τύπος	Περιγραφή
Nutr_No*	Συμβολοσειρά	Αριθμός που προσδιορίζει μοναδικά ένα θρεπτικό συστατικό
Units	Συμβολοσειρά	Μονάδα Μέρτησης (mg, g, μg, κλπ).
Tagname	Συμβολοσειρά	Μοναδική συντομογραφία για ένα θρεπτικό συστατικό
NutrDesc	Συμβολοσειρά	Όνομα θρεπτικού συστατικού
Num_Dec	Συμβολοσειρά	Αριθμός Δεκαδικών Ψηφίων στρογγυλοποίησης
SR_Order	Ακέραιος	Χρησιμοποιήται για ταξινόμηση

*Primary Key για τον Πίνακα Nutrient Definition.

Ο πίνακας αυτός δεν χρησιμοποιείται κατά τις λειτουργίες του συστήματος. Παράχει βοήθεια όμως στην κατανόηση των εγγραφών του πινακα **Nutrient Data.**

6.4.1.5 Πίνακας Weight:

Όνομα Στήλης	Τύπος	Περιγραφή
NDB_No*	Συμβολοσειρά	Αριθμός που προσδιορίζει μοναδικά ένα φαγητό
Seq*	Συμβολοσειρά	Αριθμός σειράς Μετρησης
Amount	Δεκαδικός	Μονάδα Τροποποίησης (για παράδειγμα 1 στο "1 φλιτζάνι»).
Msre_Desc	Συμβολοσειρά	Περιγραφή (για παράδειγμα φλυντζάνι, σε κύβους κλπ).
Gm_Wgt	Δεκαδικός	Βάρος σε γραμμάρια
Num_Data_Pts	Ακέραιος	Αριθμός Δεκαδικών
Std_Dev	Δεκαδικός	Τυπική απόκλιση

*Primary Key για τον Πίνακα Weight.

Ο πίνακας αυτός επίσης, χρησιμοποιείται κατά τις λειτουργίες της αυτόματης δημιουργίας διατροφών, αναζήτησης φαγητών για πρόσθεση στο ημερήσιο διαιτολόγιο κατά την λειτουργία της "χειροκίνητης" δημιουργίας διατροφής, και πρόσθεσης νέου φαγητού στην βάση.

Τα πεδία που χρησιμοποιούνται είναι το NDB_No, Amount, Msre_Desc και το Gm_Wgt.

6.4.2 Επιπρόσθετοι Πίνακες Βάσης

Οι ακόλουθοι πίνακες είναι αυτοί που δημιουργήθηκαν συγκεκριμένα για τις ανάγκες του συστήματος.

6.4.2.1 Πίνακας Candidate Categories:

Όνομα Στήλης	Τύπος	Περιγραφή
Id_candidate*	Ακέραιος	Αριθμός που προσδιορίζει μοναδικά έναν υποψήφιο χρήστη
min_age	Ακέραιος	Κατώτατο όριο ηλικίας για κατηγορία
max_age	Ακέραιος	Ανώτατο όριο ηλικίας για κατηγορία
sex	Συμβολοσειρά	Το φύλο του υποψήφιου χρήστη
age_unit	Συμβολοσειρά	Μονάδα μέτρησης ηλικίας (για παράδειγμα μήνες, χρόνια)

*Primary Key για τον Πίνακα Candidate Categories.

Ο πίνακας αυτός κρατά τις κατηγορίες χρηστών και τους κωδικούς τους ανάλογα με την ηλικία και το φύλο τους. Χρησιμοποιείται για την ευρεση των απαραίτητων θρεπτικών συστατικών για τον χρήστη.

6.4.2.2 Πίνακας Nutrient Requirements:

Όνομα Στήλης	Τύπος	Περιγραφή
id_nutr_req*	Ακέραιος	Αριθμός που προσδιορίζει μοναδικά την εγγρφή για απαιτήσεις
candidate_id*	Ακέραιος	Αριθμός που προσδιορίζει μοναδικά έναν υποψήφιο χρήστη
nutr_code	Συμβολοσειρά	Αριθμός που προσδιορίζει μοναδικά ένα θρεπτικό συστατικό
nutr_value	Δεκαδικός	Η τιμή για το θρεπτικό συστατικό

*Primary Key για τον Πίνακα Nutrient Requirements.

Στον πίνακα αυτό υπάρχουν τα δεδομένα για τις ανάγκες των κατηγορίων χρηστών. Τα δεδομένα αυτά πηγάζουν από διάφορες έρευνες που έχουν γίνει από το Υπουργείο Υγείας του Ηνωμένου Βασιλείου.

6.4.2.3 Πίνακας Log In:

Όνομα Στήλης	Τύπος	Περιγραφή
id_user*	Ακέραιος	Αριθμός που προσδιορίζει μοναδικά τον χρήστη
Username*	Συμβολοσειρά	Μοναδικό όνομα για είσοδο του χρήστη στο σύστημα
Password*	Συμβολοσειρά	Συνθηματικό για είσοδο του χρήστη στο σύστημα

*Primary Key για τον Πίνακα Log In.

Ο πίνακας Log In κρατά τις πληροφορίες για πρόσβαση του χρήστη στο σύστημα.

6.4.2.4 Πίνακας User Details:

Όνομα Στήλης	Τύπος	Περιγραφή
id_user*	Ακέραιος	Αριθμός που προσδιορίζει μοναδικά τον χρήστη
Name	Συμβολοσειρά	Το όνομα του χρήστη
Surname	Συμβολοσειρά	Το επίθετο του χρήστη
Age	Ακέραιος	Η ηλικία του χρήστη
Gender	Συμβολοσειρά	Το φύλο του χρήστη
Weight	Δεκαδικός	Το τρέχον βάρος του χρήστη
Height	Δεκαδικός	Το ύψος του χρήστη
Target_Weight	Δεκαδικός	Το επιθυμητό βάρος του χρήστη
Goal	Συμβολοσειρά	Ο στόχος του χρήστη (να χάσει , να βάλει ή να διατηρήσει το βάρος)
BMR	Δεκαδικός	Υπολογίσιμο πεδίο για τον Μεταβολικό ρυθμό του χρήστη
BMI	Δεκαδικός	Υπολογίσιμο πεδίο για τον δείκτη μάζας σώματος του χρήστη
Suggested_Calories	Δεκαδικός	Προτεινόμενες Θερμίδες για επίτευξη στόχου
Exercise	Συμβολοσειρά	Το επίπεδο της φυσικής άσκησης του χρήστη

*Primary Key για τον Πίνακα User Details.

Στον πίνακα User Details υπάρχουν όλα τα προσωπικά στοιχεία για τον χρήστη.

6.4.2.5 Πίνακας Nutrition Plans:

Όνομα Στήλης	Τύπος	Περιγραφή
idnutrition_plans*	Ακέραιος	Αριθμός που προσδιορίζει μοναδικά την διατροφή
user_id*	Ακέραιος	Αριθμός που προσδιορίζει μοναδικά τον χρήστη
Nutrition_Date	Συμβολοσειρά	Η ημερομηνία για την οποία προορίζεται η διατροφή
date_created	Ημερομηνία	Η ημερομηνία δημιουργίας της διατροφής

*Primary Key για τον Πίνακα Nutrition Plans.

Στο πίνακα Nutrition Plans υπάρχουν οι πληροφορίες για το ποιές διατροφές δημιούργήθηκαν, από ποιόν χρήστη δημιουργήθηκαν, για ποιά μέρα και πότε.

6.4.2.6 Πίνακας Nutrition Plans Food:

Όνομα Στήλης	Τύπος	Περιγραφή
Food_id*	Ακέραιος	Αριθμός που προσδιορίζει μοναδικά ένα φαγητό
Nutrition_Plans_id*	Ακέραιος	Αριθμός που προσδιορίζει μοναδικά την διατροφή
Food	Συμβολοσειρά	Το όνομα του φαγητού

Amount	Συμβολοσειρά	Η ποσότητα του φαγητού
Carbohydrate	Δεκαδικός	Οι συνολικοί υδατάνθακες στην ποσότητα του φαγητού
Fat	Δεκαδικός	Το συνολικό λίπος στην ποσότητα του φαγητού
Protein	Δεκαδικός	Οι συνολικές πρωτείνες στην ποσότητα του φαγητού
Calories	Δεκαδικός	Οι συνολικές θερμίδες στην ποσότητα του φαγητού
Meal	Συμβολοσειρά	Το γεύμα στο οποίο ανήκει το φαγητο (για παράδειγμα πρόγευμα)

*Primary Key για τον Πίνακα Nutrition Plans Food.

Στον πίνακα Nutrition Plans Food βρίσκονται όλα τα φαγητά που προσθέθηκαν σε κάποια διατροφή. Φυλάγονται στοιχεία όπως η ποσότητα του φαγητού, η θρεπτική του αξία και σε ποιό γεύμα το πρόσθεσε ο χρήστης.

6.4.3 Queries

Για την ανάκτηση και αποθήκευση των δεδομένων χρησιμοποιήθηκαν κάποια queries της γλώσσας MySQL. Τα queries αυτά βρίσκονται ενσωματωμένα στον κώδικα του συστήματος. Πρόκειται για queries τύπου SELECT, INSERT, UPDATE και DELETE. Στο Παράρτημα Α υπάρχουν όλα τα queries που χρησιμοποιήθηκαν.

6.5 Προγραμματισμός Ικανοποίησης Περιορισμών

Για την υλοποίηση της Αυτόματης Διατροφής του χρήστη χρησιμοποιήθηκε προγραμματισμός ικανοποίησης περιορισμών. Πρόκειται για ένα είδος προγραμματισμού στον οποίο οι σχέσεις μεταξύ των μεταβλητών εμφανίζονται σε μορφή περιορισμών και στο οποίο η λύση είναι η ανάθεση τιμών στις μεταβλητές ούτως ώστε να ικανοποιούνται οι περιορισμοί. Είναι δηλαδή μια μορφή δηλωτικού προγραμματισμού στην οποία εμείς δηλώνουμε το πρόβλημα με τις μεταβλητές και τους περιορισμούς και ο επιλυτής (solver) αναλαμβάνει να βρεί την λύση. Οι τιμές που δίνονται στις μεταβλητές προέρχονται από το Domain (πεδίο ορισμού) της κάθε μεταβλητής το οποίο είναι συνήθως πεπερασμένο και το οποίο μπορεί να περιέχει είτε ακέραιες ή πραγματικές τιμές. Οι περιορισμοί περιέχουν ένα υποσύνολο μεταβλητών και προσδιορίζουν τις επιτρεπόμενες τιμές για αυτές τις μεταβλητές.

Η διαδικασία δήλωσης του προβλήματος ονομάζεται μοντελοποίηση και σε αυτή βασίζονται πολλά δύσκολα συνδιαστικά προβλήματα, για επίλυση.

Μια αναμενόμενη λύση του προβλήματος ικανοποίησης περιορισμών είναι ενα σύνολο φαγητών τα οποία ανήκουν στα 6 βασικά γεύματα μιας μέρας. Το πρόγευμα, το 1ο ενδιάμεσο, το μεσημεριανό, τα δύο απογευματινά ενδιάμεσα, και το βραδυνό.
Κάθε βασικό γεύμα πρέπει να περιλαμβάνει τουλάχιστον ένα φαγητό έτσι ο χρήστης είναι απαραίτητο να κάνει επιλογές για κάθε γεύμα.
Επίσης τα φαγητά της λύσης πρέπει να πληρούν τις προϋποθέσεις και το άθροισμα της συνολικής τους θρεπτικής αξίας να είναι μέσα στα επιτρεπόμενα όρια.

6.5.1 Το Προβλήμα

Το πρόβλημα που είχα να επιλύσω στα πλαίσια του συστήματος μου ήταν η εύρεση των κατάλληλων φαγητών και της ποσότητας τους έτσι ώστε το άθροισμα των θρεπτικών τους συστατικών να είναι μέσα στα όρια των ημερησίων απαιτήσεων σε θρεπτικά συστατικά του χρήστη. Τα θρεπτικά συστατικά που λαμβάνονται υπόψη για την εύρεση βέλτιστης λύσης είναι οι υδατάνθρακες, τα λίπη, οι πρωτείνες,οι συνολικές θερμίδες του φαγητού καθώ επίσης και ένα επιπλέον θρεπτικό συστατικό που επιλέγει ο χρήστης (σάκχαρα, ασβέστιο, σίδηρος, θειαμίνη, χοληστερόλη, Βιταμίνη D και Βιταμίνη C).
Μια διατροφή πρέπει να αποτελείται από τα τρία βασικά γεύματα καθώς επίσης και τα ενδιάμεσα. Έτσι ο χρήστης καλείται να κάνει επιλογές για το πρόγευμα του, το μεσημεριανό, το δείπνο και τρία ενδιάμεσα. Οι επιλογές του χρήστη αρχικοποιούν ανάλογα τα Domain των μεταβλητών με τις τιμές των θρεπτικών συστατικών και έτσι δημιουργείται το data file στο οποίο υπάρχουν όλα τα δεδομένα του προγράμματος για ικανοποίηση περιορισμών.
Όταν ο χρήστης επιλέγει ένα τύπο φαγητού, το σύστημα καλεί το ανάλογο query και φέρνει από την βάση δεδομένων όλα τα σχετικά φαγητά και την θρεπτική τους αξία. Μετά τα δεδομένα γράφονται στο αρχείο στους ανάλογους πίνακες πεδίων ορισμού της μεταβλητής για την οποία επιλέχθηκαν.
Το πρόβλημα είναι πρόβλημα μεγιστοποίησης αφού σκοπός είναι η μεγιστοποίηση των ημερήσιων θερμίδων του χρήστη. Για ένα χρήστη που επιθυμεί να χάσει βάρος, οι

ημερήσιες θερμίδες θα είναι λιγότερες από ένα χρήστη που επιθυμεί να διατηρήσει ή να βάλει βάρος. Με αυτή τη λογική, οι θερμίδες για έναν χρήστη που θέλει να βάλει βάρος θα είναι περισσότερες. Έτσι το άθροισμα των θερμίδων των φαγητών σε κάθε γεύμα πρέπει να είναι το μέγιστο, αλλά πρέπει επίσης να είναι μικρότερο ή ίσο από τον αριθμό των προτεινόμενων ημερησίων θερμίδων. Όσο για τα υπόλοιπα θρεπτικά συστατικά, τους υδατάνθρακες, τα λίπη και τις πρωτείνες, αυτά επίσης πρέπει να είναι μέσα στα επιτρεπόμενα όρια.

Οι επιτρεπόμενες ημερήσιες θερμίδες του κάθε χρήστη υπολογίζονται από το BMR του και τον βαθμό της φυσικής του άσκησης. Ο υπολογιμσός αυτός γίνεται κατά την δημιουργία του προφίλ του χρήστη. Οι υδατάνθρακες είναι το 57% των συνολικών ημερήσιων θερμίδων, τα λίπη το 30% και οι πρωτεϊνες το 13%. Όσο για την τιμή του επιλεγόμενου θρεπτικού συστατικού, την εισάγει ο χρήστης.

Η δήλωση των μεταβλητών για μια επιλογή του χρήστη στο πρόβλημα, έγινε με τον εξής τρόπο:

```
int: B1Count;
var 1..B1Count: b1Pos;

set of int: B1Items= 1..B1Count;

array[B1Items] of int: B1Calories;
array[B1Items] of int: B1Carbohydrate;
array[B1Items] of int: B1Fat;
array[B1Items] of int: B1Protein;
int: minB1;
int: maxB1;
var minB1..maxB1: AmountB1;
int: nutrCount=11;
set of int: nutrItems=1..nutrCount;
array[nutrItems] of int: Nutrients;

array[nutrItems] of var int: temp;
```

Τα `B1Count`, `minB1`, `maxB1` και `nutrCount` είναι ακέραιες τιμές οι οποίες καθορίζονται στο data file.

Το `b1Pos` είναι μια μεταβλητή που αντιπροσωπεύει την θέση στον πίνακα με τα θρεπτικά συστατικά που χρησιμοποιείται στους περιορισμούς.

Τα `B1Calories, B1Carbohydrate, B1Fat, B1Protein` είναι οι πίνακες με τις τιμές των θρεπτικών συστατικών των φαγητών, από τα οποία κάποια θα επιλεγούν από τον solver ως λύση του προβλήματος.

Το `B1Items` είναι ένα σύνολο που αποτελείται από τις τιμές `1` μέχρι `B1Count` και οι οποίες δίνουν το μέγεθος των πινάκων. Το ίδιο ισχύει και το `nutrItems` με το αντίστοιχο `nutrCount.`

Το `AmountB1` είναι η μεταβλητή για την ποσότητα του συγκεκριμένου φαγητού που πρέπει να αποφασιστεί.

Το prexix ή suffix `B1` των πιο πάνω δηλώσεων δηλώνει ποια επιλογή του χρήστη είναι αριθμητικά και για ποιό γεύμα προορίζεται. Το `B1` είναι η πρώτη επιλογή για το πρόγευμα. Για το μεσημεριανό θα ήταν `L1` κ.ο.κ.

Το `temp` είναι ένας πίνακας μεταβλητών που κρατά το άθροισμα των θρεπτικών συστατικών κατά την διάρκεια της αναζήτησης λύσης.

Ο πίνακας `Nutrients` είναι αυτός στο οπόιον υπάρχουν τα απαραίτητα ημερήσια θρεπτικά συστατικά για τον συγκεκριμένο χρήστη.

Οι περιορισμοί που δημιουργούνται στο πρόγραμμα σχετικά με τις επιλογές `B1, L1, D1, S1, S2` και `S3` του χρήστη έχουν την ακόλουθη μορφή:

```
constraint temp[1]=
(B1Calories[b1Amount]*AmountB1*15)+(L1Calories[l1Amount]*AmountL1*15
)+(D1Calories[d1Amount]*AmountD1*15)+(S1Calories[s1Amount]*AmountS1*
15)+(S2Calories[s2Amount]*AmountS2*15)+(S3Calories[s3Amount]*AmountS
3*15);
constraint temp[1] < Nutrients[1];

 constraint temp[2]=
(B1Carbohydrate[b1Amount]*AmountB1*15)+(L1Carbohydrate[l1Amount]*Amo
untL1*15)+(D1Carbohydrate[d1Amount]*AmountD1*15)+(S1Carbohydrate[s1A
mount]*AmountS1*15)+(S2Carbohydrate[s2Amount]*AmountS2*15)+(S3Carboh
ydrate[s3Amount]*AmountS3*15);
constraint temp[2] < Nutrients[2]::domain;


constraint temp[3]=
(B1Fat[b1Amount]*AmountB1*15)+(L1Fat[l1Amount]*AmountL1*15)+(D1Fat[d
1Amount]*AmountD1*15)+(S1Fat[s1Amount]*AmountS1*15)+(S2Fat[s2Amount]
*AmountS2*15)+(S3Fat[s3Amount]*AmountS3*15);
constraint temp[3] < Nutrients[3]::domain;
```

```
constraint temp[4]=
(B1Protein[b1Amount]*AmountB1*15)+(L1Protein[l1Amount]*AmountL1*15)+
(D1Protein[d1Amount]*AmountD1*15)+(S1Protein[s1Amount]*AmountS1*15)+
(S2Protein[s2Amount]*AmountS2*15)+(S3Protein[s3Amount]*AmountS3*15);
constraint temp[4] < Nutrients[4]::domain;
```

Κάθε δύο περιορισμοί αντιστοιχούν σε ένα θρεπτικό συστατικό. Ο πρώτος ορίζει το άθροισμα του συγκεκριμένου θρεπτικού συστατικού για την συγκεκριμένη ποσότητα (`AmountB1*15`) και ο δεύτερος ότι το άθροισμα πρέπει να είναι μικρότερο από την απαραίτητη ημερήσια ποσότητα θρεπτικού συστατικού του χρήστη.

Τέλος με την δήλωση του επιλυτή (solver)

```
solve maximize temp[1];
```

γίνεται η αναζήτηση της λύσης για μεγιστοποίηση του αθροίσματος των θερμίδων που βρίσκεται στην μεταβλητή `temp[1]`.

6.5.2 Ενσωμάτωση στο σύστημα, εκτέλεση και εύρεση λύσης

Το πρόβλημα ικανοποίησης περιορισμών δημιουργείται δυναμικά κατά την εκτέλεση της λειτουργίας της αυτόματης διατροφής. Συγκεκριμένα, δημιουργείται ανάλογα με τις προτιμήσεις που εισάγει ο χρήστης σχετικά με την διατροφή του. Έτσι έχουμε μια δυναμική δημιουργία ενός String το οποίο γράφεται στο αρχείο "quick-nutrition.mzn".
Την ίδια διαδικασία ακολουθεί και η δημιουργία του αρχείου δεδομένων του προγράμματος ικανοποίησης περιορισμών και γράφεται στο αρχείο "quick-nutrition.dzn".

Η εκτέλεση των προγραμμάτων MiniZinc γίνονται από το ειδικό για την γλώσσα "MiniZinc 1.5.1 Command Prompt". Έτσι για την εκτέλεση του προγράμματος μέσα από το πρόγραμμα σε Java, έπρεπε να εκτελεστούν συνολικά τρείς εντολές οι οποίες επεξηγόνται παρακάτω:

- "C:\Program Files (x86)\MiniZinc 1.5.1\bin\mzncmd"

Η εντολή αυτή ανοίγει το "MiniZinc 1.5.1 Command Prompt".

- "cd C:\Users\Snowflake\Documents\NetBeansProjects\MyNutritionApplication\"

Η εντολή αυτή κατευθύνει το Command Prompt στον φάκελο στον οποίο βρίσκεται το πρόγραμμα σε MiniZinc.

- minizinc quick-nutrition.dzn quick-nutrition.mzn -o output.txt

Η εντολή που εκτελέι το πρόγραμμα και γράφει το αποτέλεσμα στο αρχείο output.txt

Για την εκτέλεση αυτών των εντολών χρησιμοποιείται το batch file "command_file.bat" στο οπόιο υπάρχουν οι εντολές και το οποίο καλείται να εκτελεστεί από το πρόγραμμα στην Java με την εξής εντολή:

```
try {
        Process process = Runtime.getRuntime ().exec ("cmd /k command_file.bat");

    } catch (IOException ex) {
       Logger.getLogger(QuickNutrition.class.getName()).log(Level.SEVERE, null, ex);
    }
```

Η μορφή του αρχείου "command_file.bat" έχει ως εξής:

```
@echo off
CALL "C:\Program Files (x86)\MiniZinc 1.5.1\bin\mzncmd"
cd C:\Users\Snowflake\Documents\NetBeansProjects\MyNutritionApplication\
minizinc quick-nutrition.dzn quick-nutrition.mzn -o output.txt
EXIT /B
```

Τα αποτελέσματα της εκτέλεσης του προγράμματος γράφονται στο αρχείο "output.txt" το οποίο διαβάζεται από το πρόγραμμα στην Java και εκτελεί τις απαραίτητες ενέργειες.
Σε περίπτωση που το αποτέλεσμα είναι η λύση του προβλήματος τότε τα δεδομένα διαβάζονται, γίνεται η αντιστοίχιση τους με τα φαγητά και τα γεύματα και εμφανίζονται

στην φόρμα της διατροφής με όλες τις διατροφικές πληροφορίες. Ένα παράδειγμα της εμφάνισης της λύσης δίνεται στο Παράρτημα Β, όπου υπάρχει Εχγειρίδιο Χρήσης.
Σε περίπτωση που το αποτέλεσμα δεν είναι λύση, το πρόγραμμα ξανα-αρχικοποιεί τους πίνακες με τα domain, ξανα-δημιουργεί το αρχείο με τα δεδομένα "quick-nutrition.dzn" και ξανατρέχει το πρόγραμμα ικανοποίησης περιορισμών μέχρι να βρεί μια λύση.

6.6 Δυσκολίες Υλοποίησης και Εύρεση Λύσης

Τα πλείστα δύσκολα κομμάτια της υλοποίησης του συστήματος αυτού αφορούσαν στην υλοποίηση της λειτουργίας της αυτόματης διατροφής. Για αυτό το κομμάτι της εργασίας έπρεπε να αναλυθούν τα δεδομένα της βάσης δεδομένων και να δημιουργηθούν ειδικά queries τα οποία να διαλέγουν τα φαγητά από κάθε κατηγορία τα οποία να είναι βρώσιμα σύμφωνα με την περιγραφή τους. Με αυτό εννοώ ότι έπρεπε να έχουν κάποιες ιδιότητες όπως για παράδεγμα να είναι μαγειρεμένα, να μην είναι κονσέρβες, να μην έχουν σπόρια κλπ. Για να υλοποιηθεί αυτό το κομμάτι αναλύθηκε η κάθε ομάδα τροφίμων ξεχωριστά και βρέθηκαν ποιες λέξεις δεν έπρεπε να εμφανίζονται στην περιγραφή του φαγητού. Τα ειδικά διαμορφωμένα queries βρίσκονται στο Παράρτημα Α.

Το δεύτερο κομμάτι που θεώρησα δύσκολο ήταν η απόφαση για το πως θα αρχικοποιηθούν τα Domain για τις ποσότητες των φαγητών (π.χ. B1Amount). Αρχικά στο πρόγραμμα ικανοποίησης περιορισμών είχα δηλωσει τις μεταβλητές αυτές σαν var int που παίρνουν τιμές από το 1 μέχρι το 1000, εννοώντας την ποσότητα 1 γρ με 1000 γρ. Αυτό έκανε το πρόγραμμα να τρέχει για περισσότερο από μισή ώρα αφού έπρεπε να δοκιμάσει όλες τις τιμές (από το 1-1000) και να αποφασίσει ποια είναι η κατάλληλη (εαν υπήρχε τελικά λύση).
Για επίλυση αυτού του προβλήματος έθεσα την μεταβλητή Amount κάθε φαγητού να έχει μια min και μια max τιμή. Έτσι δεν θα παίρνει τιμές από το 1-1000 αλλά από το min-max. Τα min και max είναι διαφορετικά για κάθε είδος φαγητού, ανάλογα με την κατηγορία τροφίμων στην οποία ανήκουν. Αυτό συμβαίνει γιατί για παράδειγμα το γάλα ζυγίζει πολύ περισσότερο από μια φέτα ψωμί και έχουν διαφορετικές απαιτήσεις σε ποσότητα. Επίσης για να είναι ακόμη πιο γρήγορη η εκτέλεση του προγράμματος, η μεταβλητή

πολλαπλασιάζεται με την σταθερά 15 ούτως ώστε να γίνεται επιλογή γραμμαρίων 15 γρ, 30 γρ, 45 γρ κ.ο.κ.

Επίσης πρόβλημα υπήρξε με τα πεδία των θρεπτικών συστατικών τα οποία αρχικά ήταν πολύ μεγάλα, με αποτέλεσμα να τρέχει το πρόβλημα για αρκετή ώρα χωρίς να βρίσκει λύση. Αποφάσισα να μικρύνω κατά πολύ τα πεδία και να έχουν δηλαδή μέγεθος το πολύ 5. Οι τιμές των πεδίων επιλέγονται τυχαία από τις διαθέσιμες τιμές για κάθε επιλογή του χρήστη. Έγινε και προσπάθεια για επιλογή των τιμών από το μέσο των διαθέσιμων τιμών. Η προσπάθεια αυτή ήταν ανεπιτυχής αφού οι τιμές που επέστρεφε ήταν πάντα οι ίδιες με αποτέλεσμα ο χρήστης να μην παίρνει μια διαφορετική προτεινόμενη διατροφή. Έτσι η χρήση της τυχαίας επιλογής των τιμών προτιμήθηκε από την χρήση της επιλογής από την μέση του πεδίου.

Πιο κάτω φαίνεται ο κώδικας για επιλογή από το μέσο του πεδίου:

```
int []getMedIndex(int count){σ
        ArrayList<Integer> numbers = new ArrayList<Integer>(count);
        for(int i = 0; i < count; i++)
        {
           numbers.add(i);
        }
        int min=(count/2)-2;
        int max=(count/2)+2;

        int counter=0;
        for(int j =min; j < max; j++)
        {
           counter++;
        }
        if((counter%2)==0){
           counter++;
           max++;
        }
        System.out.println("Counter for Indexes: "+counter);
```

```
        int[] array=new int[counter];
        for(int j =min; j < max; j++)
        {
            array[j]=numbers.get(j);
        }
        return array;
    }
```

Σύμφωνα με τον πιο πάνω κώδικα γίνεται η επιλογή των θέσεων των φαγητών από τον πίνακα διαθέσιμων φαγητών, τα οποία επιλεγμένα φαγητά θα αξιολογηθούν αν ικανοποιούν τα κριτήρια επιλογής.

Με παρόμοιο τρόπο, ο κώδικας για επιλογή τυχαία των θέσεων των φαγητών που θα αξιολογηθούν παρουσιάζεται πιο κάτω:

```
int []getRandomIndexes(int count)
{
    int[] array=new int[5];
        ArrayList<Integer> numbers = new ArrayList<Integer>(count);
        for(int i = 0; i < count; i++)
        {
           numbers.add(i);
        }
      Collections.shuffle(numbers);
      for(int j =0; j < 5; j++)
      {
        array[j]=numbers.get(j);
      }
      return array;
 }
```

Με την αξιολόγηση των επιλεγμένων φαγητών εάν το πρόγραμμα βρεί λύση θα τερματίσει χωρίς να αφήσει τον χρήστη να περιμένει πολύ. Εαν δεν υπάρχει όμως λύση, τα πεδία θα ξανά-αρχικοποιηθούν με νέες τιμές και το πρόγραμμα θα ξανά τρέξει. Η διαδικασία αυτή επαναλαμβάνετε μέχρι να βρεθεί μια λύση.

Επιπλέον σε περίπτωση που το πρόγραμμα δεν βρει λύση μετά από εκτέλεση περισσότερης των 20 δευτερολέπτων το πρόγραμμα θα ενημερώσει τον χρήστη ότι δεν υπάρχει διατροφή κατάλληλη για τις προτιμήσεις του και θα του επιδείξει να κάνει νέες επιλογές.
Τέλος υπήρξε δυσκολία διαβάσματος του αρχείου "output.txt" γιατί η έξοδος του προγράμματος ικανοποίησης περιορισμών δεν γραφόταν στο αρχείο έγκαιρα για να διαβαστεί από το πρόγραμμα της Java. Έτσι δημιούργησα μια βοηθητική συνάρτηση για καθυστέρηση της εκτέλεσης του κώδικα για διάβασμα του αρχείου για n δευτερόλεπτα όπου το n είναι μέχρι και 3.

Η συνάρτηση έχει ως εξής:

```
public static void waiting (int n){

    long t0, t1;

    t0 =  System.currentTimeMillis();

    do{
      t1 = System.currentTimeMillis();
    }
    while ((t1 - t0) < (n * 1000));
  }
```

Κεφάλαιο 7

Συμπεράσματα

7.1 Εισαγωγή

7.2 Συμπεράσματα

7.3 Μειονεκτήματα Συστήματος

7.4 Μελλοντική Εργασία

7.1 Εισαγωγή

Σε αυτό το κεφάλαιο αναγράφονται τα συμπεράσματα που προκύπτουν από την υλοποίηση της διπλωματικής αυτής εργασίας. Επίσης αναφέρονται τα μειονεκτήματα του συστήματος και το πως θα μπορούσε να επεκταθεί και να βελτιωθεί στο μέλλον.

7.2 Συμπεράσματα

Με την υλοποίηση του συστήματος αυτού επιτεύχθηκε ο σκοπός της διπλωματικής αυτής ο οποίος ήταν η δημιουργία ενός συστήματος το οποίο να βοηθά τον χρήστη να ελέχγει τις διατροφικές του συνήθειες. Με την λειτουργία της δημιουργίας διατροφής ο χρήστης μπορεί να βρίσκει ένα φαγητό που έχει σκοπό να φάει και την ποσότητα του, να το προσθέτει στην διατροφή του και να βλέπει αν αυτό το φαγητό είναι κατάλληλο. Αν όχι θα μπορεί να ρυθμίζει την ποσότητα στα δικά του μέτρα ούτως ώστε να δικαιούται να το κρατήσει στο διαιτολόγιο του. Με κάθε πρόσθεση νέου φαγητού στην διατροφή του, βλέπει πόσες είναι οι διαθέσιμες θερμίδες για να μπορεί να διαλέξει το επόμενο του γεύμα με τον ίδιο τρόπο. Επίσης το σύστημα έχει την δυνατότητα να φυλάει τις διατροφές, σε περίπτωση που ο χρήστης θέλει να δει τι έφαγε τις προηγούμενες μέρες και να τα αποφύγει την τρέχον μέρα.

Η λειτουργία της αυτόματης διατροφής, διευκολύνει τον χρήστη, αφού σύμφωνα με τις δικές του προτιμήσεις ετοιμάζει μια διατροφή που είναι κατάλληλη για αυτόν. Σε

περίπτωση που δεν υπάρχει κατάλληλη διατροφή, ο χρήστης καλείται να κάνει άλλες επιλογές.

Τέλος παρατηρούμε ότι η επιλογή του προγραμματισμού ικανοποίησης περιορισμών για την ευρεση λύσης στο πρόβλημα ήταν καλή επιλογή αφού όντως επιστρέφει κάποια λύση μετά την εκτέλεση του. Η χρήση του ήταν ένας τρόπος να επαναεπιβεβαιωθεί το γεγονός ότι μπορεί να χρησιμοποιηθεί για την επίλυση πολλών προβλημάτων που συναντούμε πολύ συχνά στην ζωή μας.

Με την ολοκλήρωση της εργασίας αυτής, δημιουργήθηκε ένα χρήσιμο εργαλείο για κάθε χρήστη που επιθυμεί να μπεί σε ένα πρόγραμμα διατροφής και να επιτεύξει τους στόχους που έχει θέσει.

7.3 Μειονεκτήματα Συστήματος

Το κύριο μειονέκτημα του συστήματος είναι το ότι πρόκειται για desktop application και όχι web. Η ύπαρξη της βάσης δεδομένων κάνει δύσκολη την εγκατάσταση του συστήματος σε οποιονδήποτε υπολογιστή. Είναι απαραίτητο δηλαδή να υπάρχει εγκατεστημένη η MySQL. Αυτό θα αποφευγόταν εαν ήταν web application αφού η βάση θα υπήρχε πάνω σε έναν μόνο υπολογιστή (database server).

7.4 Μελλοντική Εργασία

Με την υλοποίηση του συστήματος αυτού επιβεβαιώνεται το γεγονός ότι είναι εφικτή η δημιουργία λογισμικού που να στοχέυει στην δημιουργία βέλτιστης διατροφής. Έτσι η επέκταση του σύστηματος, στο μέλλον, και η εισαγωγή περισσότερων λειτουργιών δεν είναι καθόλου απίθανη. Κάποιες λειτουργίες που σίγουρα θα βελτίωναν την ποιότητα του συστήματος περιγράφονται πιο κάτω.

Το σύστημα θα μπορούσε να λειτουργεί σαν προσωπικός διατροφολόγος. Αυτό μπορεί να γίνει πράξη με την δημιουργία επιπλέον λειτουργιών οι οποίες να είναι πιο εξειδικευμένες σε θέματα διατροφής. Θα μπορούσε δηλαδή το σύστημα να παρέχει λειτουργίες για εύρεση διατροφής κατάλληλη για διαβητικούς. Ακόμη θα μπορούσε να παρέχει συμβουλές στον χρήστη. Για παράδειγμα εαν πρόσθεσει ένα φαγητό στην διατροφή του, και αυτό ξεπερνά τις απαιτήσεις του χρήστη σε ενέργεια, τότε ίσως να μην τον αναγκάσει να το

αφαιρέσει φτάνει οι επόμενες n διατροφές που θα δημιουργήσει να περιέχουν ανάλογες τροφές. Επίσης θα μπορούσε να υπολογίζει μέχρι πότε είναι εφικτό να επιτευχθούν οι στόχοι του χρήστη.

Λειτουργίες όπως καταγραφή των μετρήσεων του σώματος του χρήστη και του βάρους του ανά τακτά χρονικά διαστήματα, θα ήταν επίσης χρήσιμα για τον χρήστη, αλλά και για το σύστημα. Θα έδινε αρκετές πληροφορίες στο σύστημα για τον καλυτερο υπολογισμό του πότε αναμένεται η επίτευξη του στόχου.

Τέλος για να παρέχεται μεγαλύτερη ευχρηστία στον χρήστη, το σύστημα πρέπει να υποστηρίζει περισσότερες και πιο σύνθετες λειτουργίες. Για παράδειγμα, θα μπορούσε να καταγράφει συγκεκριμένα είδη άσκησης και πως αυτά επιρεάζουν τις διατροφικές του συνήθειες, μακρυπρόθεσμα ή βραχυπρόθεσμα.

Βιβλιογραφία

[1] Addison-Wesley, Ian Sommerville, "Software Engineering", (8th ed., 2008)

[2] Δρ. Ανδρέου Ανδρέα, Σημειώσεις Μαθήματος «ΕΠΛ 361 Τεχνολογία Λογισμικού», Ενότητες 3 - 9

[3] Stephen R. Schach, McGraw-Hill, "Object-Oriented and Classical Software Engineering", 7th Edition New York, 2007

[4] Department of Health, "Dietary Reference Values For Food Energy and Nutrients for the United Kingdom", Report of the Panel on Dietary Reference Values of the Commitee on Medical Aspects of Food Policy, Report on Health and Social Subjects 41

[5] http://ndb.nal.usda.gov/

[6] http://www.g12.csse.unimelb.edu.au/minizinc/

[7] http://www.bmi-calculator.net/bmr-calculator/harris-benedict-equation/

[8] http://www.brianmac.co.uk/nutrit.htm

[9] http://web.njit.edu/all_topics/Prog_Lang_Docs/cplex80/doc/userman/html/callableLibrary31.html

[10] http://financialsoft.about.com/od/howtochoosetaxsoftware/tp/Online_or_Desktop_Tax_Sofware.htm

[11] http://www.daniweb.com/software-development/java/threads/37743/executing-dos-commands-in-java

[12] http://www.linglom.com/2007/06/06/how-to-run-command-line-or-execute-external-application-from-java/

[13] http://en.wikipedia.org/wiki/Nutrition

[14] http://www.choosemyplate.gov/food-groups/

[15] http://www.nichd.nih.gov/health/topics/diet_and_nutrition.cfm

[16] http://en.wikipedia.org/wiki/Basal_metabolic_rate

[17] http://www.hsph.harvard.edu/nutritionsource/what-should-you-eat/carbohydrates-full-story/index.html#what-are-carbohydrates

[18] http://en.wikipedia.org/wiki/Body_mass_index

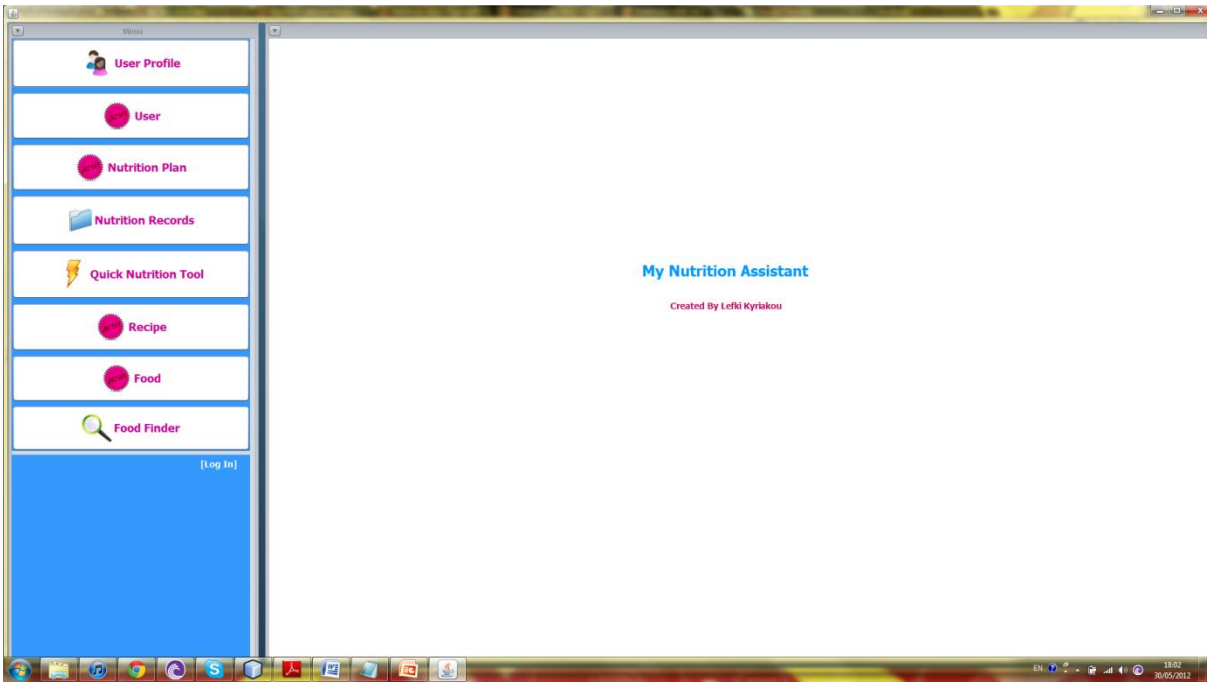
Παράρτημα Α

Εγχειρίδιο Χρήσης

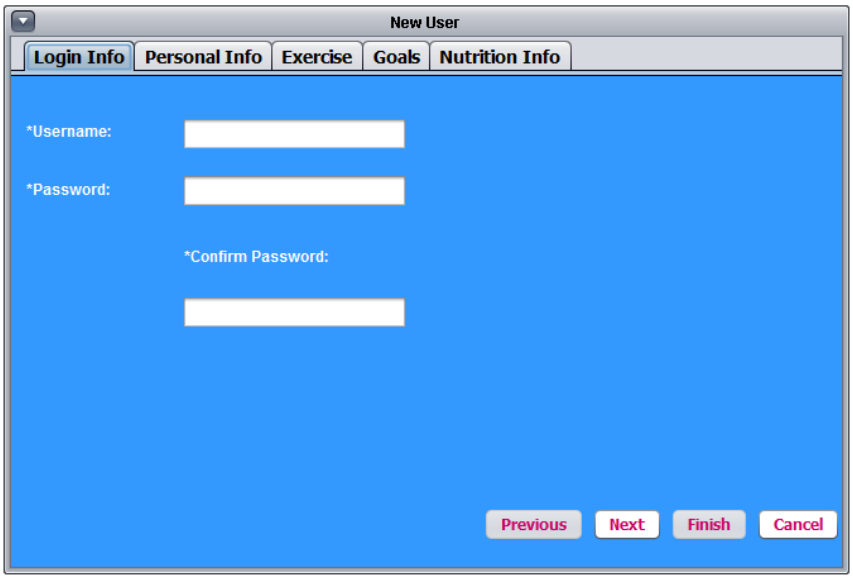
Κατά την εκκίνηση του συστήματος εμφανίζεται η εξής οθόνη στον υπολογιστή. Πρόκειται για το κεντρικό παράθυρο του συστήματος.

Α.1 Εισαγωγή Νέου Χρήστη

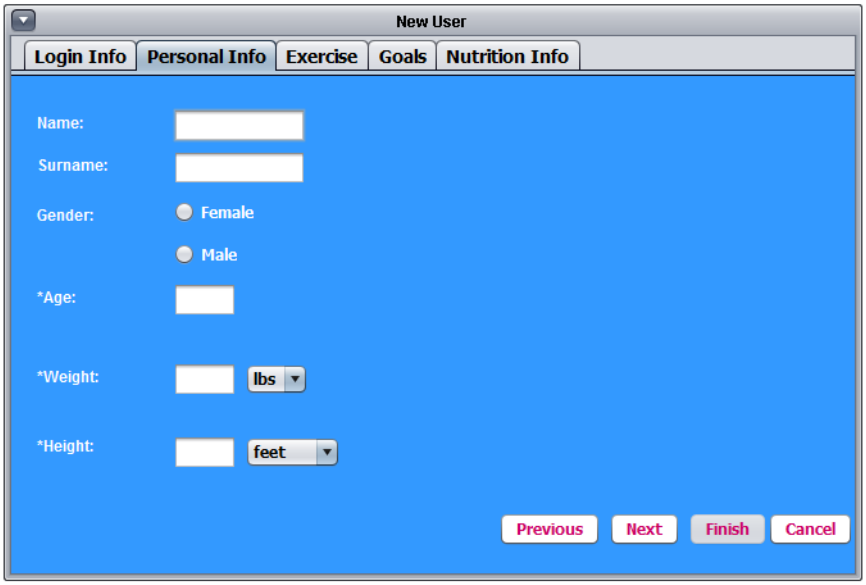
1. Για την δημιουργία Νέου χρήστη επιλέξτε "New User".



Με την επιλογή νέου χρήστη ανοίγει το ακόλουθο παράθυρο.

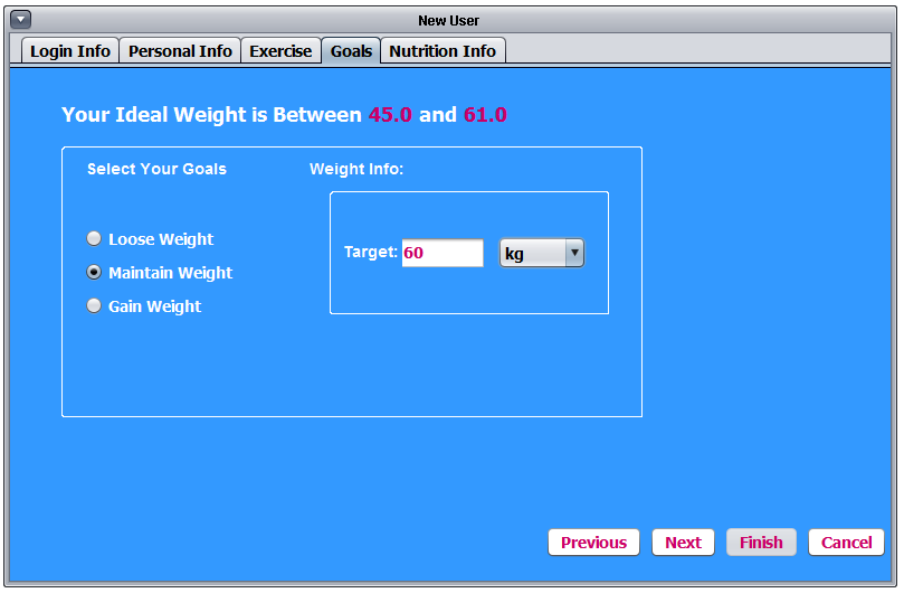


2. Εισάγετε ένα Username και Password. Επιλέξτε "Next".



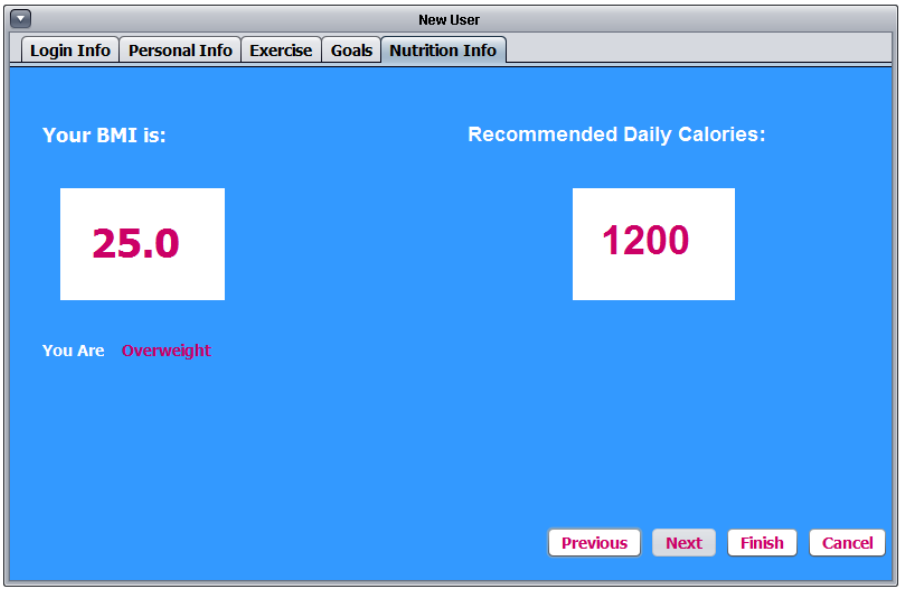
3. Εισάγετε τα προσωπικά σας στοιχεία. Η ηλικία, το βάρος και το ύψος είναι απαραίτητο να εισαχθούν. Επιλέξτε "Next".

4. Επιλέξτε το επίπεδο της φυσικής σας άσκησης και στη συνέχεια "Next".

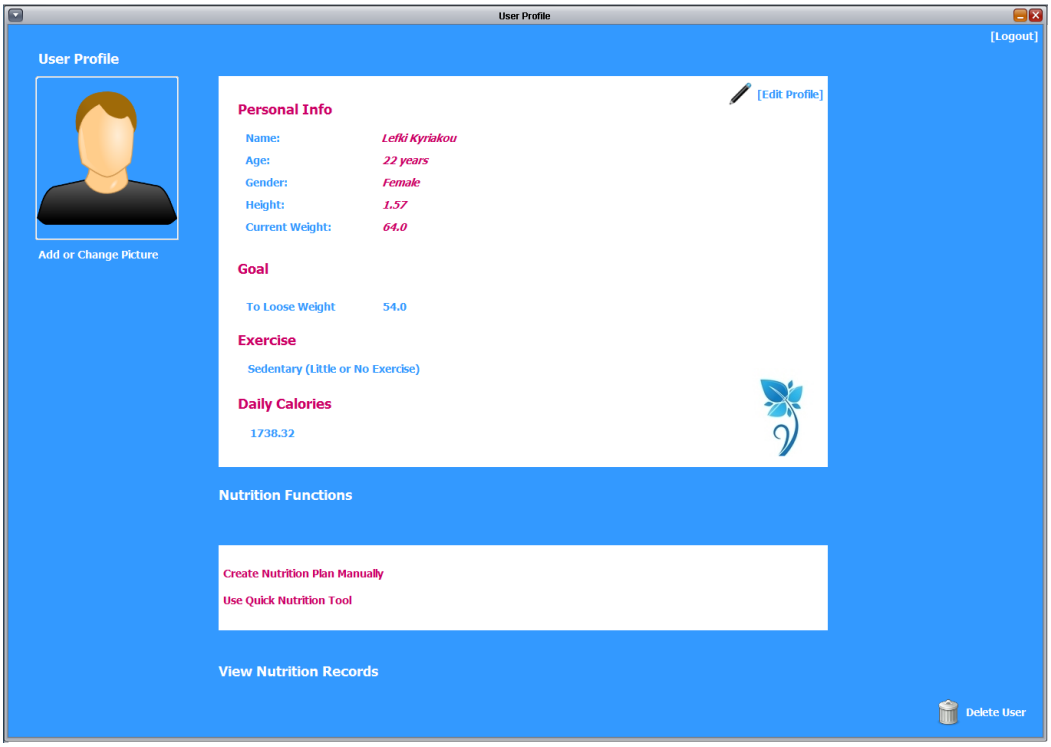


5. Επιλέξτε το στόχο σας και το επιθυμητό βάρος το οποίο να είναι μέσα στα όρια που αναγράφονται. Επιλέξτε "Next".

Θα ανοίξει το παράθυρο με τις προτεινόμενες θερμίδες και το τρέχον BMI σας.

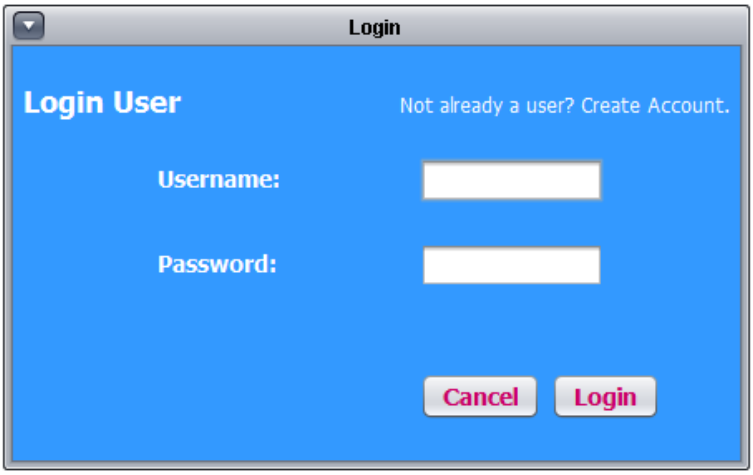


6. Επιλέξτε "Finish" για την δημιουργία του προφίλ σας. Με την επιλογή αυτή προβάλεται αυτόματα το προφίλ σας.



A.2 Log In

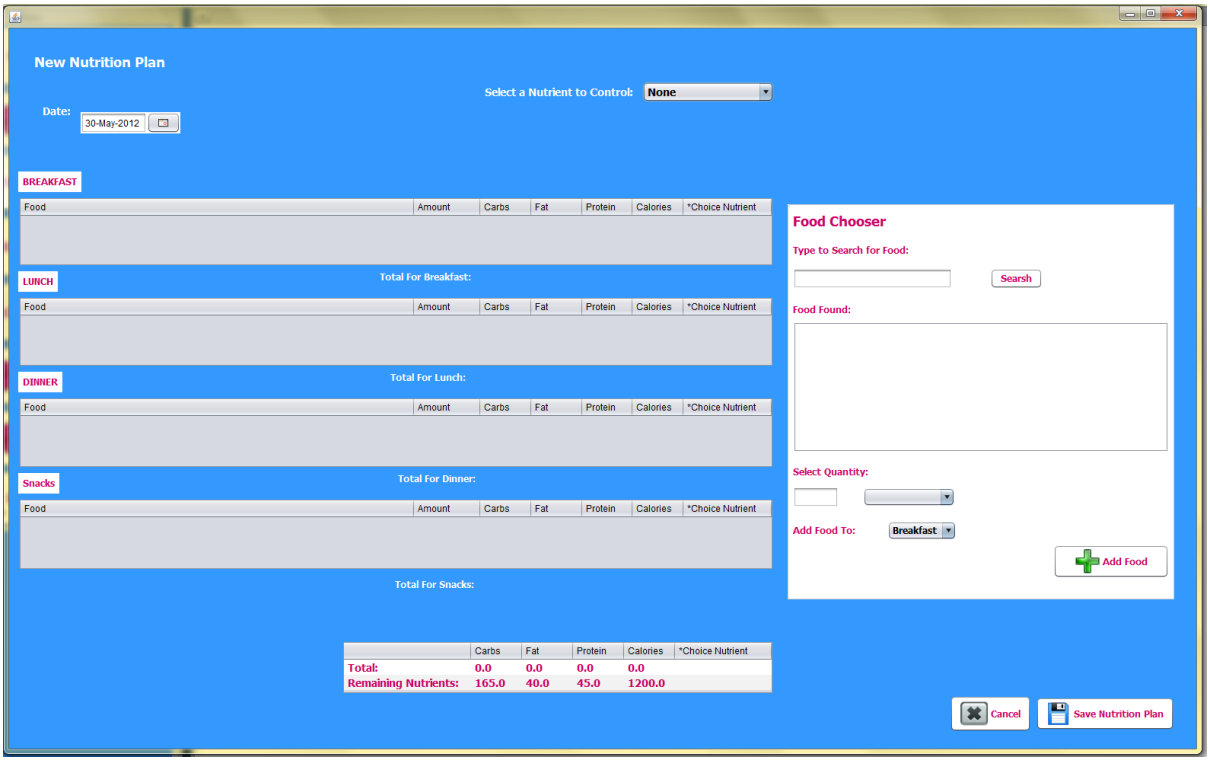
Από την κεντρική σελίδα επιλέξετε Log In. Θα ανοίξει τα ακόλουθο παράθυρο.



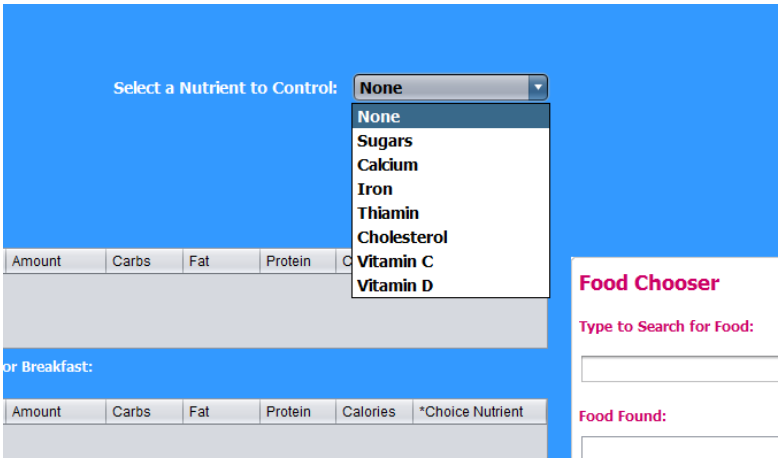
1. Εισάγετε τα στοιχεία σας και επιλέξτε "Login".

A.3 Δημιουργία Νέας Διατροφής

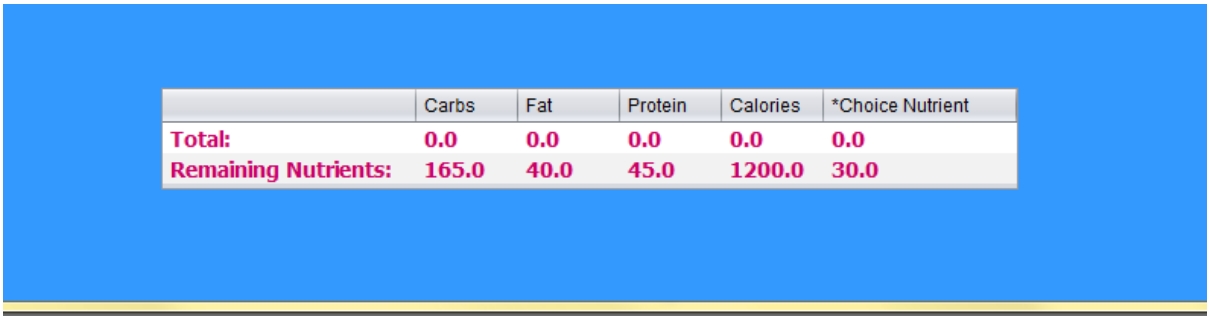
1. Από το προφίλ σας επιλέξτε "Create Nutrition Plan Manually" ή από το κεντρικό παράθυρο "New Nutrition Plan". Θα ανοίξει το ακόλουθο παράθυρο.



Στον πίνακα στο κάτω μέρος του παραθύρου αναγράφονται τα επιτρεπόμενα θρεπτικά συστατικά που μπορείτε να έχετε στην ημερήσια αυτή διατροφή. Μπορείτε να επιλέξετε ένα επιπλέον θρεπτικό συστατικό για να ρυθμίσετε στην διατροφή σας από την επιλογή "Select a Nutrient to Control".

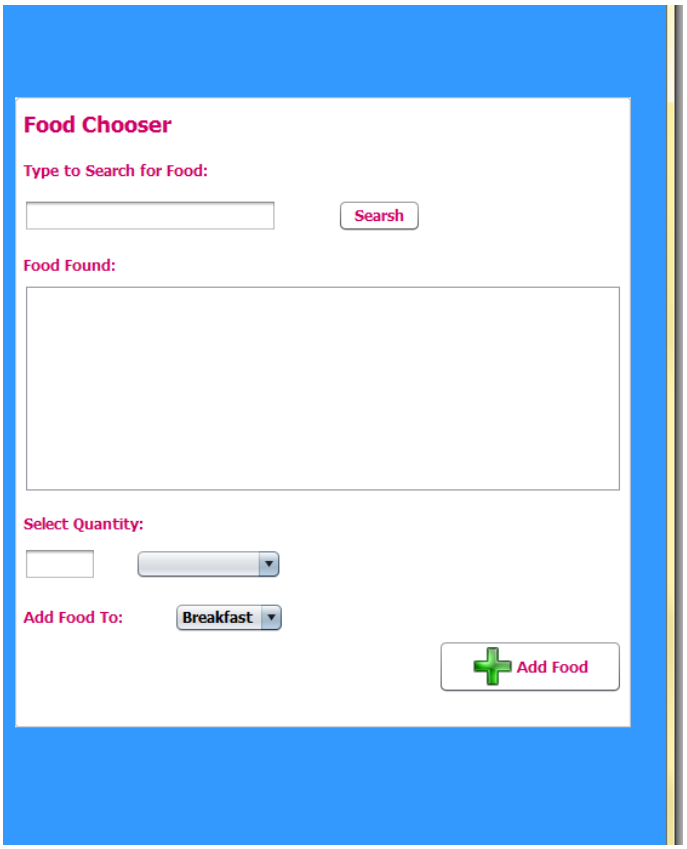


Όταν γίνει επιλογή του επιπλέον θρεπτικού συστατικού τα επιτρεπόμενα του όρια εμφανίζονται και αυτά στον πίνακα στο κάτω μέρος του παραθύρου.

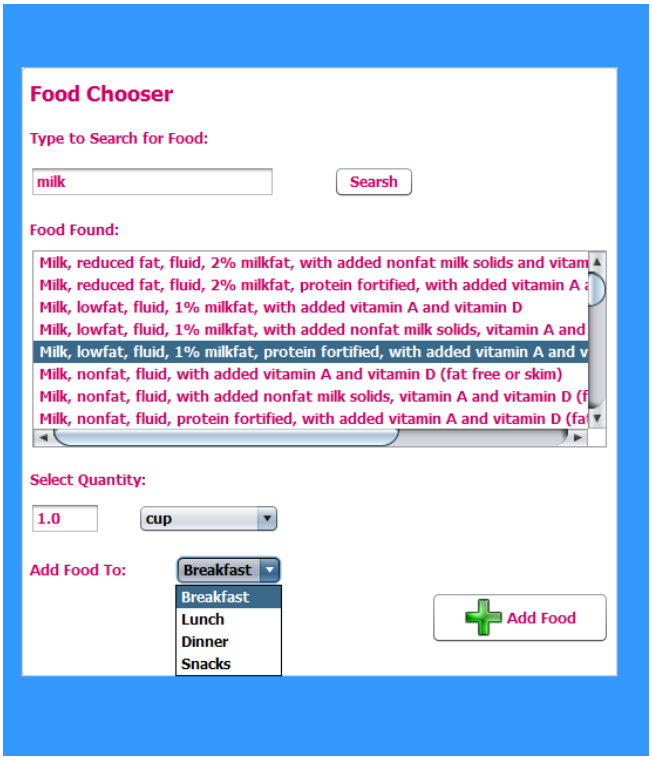


Στο δεξί μέρος της σελιδας υπάρχει το "Food Chooser".

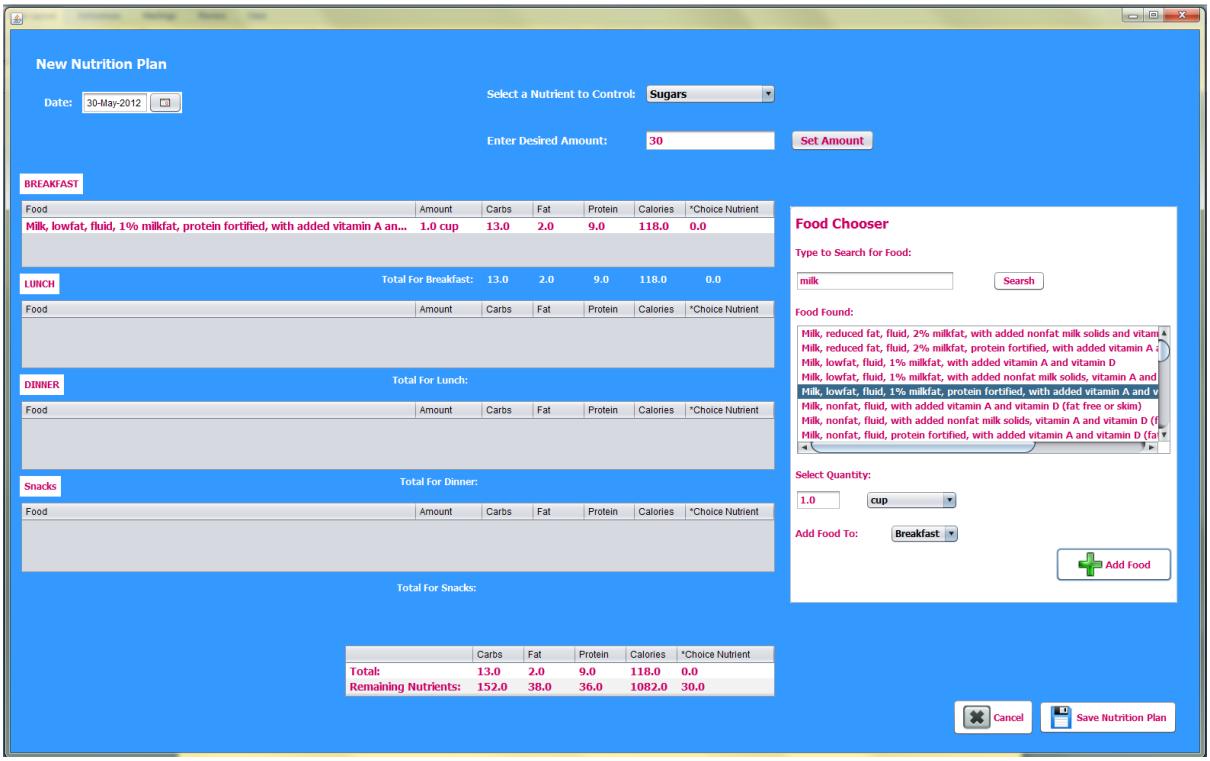
2. Πληκτρολογείστε ένα φαγητό και επιλέξτε "Search". Στην λίστα εμφανίζονται τα αποτελέσματα της αναζήτησης.



3. Επιλέξτε ένα φαγητό από την λίστα και στην συνέχεια μια ποσότητα.



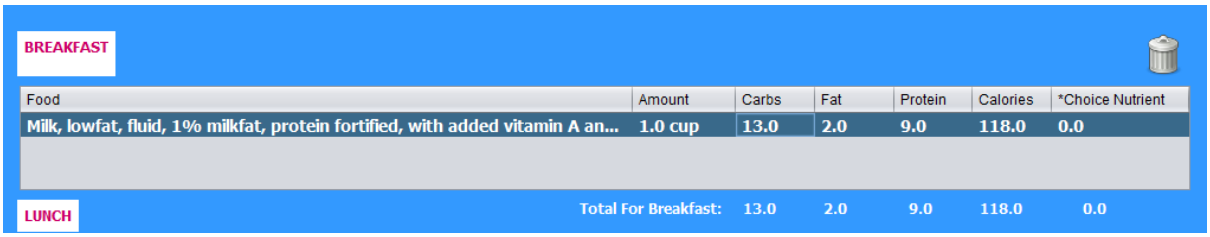
4. Επιλέξτε σε ποιό γεύμα θέλετε να προσθέσετε το φαγητό αυτό και στην συνέχεια "Add Food".



Όπως βλέπετε το φαγητό έχει προστεθεί στην λίστα του Προγεύματος και τα συνολικά θρεπτικά συστατικά έχουν υπολογιστέι καθώς επίσης και τα θρεπτικά συστατικά που έχει ακόμα διαθέσιμα ο χρήστης.

5. Επαναλάβετε τα βήματα 2-4 όσες φορές επιθυμείτε μέχρι να έχετε μια ικανοποιητική διατροφή.

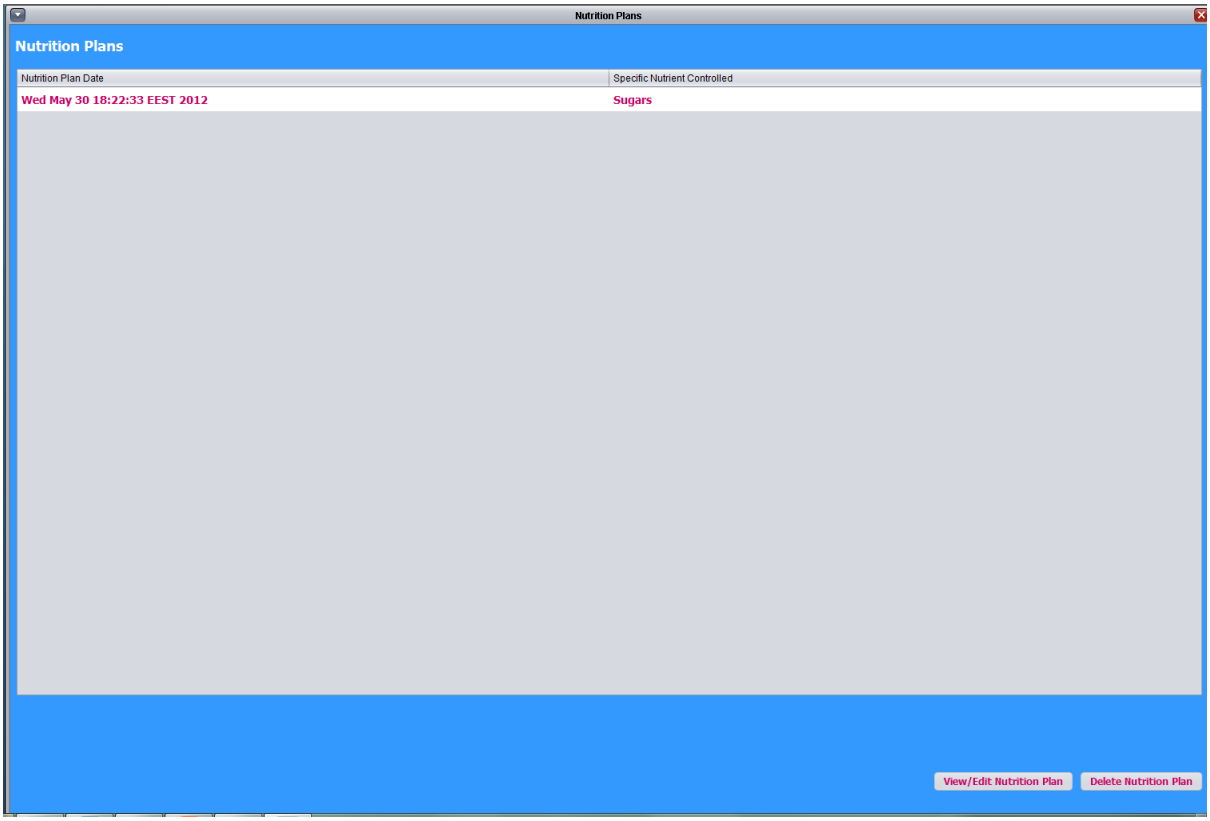
6. Έαν δεν αλλάξετε γνώμη για κάποιο φαγητό επιλέξτε το και μετά πατήστε το εικονίδιο του κάλάθου αχρήστων που εμφανίζετε. Έτσι διαγράφετε το φαγητό από την διατροφή σας.



7. Έαν έχετε τελειώσει και θέλετε να φυλάξετε την διατροφή σας επιλέξτε "Save Nutrition Plan".

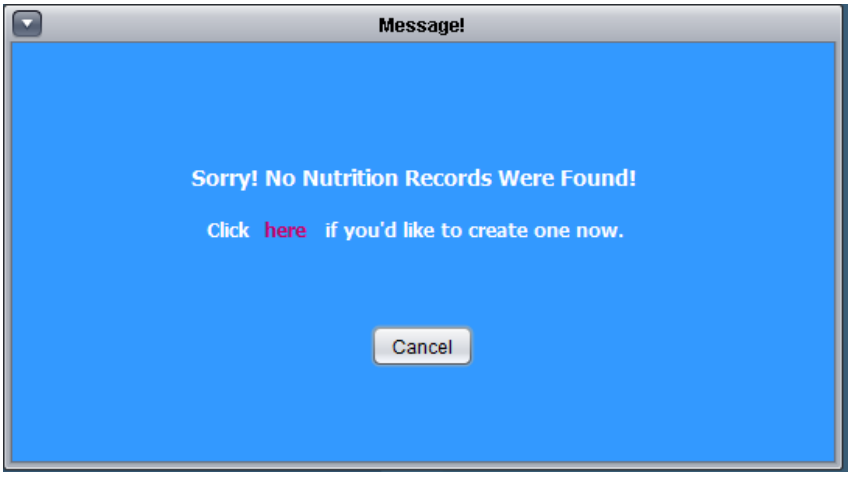
Α.4 Προβολή των Διατροφών του Χρήστη

1. Από το προφίλ σας επιλέξτε "View Nutrition Records" ή από το κεντρικό παράθυρο "Nutrition Records". Θα ανοίξει το ακόλουθο παράθυρο.



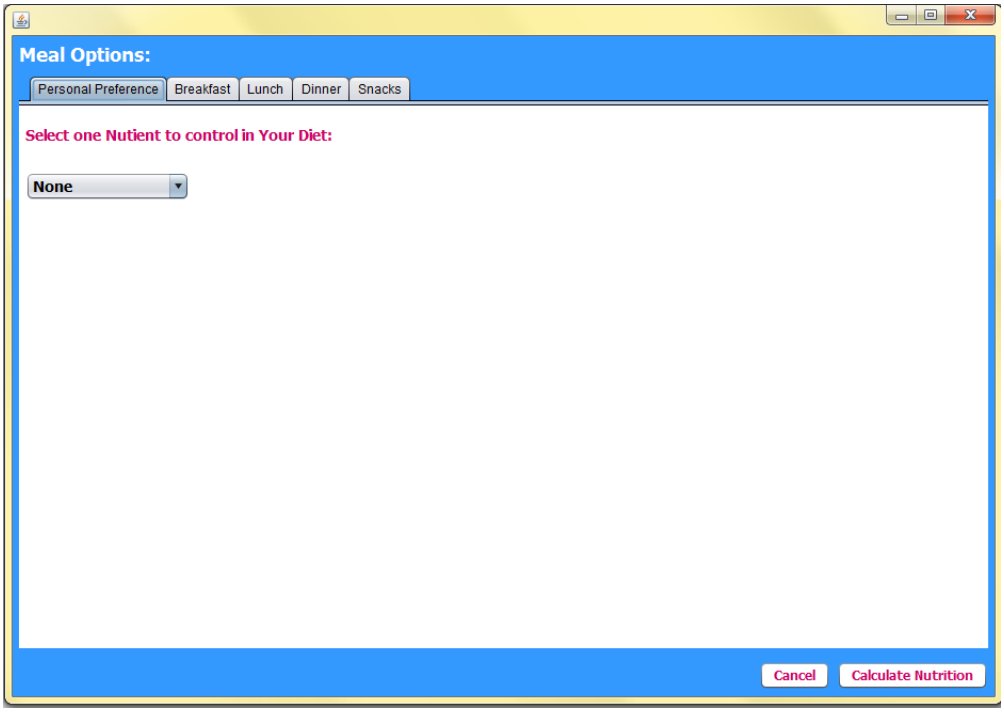
2. Επιλέξετε μια διατροφή και στην συνέχεια "View/Edit Nutrition Plan" ή "Delete Nutrition Plan".

3. Εαν δεν υπάρχουν διατροφές καταχωρημένες, θα εμφανιστεί το εξής παράθυρο από το οποίο θα μπορείτε να επιλέξετε να δημιουργήσετε μια νέα διατροφή έαν το επιθυμείτε.

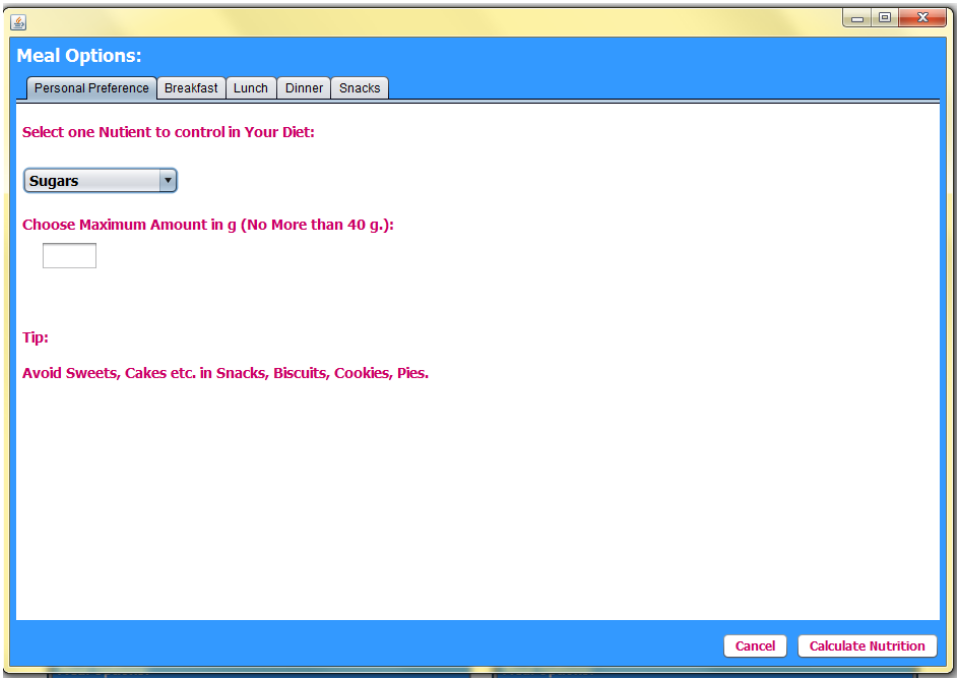


A.5 Δημιουργία Αυτόματης Διατροφής

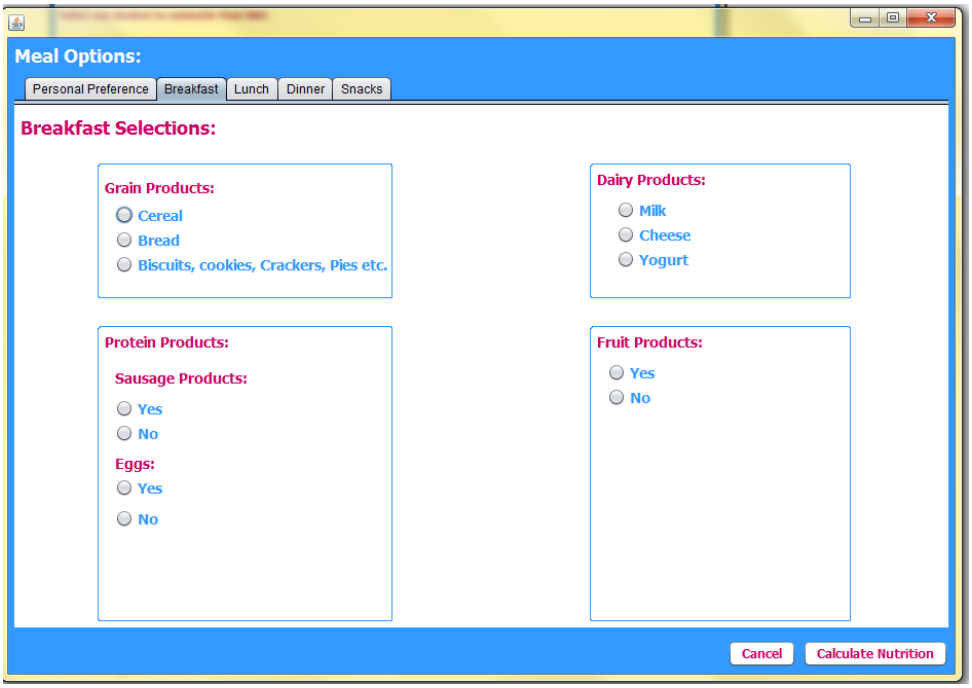
1. Από το προφίλ σας επιλέξτε "Use Quick Nutrition Tool" ή από το κεντρικό παράθυρο "Quick Nutrition Tool". Θα ανοίξει το ακόλουθο παράθυρο.



2. Κάνετε μια επιλογή από τα θρεπτικά συστατικά που υπάρχουν στην λίστα σε περίπτωση που επιθυμείται να ρυθμίσετε την ποσότητα κάποιου συγκεκριμένου στην διατροφή σας.



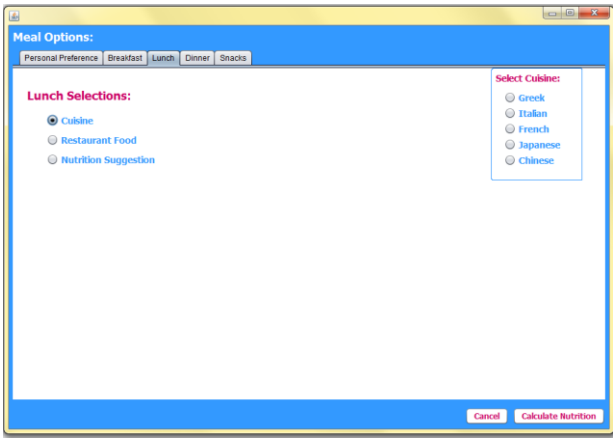
3. Επιλέξετε την επιθυμητή ποσότητα του θρεπτικού συστατικού. Συνεχίστε στην καρτέλα "Breakfast".

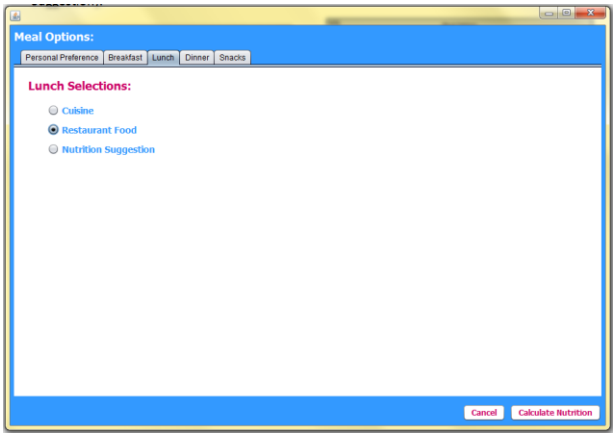


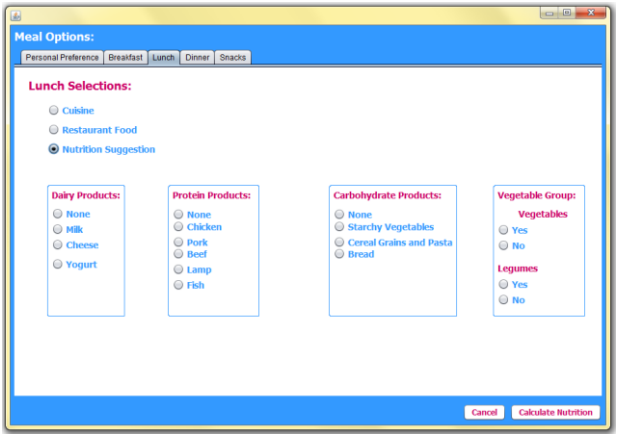
4. Εισάγετε τις επιλογές σας για το πρόγευμα. Πρέπει να επιλέξετε ένα φαγητό από κάθε ομάδα. Πρέπει να γίνει επιλογή τουλάχιστον από τα "Grain Products" και "Dairy Products".

5. Συνεχίστε στην καρτέλα "Lunch".

6. Εισάγετε τις επιλογές σας για το μεσημεριανό. Μπορείτε να διαλέξετε φαγητό κουζίνας (Ελληνικής, Κινέζικης, Ιαπωνικής, Γαλλικής, Ιταλικής), φαγητό από εστιατόριο, ή να διαλέξετε εσείς τις ομάδες τροφίμων από τις οποίες να αποτελείται το γεύμα σας (Nutrition Suggestion).





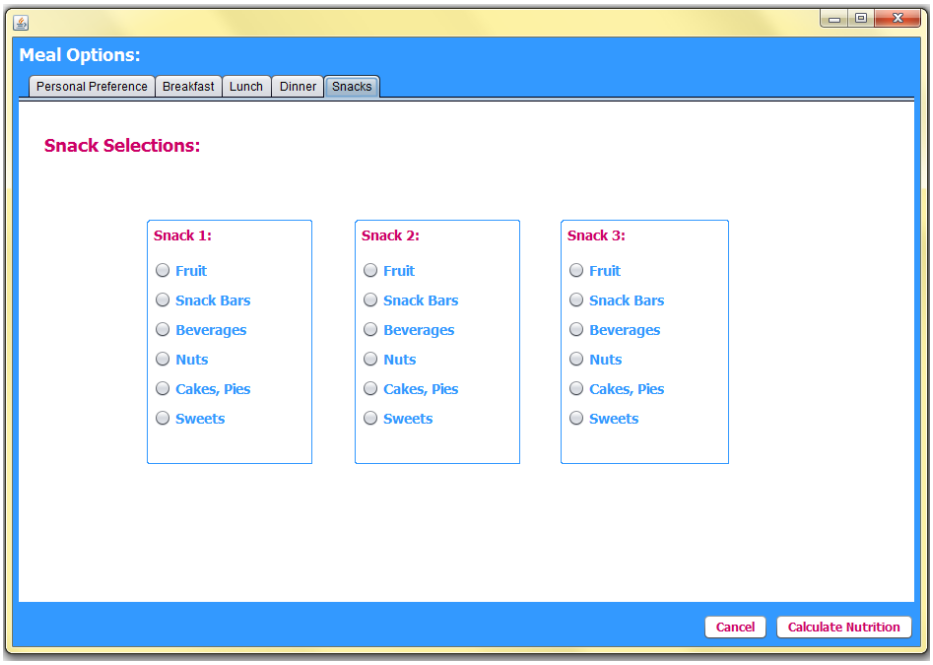


Εαν επιλέξετε " Nutrition Suggestion", θα πρέπει να κάνετε επιλογή τουλάχιστον από τα Protein Products και Carbohydrate Products.

5. . Συνεχίστε στην καρτέλα "Dinner".

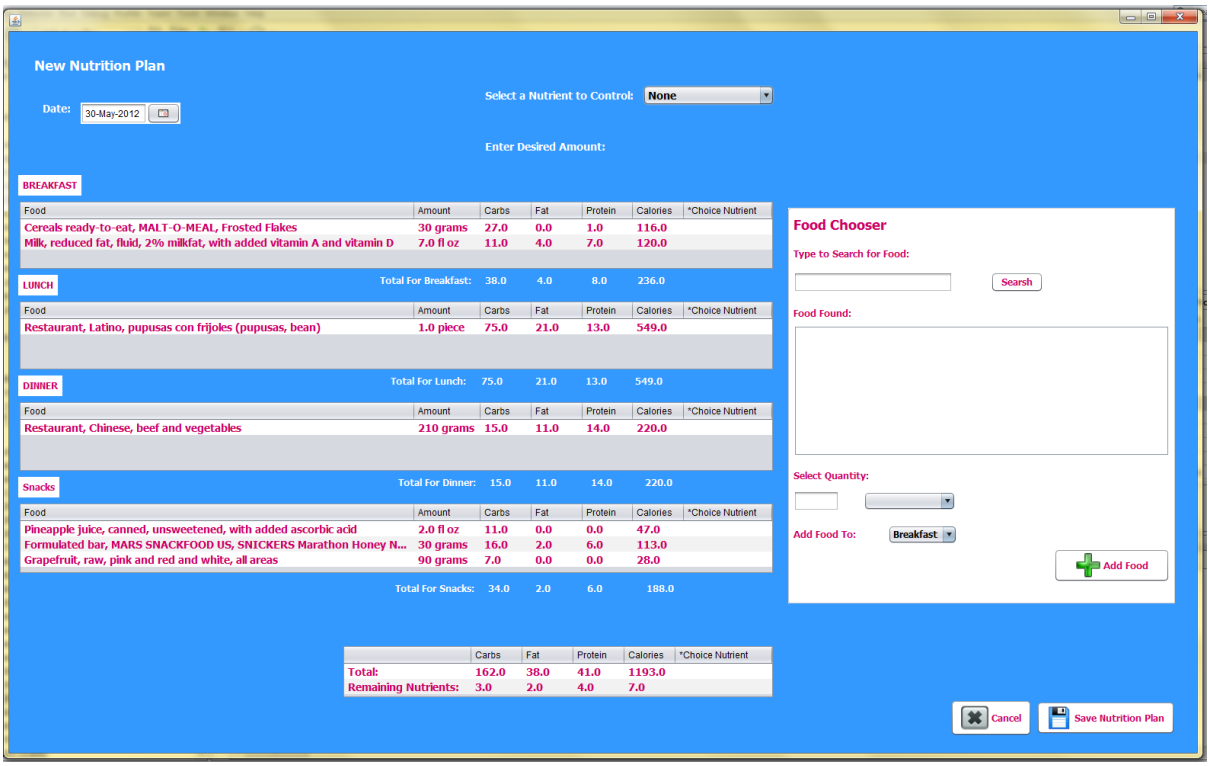
Οι επιλογές για το δείπνο είναι οι ίδιες με το μεσημεριανό με την επιπλέον επιλογή για δημητριακά στα "Carbohydrate Products".

6. Κάντε την επιλογή σας και συνεχίστε στην καρτέλα "Snacks".



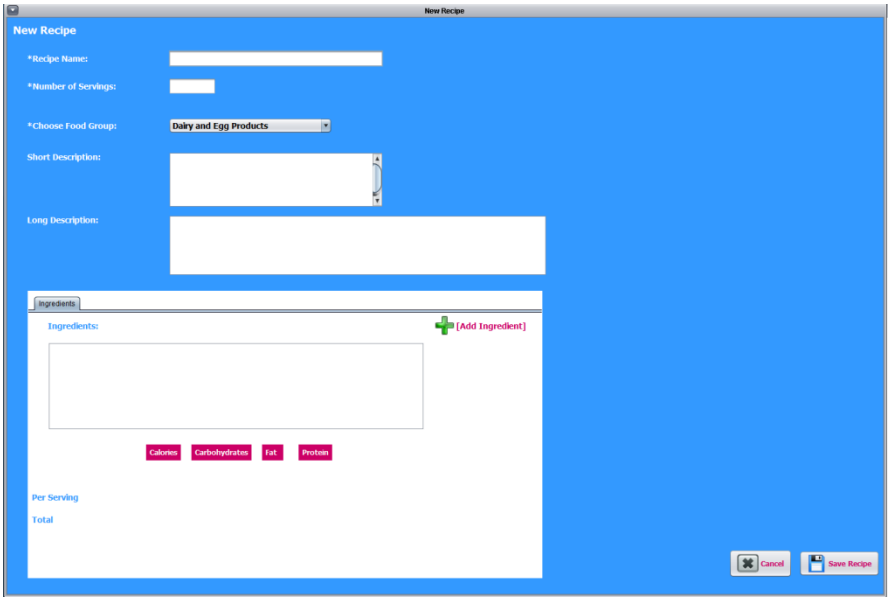
7. Έχει απομείνει να επιλέξετε τι θα θέλατε να έχετε για ενδιάμεσα. Κάνετε μια επιλογή και για τα τρία.

8. Επιλέξετε "Calculate Nutrition". Θα ανοίξει το παράθυρο με την διατροφή που έχει να προτείνει το σύστημα.



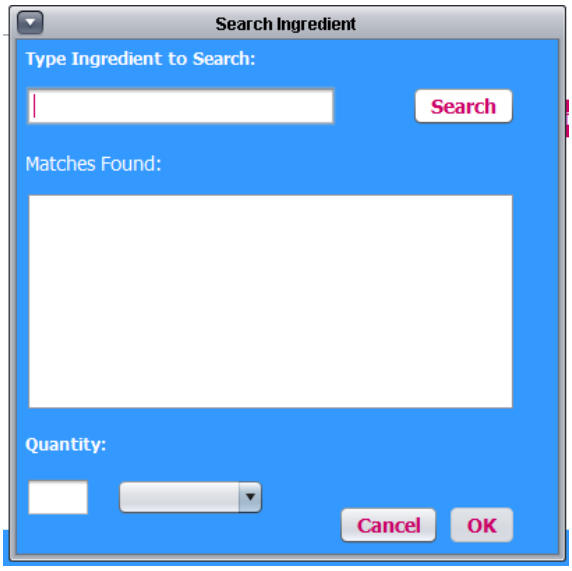
A.6 Εισαγωγή Νέου Φαγητού από Διάφορα Υλικά (Recipe)

1. Επιλέξτε "New Recipe" από το κυρίως παράθυρο. Θα ανοίξει το εξής παράθυρο στην οθόνη σας.



2. Εισάγεται το όνομα του φαγητού και τον αριθμό των μερίδων.

3. Επιλέξετε "Add Ingredient". Θα ανοίξει το παράθυρο για αναζήτηση φαγητού.

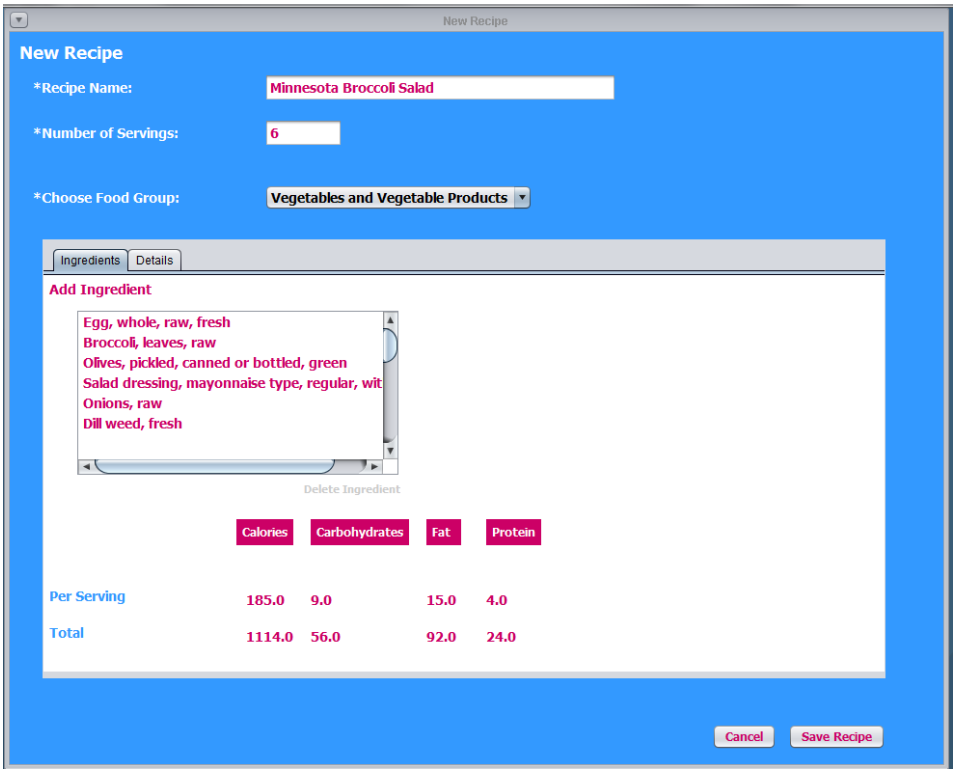


4. Εισάγετε το υλικό που θα θέλατε να ψάξετε για να προσθέσετε στο φαγητό και επιλέξτε "Search".

5. Επιλέξτε ένα φαγητό από την λίστα μια ποσότητα και στη συνέχεια "OK".

6. Το φαγητό προστέθηκε στην λίστα και υπολογίστηκαν γι'αυτό τα θρεπτικά συστατικά για όλες τις μερίδες και ανά μερίδα.

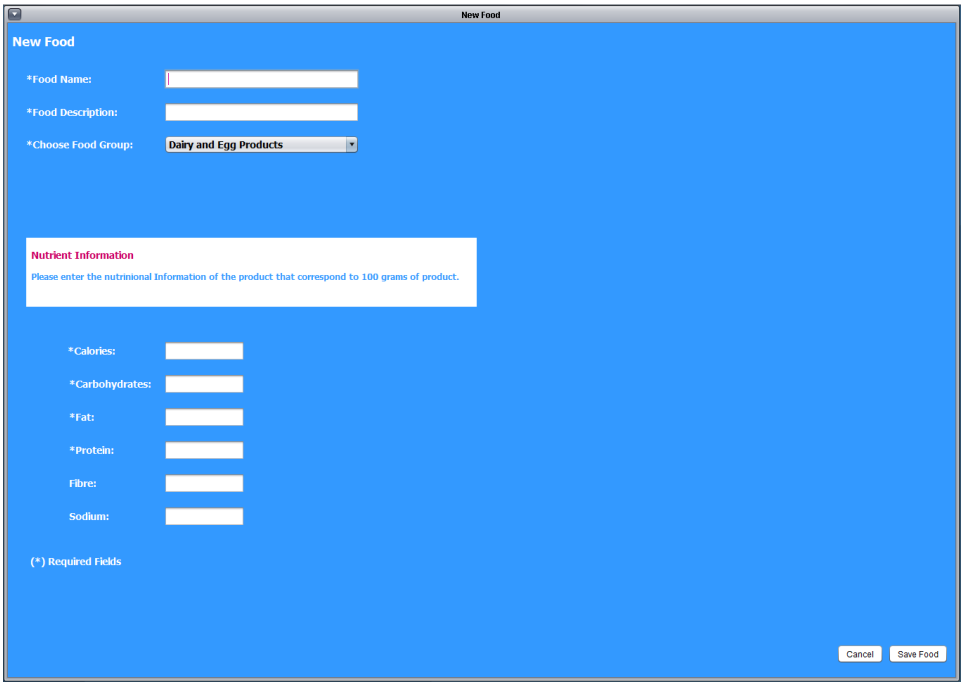
7. Επαναλάβετε τα βήματα 4-5 όσες φορές επιθυμείτε.



8. Επιλέξτε "Save Recipe", για να το αποθηκεύσετε.

Α.7 Εισαγωγή Νέου Φαγητού

1. Επιλέξετε "New Food" από το κυρίως παράθυρο.



2. Εισάγετε το όνομα και την περιγραφή του φαγητού.

3. Επιλέξετε σε ποιά ομάδα τροφίμων ανήκει.

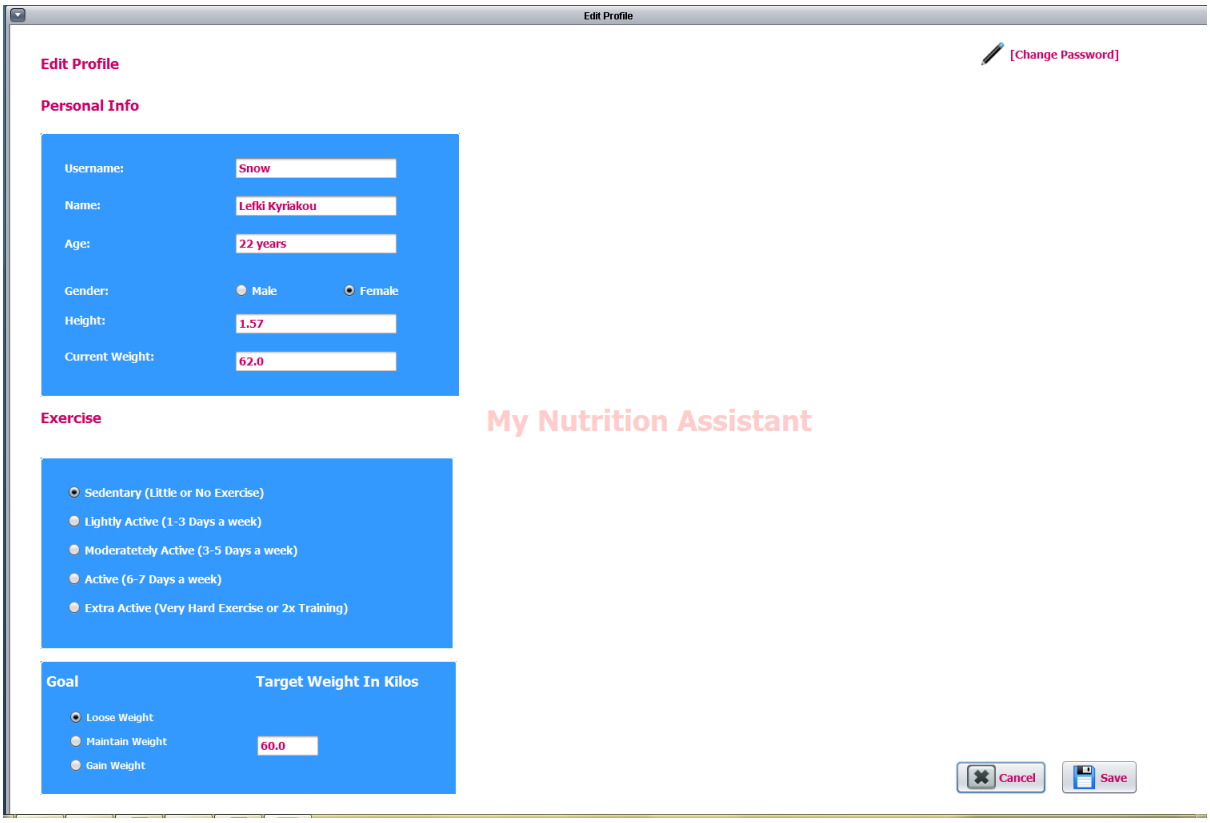
4. Εισάγετε τα στοιχεία του φαγητού όπως αυτά αναγράφονται στην ετικέτα της συσκευασίας του.

5. Επιλέξετε "Save Food".

A.8 Διόρθωση Στοιχείων Χρήστη

1. Από το κύριο παράθυρο επιλέξτε " User Profile". Εάν είναι απαραίτητο κάντε Log In.

2. Επιλέξετε "Edit Profile".



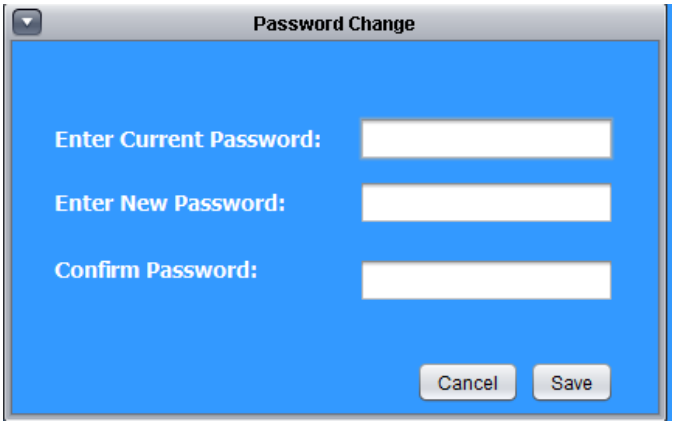
3. Αλλάξετε όποιο στοιχείο επιθυμείτε.

4. Επιλέξετε "Save".

A.9 Αλλαγή Password

1. Ανοίξετε το παράθυρο για διόρθωση των στοιχείων του χρήστη.

2. Επιλέξετε "Change Password".



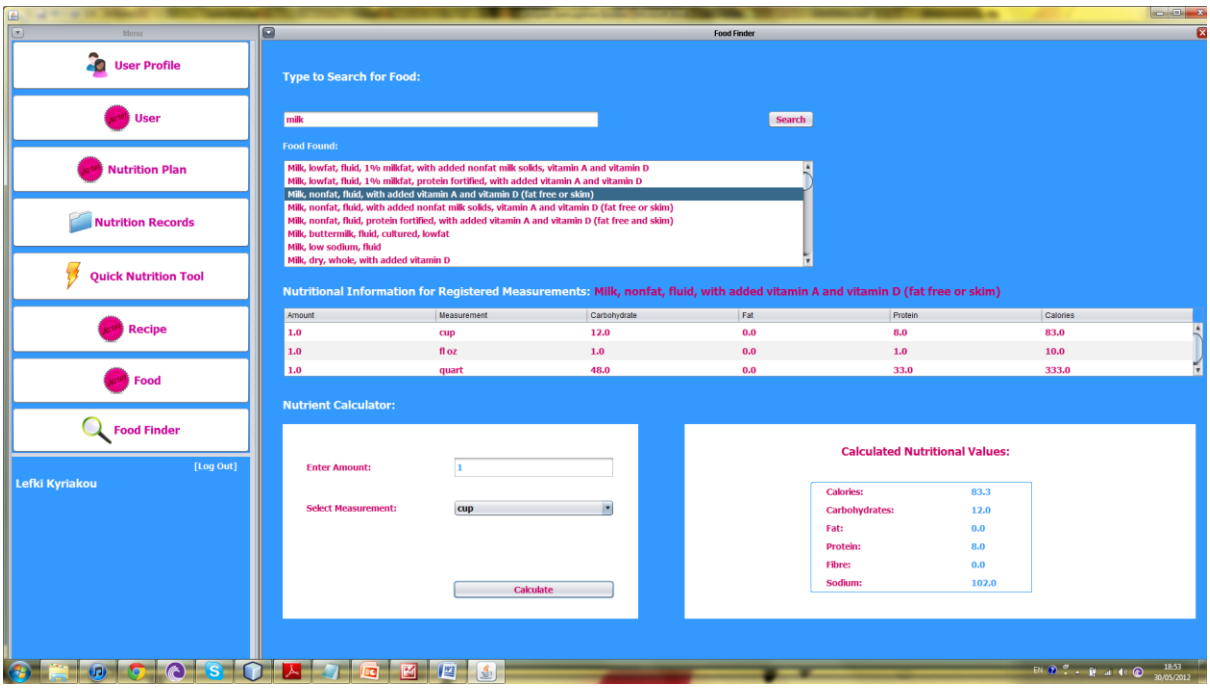
3. Εισάγετε το τρέχον σας Password και το νέο.

4. Επιλέξετε "Save".

Α.10 Προβολή Φαγητού και υπολογισμός θρεπτικών συστατικών επιλεγόμενης ποσότητας

1. Από το κεντρικό παράθυρο επιλέξετε "Food Finder".

2. Κάντε αναζήτηση οποίου φαγητού επιθυμείτε.



Με την επιλογή κάποιου φαγητού από τα αποτελέσματα της αναζήτησης εμφανίζονται όλες οι καταχωρημένες μετρήσεις για το επιλεγμένο φαγητό και τα αντίστοιχα θρεπτικά του συστατικά.

Επίσης με το "Nutrient Calculator" μπορεί να υπολογιστούν τα θρεπτικά συστατικά και η ενέργεια συγκεκριμένης ποσότητας φαγητού.

Παράρτημα Β

Εδώ θα βρείτε τα διάφορα queries που χρησιμοποιήθηκαν για την διαχείρηση των δεδομένων της βάσης.

Β.1 Queries για ανάκτηση των Nutrients για κάθε ομάδα τροφίμων:

Β.1.1 BREAKFAST

Β.1.1.1 Eggs:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 0100 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού υπάρχουν οι λέξεις " egg ", " cooked " και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select DISTINCT *

from nutrient_data.food_description A,  nutrient_data.nutrient_data B

where A.fdgrp_cd=0100

and (A.long_desc like '%egg%' and A.long_desc like '%cooked%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303
```

```
or B.nutr_no=304 );
```

B.1.1.2 Dairy Products:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 0100 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: cream, whey, topping, raw, sause, fondue, sour, protein, buttermilk, oil, imitation, dulce de leche, ensure, whipped και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select DISTINCT *

from nutrient_data.food_description A,  nutrient_data.nutrient_data B

where A.fdgrp_cd=0100

and !(A.long_desc like '%cream%')

and !(A.long_desc like '%whey%')

and !(A.long_desc like '%topping%')

and !(A.long_desc like '%raw%')

and !(A.long_desc like '%sause%')

and !(A.long_desc like '%fondue%')

and !(A.long_desc like '%sour%')

and !(A.long_desc like '%protein%')

and !(A.long_desc like '%buttermilk%')

and !(A.long_desc like '%oil%')

and !(A.long_desc like '%imitation%')

and !(A.long_desc like '%dulce de leche%')

and !(A.long_desc like '%ensure%')

and !(A.long_desc like '%whipped%')
```

```
and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.1.3 Cheese Products:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 0100 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: cream, whey, topping, raw, sause, fondue, sour, protein, buttermilk, oil, imitation, dulce de leche, ensure, whipped αλλά περιέχουν την λέξη cheese και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select DISTINCT *

from nutrient_data.food_description A,  nutrient_data.nutrient_data B

where A.fdgrp_cd=0100

and (!(A.long_desc like '%cream%')

and !(A.long_desc like '%whey%')

and !(A.long_desc like '%topping%')
```

```
and !(A.long_desc like '%raw%')

and !(A.long_desc like '%sause%')

and !(A.long_desc like '%fondue%')

and !(A.long_desc like '%sour%')

and !(A.long_desc like '%protein%')

and !(A.long_desc like '%buttermilk%')

and !(A.long_desc like '%oil%')

and !(A.long_desc like '%imitation%')

and !(A.long_desc like '%dulce de leche%')

and !(A.long_desc like '%ensure%')

and !(A.long_desc like '%whipped%')

and A.long_desc like 'cheese%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303
```

```
or B.nutr_no=304 );
```

B.1.1.4 Breakfast Cereal:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 0800 από τους πίνακες food_description και nutrient_data και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select distinct *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=0800

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.1.5 Fruits and Fruit Juices:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 0900 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: cooked, frozen, canned μαζί με την λέξη juice, sour, dehydrated, peel, juice pack και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *
from nutrient_data.food_description A, nutrient_data.nutrient_data B
where fdgrp_cd=0900
and !(A.long_desc like '%cooked%')
and !(A.long_desc like '%frozen%')
and (!(A.long_desc like '%canned%') or A.long_desc like '%juice%')
and !(A.long_desc like '%sour%')
and !(A.long_desc like '%dehydrated%')
and !(A.long_desc like '%stewed%')
and !(A.long_desc like '%peel%')
and !(A.long_desc like '%juice pack%')
and A.ndb_no=B.ndb_no
and (B.nutr_no=208
or  B.nutr_no=205
or B.nutr_no=204
or B.nutr_no=203
or B.nutr_no=307
or B.nutr_no=306
or B.nutr_no=320
```

```
or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.1.6 Baked Products:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 1800 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: crumbs, stuffing, croutons, ice cream, crust, phyllo, taco, tortillas, leavening, flour, bread και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A,  nutrient_data.nutrient_data B

where A.fdgrp_cd=1800

and !(A.long_desc like '%crumbs%')

and !(A.long_desc like '%stuffing%')

and !(A.long_desc like '%croutons%')

and !(A.long_desc like '%ice cream%')

and !(A.long_desc like '%crust%')

and !(A.long_desc like '%phyllo%')

and !(A.long_desc like '%taco%')

and !(A.long_desc like '%tortillas%')

and !(A.long_desc like '%leavening%')

and !(A.long_desc like '%flour%')

and !(A.long_desc like 'bread%')
```

```
and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.1.7 Bread:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 1800 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: crumbs, stuffing, croutons, ice cream, crust, phyllo, taco, tortillas, leavening, flour αλλά περιέχεται η λέξη: bread και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=1800

and (!(A.long_desc like '%crumbs%')

and !(A.long_desc like '%stuffing%')

and !(A.long_desc like '%croutons%')
```

```
and !(A.long_desc like '%ice cream%')

and !(A.long_desc like '%crust%')

and !(A.long_desc like '%phyllo%')

and !(A.long_desc like '%taco%')

and !(A.long_desc like '%tortillas%')

and !(A.long_desc like '%leavening%')

and !(A.long_desc like '%flour%')

and A.long_desc like 'bread%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.1.8 Sausages and Luncheon Meats:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 0700 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού υπάρχουν οι λέξεις: sausage ή salami και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=0700

and (A.long_desc like 'sausage%' or A.long_desc like 'salami%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2 LUNCH & DINNNER

B.1.2.1 Poultry Products:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 0500 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: uncooked , raw, USDA αλλά υπάρχει η λέξη chicken και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=0500

and (!(A.long_desc like '%uncooked%')

and !(A.long_desc like '%raw%')

and !(A.long_desc like '%USDA%')

and A.long_desc like 'chicken%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2.2 Pork Products:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 1000 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: uncooked , raw, USDA, HORMEL, unheated, pickled, dehydrated και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=1000

and !(A.long_desc like '%uncooked%')

and !(A.long_desc like '%raw%')

and !(A.long_desc like '%USDA%')

and !(A.long_desc like '%HORMEL%')

and !(A.long_desc like '%unheated%')

and !(A.long_desc like '%pickled%')

and !(A.long_desc like '%dehydrated%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401
```

```
or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2.3 Vegetables and Vegetable Products:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 1100 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: unprepared, seeds, pods, garlic, ginger, potato με την λέξη raw, corn, USDA, flour, poi, powder, juice, yeast, sauce και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where fdgrp_cd=1100

and !(A.long_desc like '%unprepared%')

and !(A.long_desc like '%seeds%')

and !(A.long_desc like '%pods%')

and !(A.long_desc like '%garlic%')

and !(A.long_desc like '%ginger%')

and (!(A.long_desc like '%potato%') or !(A.long_desc like '%raw%'))

and !(A.long_desc like '%corn%')

and !(A.long_desc like '%USDA%')

and !(A.long_desc like '%flour%')

and (A.long_desc!='poi')

and !(A.long_desc like '%powder%')

and !(A.long_desc like '%juice%')
```

```
and !(A.long_desc like '%yeast%')

and !(A.long_desc like '%sauce%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2.4 Corn & Potato:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 1100 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού υπάρχουν οι λέξεις: potato ή corn χωρίς τις λέξεις: raw, unprepared, USDA, flour, poi, powder, juice, yeast, sauce, pancakes και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where fdgrp_cd=1100

and (((A.long_desc like 'potato%') or (A.long_desc like 'corn%'))and !(A.long_desc like '%raw%')
```

```
and !(A.long_desc like '%USDA%')

and !(A.long_desc like '%flour%')

and (A.long_desc!='poi')

and !(A.long_desc like '%unprepared%')

and !(A.long_desc like '%canned%')

and !(A.long_desc like '%pancakes%')

and !(A.long_desc like '%powder%')

and !(A.long_desc like '%juice%')

and !(A.long_desc like '%yeast%')

and !(A.long_desc like '%sauce%'))

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2.5 Beef Products:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 1300 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: uncooked, raw, unprepared, USDA, HORMEL, unheated, pickled, dehydrated, calculator και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=1300

and !(A.long_desc like '%uncooked%')

and !(A.long_desc like '%raw%')

and !(A.long_desc like '%USDA%')

and !(A.long_desc like '%HORMEL%')

and !(A.long_desc like '%unheated%')

and !(A.long_desc like '%pickled%')

and !(A.long_desc like '%dehydrated%')

and !(A.long_desc like '%calculator%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320
```

```
or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2.6 Finfish & Shellfish Products:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 1500 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: uncooked, raw, unprepared, USDA, HORMEL, unheated, pickled, dehydrated και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=1500

and !(A.long_desc like '%uncooked%')

and !(A.long_desc like '%raw%')

and !(A.long_desc like '%USDA%')

and !(A.long_desc like '%HORMEL%')

and !(A.long_desc like '%unheated%')

and !(A.long_desc like '%pickled%')

and !(A.long_desc like '%dehydrated%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204
```

```
or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2.7 Legumes & Legume Products:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 1600 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: raw, canned, flour, peanut, sauce, soymilk, protein, unprepared και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=1600

and !(A.long_desc like '%raw%')

and !(A.long_desc like '%canned%')

and !(A.long_desc like '%flour%')

and !(A.long_desc like '%peanut%')

and !(A.long_desc like '%sauce%')

and !(A.long_desc like '%soymilk%')

and !(A.long_desc like '%protein%')
```

```
and !(A.long_desc like '%unprepared%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2.8 Lamb, Veal and Game Products:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 1700 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: raw, canned, sauce, unprepared αλλά περιέχουν την λέξη lamb και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=1700

and (!(A.long_desc like '%raw%')

and !(A.long_desc like '%canned%')
```

```
and !(A.long_desc like '%sauce%')

and !(A.long_desc like '%unprepared%')

and A.long_desc like 'lamb%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2.9 Cereal Grains & Pasta:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 2000 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: raw, canned, unprepared, uncooked, flour, hulled, oat, buckwheat, dry, amaranth, cornmeal, wheat αλλά περιέχουν την λέξη lamb και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=2000
```

```
and !(A.long_desc like '%raw%')

and !(A.long_desc like '%canned%')

and !(A.long_desc like '%unprepared%')

and !(A.long_desc like '%uncooked%')

and !(A.long_desc like '%flour%')

and !(A.long_desc like '%hulled%')

and !(A.long_desc like '%oat%')

and !(A.long_desc like '%buckwheat%')

and !(A.long_desc like '%cornstarch%')

and !(A.long_desc like '%dry%')

and !(A.long_desc like '%amaranth%')

and !(A.long_desc like '%cornmeal%')

and !(A.long_desc like '%wheat%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301
```

```
or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2.10 Restaurant Foods:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 3600 από τους πίνακες food_description και nutrient_data και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=3600

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2.11 Greek Cuisine:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 4000 από τους πίνακες food_description και nutrient_data και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A,  nutrient_data.nutrient_data B

where A.fdgrp_cd=4000

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2.12 Chinese Cuisine:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 5000 από τους πίνακες food_description και nutrient_data και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *
```

```
from nutrient_data.food_description A,  nutrient_data.nutrient_data B

where A.fdgrp_cd=5000

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2.13 Italian Cuisine:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 6000 από τους πίνακες food_description και nutrient_data και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A,  nutrient_data.nutrient_data B

where A.fdgrp_cd=6000

and A.ndb_no=B.ndb_no

and (B.nutr_no=208
```

```
or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2.14 French Cuisine:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 7000 από τους πίνακες food_description και nutrient_data και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A,  nutrient_data.nutrient_data B

where A.fdgrp_cd=7000

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307
```

```
or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.2.15 Japanese Cuisine:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 8000 από τους πίνακες food_description και nutrient_data και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A,  nutrient_data.nutrient_data B

where A.fdgrp_cd=8000

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301
```

```
or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.3 SNACKS

B.1.3.1 Fruits & Fruit Juices:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 0900 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: cooked, frozen, canned μαζί με την λέξη juice, sour, dehydrated, stewed, peel, juice pack, dry, amaranth, cornmeal, wheat αλλά περιέχουν την λέξη lamb και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description

where fdgrp_cd=0900

and !(long_desc like '%cooked%')

and !(long_desc like '%frozen%')

and (!(long_desc like '%canned%') or long_desc like '%juice%')

and !(long_desc like '%sour%')

and !(long_desc like '%dehydrated%')

and !(long_desc like '%stewed%')

and !(long_desc like '%peel%')

and !(long_desc like '%juice pack%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204
```

```
or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.3.2 Nut & Seed Products:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 1200 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχει η λέξη flour αλλά περιέχεται η λέξη nuts και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=1200

and A.long_desc like '%nuts%'

and !(A.long_desc like '%flour%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307
```

```
or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.3.3 Beverages:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 1400 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχει η λέξη beef αλλά περιέχεται η λέξη juice και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *
from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=1400

and !(A.long_desc like '%beef%')

and A.long_desc like '%juice%'

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320
```

```
or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.3.4 Baked Products:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 1800 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: crumbs, stuffing, croutons, ice cream, crust, phyllo, taco, tortillas, leavening, flour, bread και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description

where fdgrp_cd=1800

and !(long_desc like '%crumbs%')

and !(long_desc like '%stuffing%')

and !(long_desc like '%croutons%')

and !(long_desc like '%ice cream%')

and !(long_desc like '%crust%')

and !(long_desc like '%phyllo%')

and !(long_desc like '%taco%')

and !(long_desc like '%tortillas%')

and !(long_desc like '%leavening%')

and !(long_desc like '%flour%');
```

B.1.3.5 Sweets:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 1900 από τους πίνακες food_description και nutrient_data, όπου, στην περιγραφή του φαγητού δεν υπάρχουν οι λέξεις: gum, baking, coating, syrups, powder, dry mix μαζί με την λέξη prepared, frostings, sugar, fillings, toppings, sweeteners, pectin και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=1900

and !(A.long_desc like '%gum%')

and !(A.long_desc like '%baking%')

and !(A.long_desc like '%coating%')

and !(A.long_desc like '%syrups%')

and !(A.long_desc like '%powder%')

and (!(A.long_desc like '%dry mix%') or A.long_desc like '%prepared%')

and !(A.long_desc like '%frostings%')

and !(A.long_desc like '%sugar%')

and !(A.long_desc like '%fillings%')

and !(A.long_desc like '%toppings%')

and !(A.long_desc like '%sweeteners%')

and !(A.long_desc like '%pectin%')

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204
```

```
or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320

or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.1.3.6 Snack Bars:

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 2500 από τους πίνακες food_description και nutrient_data και που έχουν συγκεκριμένο κωδικό θρεπτικών συστατικών. (Υδατάνθρακες, Πρωτείνες, Λίπη, κλπ.).

```
select *

from nutrient_data.food_description A, nutrient_data.nutrient_data B

where A.fdgrp_cd=2500

and A.ndb_no=B.ndb_no

and (B.nutr_no=208

or  B.nutr_no=205

or B.nutr_no=204

or B.nutr_no=203

or B.nutr_no=307

or B.nutr_no=306

or B.nutr_no=320
```

```
or B.nutr_no=401

or B.nutr_no=301

or B.nutr_no=303

or B.nutr_no=304 );
```

B.2 Queries για ανάκτηση των Nutrients για κάθε ομάδα τροφίμων:

Με πάρόμοιο τρόπο διατυπώθηκαν τα queries για ανάκτηση των πληροφοριών για τις μετρήσεις από την βάση δεδομένων.

Ένα παράδειγμα είναι:

B.2.1 Eggs

Επιλογή των εγγραφών με αριθμό ομάδας τροφίμων 0100 από τους πίνακες food_description και weight_file και περιέχουν τις λέξεις egg και cooked.

```
select DISTINCT *

from nutrient_data.food_description A,  nutrient_data.weight_file B

where A.fdgrp_cd=0100

and (A.long_desc like '%egg%' and A.long_desc like '%cooked%')

and A.ndb_no=B.ndb_no;
```

B.3 Query για αποθήκευση του χρήστη:

```
insert into `nutrient_data`.`user_details` (`Username`, `Password`, `Name`, `Surname`, `Age`, `Gender`, `Weight`, `Weight_unit`, `Height`, `Height_unit`, `Target_Weight`, `Target_unit`, `Goal`, `BMR`, `Suggested_Calories`, `Exercise`) values(username , password, name ,surname ,age,gender, weight, w_unit, height, h_unit,Target_W,  target_unit,Goal ,BMR,tempcal, User_Exercise);
```

B.4 Query για διόρθωση του χρήστη:

```
UPDATE nutrient_data.user_details SET Username='jTextField25.getText()',
```

```
Name='Name[0]',

Surname='Name[1]',

Age=a[0],

Gender='g',

Weight=jTextField27.getText(),

Height=jTextField26.getText(),

Target_Weight=jTextField28.getText(),

Goal='goal',

BMR=bmr,

Suggested_Calories=sug_cal,

Exercise='User_Exercise'

WHERE id_user=ID;
```

B.5 Queries για διαγραφή του χρήστη:

```
DELETE FROM nutrient_data.user_details WHERE id_user=ID;
```

B.6 Queries για αποθήκευση μιας διατροφής:

```
insert into `nutrient_data`.`nutrition_plans` ( `user_id`, `Nutrition_Date`, `date_created`)

values(ID ,  Nutrition_Date , NOW());
```

B.7 Queries για αποθήκευση ενός φαγητού διατροφής:

```
insert into `nutrient_data`.`nutrition_plans_food` ( `Nutrition_Plans_id`,`Food`, `Amount`, `Carbohydrate`, `Fat`, `Protein`, `Calories`, `Meal`)
values((select idnutrition_plans from nutrient_data.nutrition_plans where date_created=(
```

select max(date_created) from nutrient_data.nutrition_plans where user_id=ID)),Food , Amount , Carbs, Fat , Protein, Calories , Meal);

B.8 Queries για διόρθωση μιας διατροφής:

```
UPDATE nutrient_data.nutrition_plans_food
        SET Food=Food ,
        Amount=Amount ,
         Carbohydrate=Carbs,
        Fat=Fat,
        Protein=Protein,
        Calories=Calories,
        Prefered=PreferedValue
        WHERE id=Food_Id;
```

B.9 Queries για διόρθωση ενός φαγητού διατροφής:

B.10 Queries για διαγραφή μιας διατροφής:

```
DELETE FROM nutrient_data.nutrition_plans
WHERE idnutrition_plans=idnutritions[Nutritionrow];
```

B.11 Queries για διαγραφή ενός φαγητού διατροφής:

```
DELETE FROM nutrient_data.nutrition_plans_food
WHERE Nutrition_Plans_id=idnutritions[Nutritionrow];
```

B.12 Queries για αποθήκευση ενός φαγητού:

```
insert into `nutrient_data`.`food_description` (`ndb_no`, `fdgrp_cd`, `long_desc`, `shrt_desc`)

values(fcd , fdg_no , FoodName_long,FoodName_short );


insert into `nutrient_data`.`nutrient_data`(NDB_No, Nutr_No, Nutr_Val, Num_Data_Pts, Src_Cd)

values(fcd, 208,cal,0,91),(fcd ,205,carb,0, 91),

 (fcd,204, fat,0,91),(fcd ,203, prot,0,91),

( fcd,291, fibre,0,91),( fcd ,307,sod,0,91);
```

B.13 Queries για αλλαγή ενός password:

```
UPDATE nutrient_data.user_details  SET Password='newPass1' WHERE id_user=ID;
```

Παράρτημα Γ

Παράδειγμα Αρχείου Προγράμματος και Αρχείου Δεδομένων

Πιο κάτω ακολουθεί ένα παράδειγμα των παραγόμενων αρχείων του συστήματος. Αρχικά το αρχείο του εκτελέσιμου προγράμματος ικανοποίησης περιορισμών και στη συνέχεια το αρχείο δεδομένων που χρησιμοποιεί το πρόγραμμα.

Γ.1 Αρχείο Προγράμματος (.mzn)

```
int: nutrCount=11;
int: B1Count;
int: minB1;
int: maxB1;
set of int: B1Items= 1..B1Count;
array[B1Items] of int: B1Calories;
array[B1Items] of int: B1Carbohydrate;
array[B1Items] of int: B1Fat;
array[B1Items] of int: B1Protein;
array[B1Items] of int: B1Sodium;
array[B1Items] of int: B1Potassium;
array[B1Items] of int: B1VitaminA;
array[B1Items] of int: B1VitaminC;
array[B1Items] of int: B1Calcium;
array[B1Items] of int: B1Iron;
array[B1Items] of int: B1Magnesium;

var minB1..maxB1: AmountB1;

var 1..B1Count: b1Amount;
constraint (B1Calories[b1Amount]!=0);

int: B2Count;
int: minB2;
int: maxB2;
set of int: B2Items= 1..B2Count;
```

```
array[B2Items] of int: B2Calories;
array[B2Items] of int: B2Carbohydrate;
array[B2Items] of int: B2Fat;
array[B2Items] of int: B2Protein;
array[B2Items] of int: B2Sodium;
array[B2Items] of int: B2Potassium;
array[B2Items] of int: B2VitaminA;
array[B2Items] of int: B2VitaminC;
array[B2Items] of int: B2Calcium;
array[B2Items] of int: B2Iron;
array[B2Items] of int: B2Magnesium;

var minB2..maxB2: AmountB2;

var 1..B2Count: b2Amount;
constraint (B2Calories[b2Amount]!=0);

int: L1Count;
int: minL1;
int: maxL1;
set of int: L1Items= 1..L1Count;
array[L1Items] of int: L1Calories;
array[L1Items] of int: L1Carbohydrate;
array[L1Items] of int: L1Fat;
array[L1Items] of int: L1Protein;
array[L1Items] of int: L1Sodium;
array[L1Items] of int: L1Potassium;
array[L1Items] of int: L1VitaminA;
array[L1Items] of int: L1VitaminC;
array[L1Items] of int: L1Calcium;
array[L1Items] of int: L1Iron;
array[L1Items] of int: L1Magnesium;

var minL1..maxL1: AmountL1;

var 1..L1Count: l1Amount;
constraint (L1Calories[l1Amount]!=0);
```

```
int: D1Count;
int: minD1;
int: maxD1;
set of int: D1Items= 1..D1Count;
array[D1Items] of int: D1Calories;
array[D1Items] of int: D1Carbohydrate;
array[D1Items] of int: D1Fat;
array[D1Items] of int: D1Protein;
array[D1Items] of int: D1Sodium;
array[D1Items] of int: D1Potassium;
array[D1Items] of int: D1VitaminA;
array[D1Items] of int: D1VitaminC;
array[D1Items] of int: D1Calcium;
array[D1Items] of int: D1Iron;
array[D1Items] of int: D1Magnesium;

var minD1..maxD1: AmountD1;

var 1..D1Count: d1Amount;
constraint (D1Calories[d1Amount]!=0);

int: S1Count;
int: minS1;
int: maxS1;
set of int: S1Items= 1..S1Count;
array[S1Items] of int: S1Calories;
array[S1Items] of int: S1Carbohydrate;
array[S1Items] of int: S1Fat;
array[S1Items] of int: S1Protein;
array[S1Items] of int: S1Sodium;
array[S1Items] of int: S1Potassium;
array[S1Items] of int: S1VitaminA;
array[S1Items] of int: S1VitaminC;
array[S1Items] of int: S1Calcium;
array[S1Items] of int: S1Iron;
array[S1Items] of int: S1Magnesium;
```

```
var minS1..maxS1: AmountS1;

var 1..S1Count: s1Amount;
constraint (S1Calories[s1Amount]!=0);

int: S2Count;
int: minS2;
int: maxS2;
set of int: S2Items= 1..S2Count;
array[S2Items] of int: S2Calories;
array[S2Items] of int: S2Carbohydrate;
array[S2Items] of int: S2Fat;
array[S2Items] of int: S2Protein;
array[S2Items] of int: S2Sodium;
array[S2Items] of int: S2Potassium;
array[S2Items] of int: S2VitaminA;
array[S2Items] of int: S2VitaminC;
array[S2Items] of int: S2Calcium;
array[S2Items] of int: S2Iron;
array[S2Items] of int: S2Magnesium;

var minS2..maxS2: AmountS2;

var 1..S2Count: s2Amount;
constraint (S2Calories[s2Amount]!=0);

int: S3Count;
int: minS3;
int: maxS3;
set of int: S3Items= 1..S3Count;
array[S3Items] of int: S3Calories;
array[S3Items] of int: S3Carbohydrate;
array[S3Items] of int: S3Fat;
array[S3Items] of int: S3Protein;
array[S3Items] of int: S3Sodium;
array[S3Items] of int: S3Potassium;
```

```
array[S3Items] of int: S3VitaminA;
array[S3Items] of int: S3VitaminC;
array[S3Items] of int: S3Calcium;
array[S3Items] of int: S3Iron;
array[S3Items] of int: S3Magnesium;

var minS3..maxS3: AmountS3;

var 1..S3Count: s3Amount;
constraint (S3Calories[s3Amount]!=0);%%%%
set of int: nutrItems=1..nutrCount;
array[nutrItems] of int: Nutrients;
array[nutrItems] of var int: temp;

constraint                                                    temp[1]=
(B1Calories[b1Amount]*AmountB1*15)+(B2Calories[b2Amount]*AmountB2*15
)+(L1Calories[l1Amount]*AmountL1*15)+(D1Calories[d1Amount]*AmountD1*
15)+(S1Calories[s1Amount]*AmountS1*15)+(S2Calories[s2Amount]*AmountS
2*15)+(S3Calories[s3Amount]*AmountS3*15);
constraint temp[1] < Nutrients[1];
constraint                                                    temp[2]=
(B1Carbohydrate[b1Amount]*AmountB1*15)+(B2Carbohydrate[b2Amount]*Amo
untB2*15)+(L1Carbohydrate[l1Amount]*AmountL1*15)+(D1Carbohydrate[d1A
mount]*AmountD1*15)+(S1Carbohydrate[s1Amount]*AmountS1*15)+(S2Carboh
ydrate[s2Amount]*AmountS2*15)+(S3Carbohydrate[s3Amount]*AmountS3*15)
;
constraint temp[2] < Nutrients[2]::domain;
constraint                                                    temp[3]=
(B1Fat[b1Amount]*AmountB1*15)+(B2Fat[b2Amount]*AmountB2*15)+(L1Fat[l
1Amount]*AmountL1*15)+(D1Fat[d1Amount]*AmountD1*15)+(S1Fat[s1Amount]
*AmountS1*15)+(S2Fat[s2Amount]*AmountS2*15)+(S3Fat[s3Amount]*AmountS
3*15);
constraint temp[3] < Nutrients[3]::domain;
constraint                                                    temp[4]=
(B1Protein[b1Amount]*AmountB1*15)+(B2Protein[b2Amount]*AmountB2*15)+
(L1Protein[l1Amount]*AmountL1*15)+(D1Protein[d1Amount]*AmountD1*15)+
```

```
(S1Protein[s1Amount]*AmountS1*15)+(S2Protein[s2Amount]*AmountS2*15)+
(S3Protein[s3Amount]*AmountS3*15);
constraint temp[4] < Nutrients[4]::domain;

solve maximize temp[1];
output ["B1 Amount: ", show(AmountB1), " in pos: ",show(b1Amount),
"\n"
++"B2 Amount: ", show(AmountB2), " in pos: ",show(b2Amount), "\n"
++"L1 Amount: ", show(AmountL1), " in pos: ",show(l1Amount), "\n"
++"D1 Amount: ", show(AmountD1), " in pos: ",show(d1Amount), "\n"
++"S1 Amount: ", show(AmountS1), " in pos: ",show(s1Amount), "\n"
++"S2 Amount: ", show(AmountS2), " in pos: ",show(s2Amount), "\n"
++"S3 Amount: ", show(AmountS3), " in pos: ",show(s3Amount), "\n"];
```

Γ.2 Αρχείο Δεδομένων (.dzn)

```
B1Count = 5;
minB1=1;
maxB1=3;
B1Calories=[530,3900,3870,3980,3720];
B1Carbohydrate=[109,815,815,909,770];
B1Fat=[3,21,52,13,47];
B1Protein=[18,96,80,55,85];
B1Sodium=[180,7000,2190,930,5380];
B1Potassium=[230,3090,2490,1250,2480];
B1VitaminA=[0,9410,4600,8340,6120];
B1VitaminC=[0,306,20,0,0];
B1Calcium=[970,3970,390,150,2040];
B1Iron=[53,170,37,66,73];
B1Magnesium=[70,1020,760,610,870];

B2Count = 5;
minB2=8;
maxB2=10;
B2Calories=[370,630,510,4960,780];
B2Carbohydrate=[50,47,49,384,121];
B2Fat=[2,34,19,267,19];
```

```
B2Protein=[35,33,34,263,29];
B2Sodium=[530,570,520,3710,660];
B2Potassium=[1710,1390,1620,13300,1230];
B2VitaminA=[640,20,560,2580,640];
B2VitaminC=[10,9,10,86,0];
B2Calcium=[1290,1280,1280,9120,1940];
B2Iron=[0,0,0,4,2];
B2Magnesium=[150,130,140,850,140];

L1Count = 5;
minL1=7;
maxL1=10;
L1Calories=[2320,1510,2910,2920,3290];
L1Carbohydrate=[230,244,350,366,248];
L1Fat=[104,38,151,145,189];
L1Protein=[115,46,34,36,148];
L1Sodium=[4260,4200,460,3140,7940];
L1Potassium=[2550,2240,6200,6100,1150];
L1VitaminA=[80,0,0,0,890];
L1VitaminC=[0,0,0,0,0];
L1Calcium=[490,370,190,190,3480];
L1Iron=[10,15,10,9,6];
L1Magnesium=[380,260,330,330,220];

D1Count = 5;
minD1=9;
maxD1=16;
D1Calories=[1510,1740,1860,1050,1860];
D1Carbohydrate=[244,157,266,72,0];
D1Fat=[38,90,72,53,78];
D1Protein=[46,73,34,70,287];
D1Sodium=[4200,4730,2770,4090,3070];
D1Potassium=[2240,1520,1860,2040,3630];
D1VitaminA=[0,120,20,630,0];
D1VitaminC=[0,0,0,115,0];
D1Calcium=[370,750,240,220,90];
D1Iron=[15,8,5,11,28];
```

```
D1Magnesium=[260,260,290,150,250];

S1Count = 5;
minS1=3;
maxS1=6;
S1Calories=[2870,480,530,320,680];
S1Carbohydrate=[736,111,128,76,143];
S1Fat=[11,3,1,3,9];
S1Protein=[37,13,3,4,25];
S1Sodium=[90,10,20,70,20];
S1Potassium=[5310,2590,1300,1460,4170];
S1VitaminA=[0,960,0,380,310];
S1VitaminC=[130,100,100,16776,2283];
S1Calcium=[790,130,130,120,180];
S1Iron=[18,3,3,2,2];
S1Magnesium=[370,100,120,180,220];

S2Count = 5;
minS2=1;
maxS2=3;
S2Calories=[4710,4150,5150,5310,3420];
S2Carbohydrate=[644,730,692,638,827];
S2Fat=[198,119,259,295,10];
S2Protein=[100,37,13,22,6];
S2Sodium=[2940,3410,2960,2020,1110];
S2Potassium=[3360,0,8680,7860,0];
S2VitaminA=[0,0,0,690,0];
S2VitaminC=[9,2,0,321,816];
S2Calcium=[610,9430,520,90,0];
S2Iron=[29,8,7,9,0];
S2Magnesium=[970,0,460,710,0];

S3Count = 5;
minS3=3;
maxS3=6;
S3Calories=[710,540,530,630,420];
S3Carbohydrate=[174,131,111,153,106];
```

```
S3Fat=[0,2,6,4,2];
S3Protein=[6,1,19,13,5];
S3Sodium=[40,90,0,20,0];
S3Potassium=[2760,2140,0,3220,1210];
S3VitaminA=[0,0,360,120,0];
S3VitaminC=[40,1,110,1810,38];
S3Calcium=[120,110,90,550,40];
S3Iron=[11,1,10,15,0];
S3Magnesium=[140,70,0,240,80];
Nutrients =
[1200000,165000,40000,45000,1600000,3500000,600000,40000,700000,1480
0,270000];
```

Παράρτημα Δ

Εδώ παρουσιάζονται αποσπάσματα κώδικα του συστήματος.

Δ.1 Κώδικας Υπολογισμού BMR, BMI και δημιουργίας του νέου χρήστη.

```
    username = jTextField13.getText();
    password = jPasswordField2.getText();
    String passconfirm = jPasswordField3.getText();
    if (!username.equals("") && !password.equals("") && !passconfirm.equals("")) {
       //check if password is confirmed
       if (password.equals(passconfirm)) {
          jLabel32.setText("");
          jLabel33.setText("");
          jButton10.setEnabled(false);
          jTabbedPane1.setSelectedComponent(jPanel3);
       } else {
          jPasswordField2.setText("");
          jPasswordField3.setText("");
          jLabel33.setText("*Your Passwords don't match. Please try again!");
       }
 } else {
       jLabel32.setText("(*): Required Fields");
  }
 double weightfactor = 0;
    double heightfactor = 0;
    double genderfactor;
    double agefactor;

    double BMI;
    //get age

    String ageString = jTextField6.getText();
    String weightString = jTextField3.getText();
    String heightString = jTextField4.getText();
```

```
//if fields not filled

if (!ageString.equals("") && !weightString.equals("") && !heightString.equals("")) {
   System.out.println("Please add Age");
   jLabel15.setText(" ");
   jLabel16.setText(" ");
   jLabel17.setText(" ");

   int age = Integer.parseInt(ageString);
   //if age not valid
   if (age > 120 && age < 12) {
   }
   System.out.println("Age: " + age);

   //get gender
   //if female
   if (jRadioButton1.isSelected()) {
      genderfactor = 655;
      agefactor = 4.7;
      //pounds or kilogramms
      int weightUnitIndex = jComboBox4.getSelectedIndex();

      //if pounds
      if (weightUnitIndex == 0) {
         weightfactor = 4.35;
      } //else if kilos
      else if (weightUnitIndex == 1) {
         weightfactor = 9.6;
      }
      System.out.println("Selected weight unit: " + weightUnitIndex);

      //feet or meters
      int heightUnitIndex = jComboBox5.getSelectedIndex();
      //if feet
```

```
    if (heightUnitIndex == 0) {
        heightfactor = 4.7;
    } //else if meters
    else if (heightUnitIndex == 1) {
        heightfactor = 1.8;
    }
    System.out.println("Selected weight unit: " + weightUnitIndex);

} //else if male
else {
    genderfactor = 66;
    agefactor = 6.8;
    //pounds or kilogramms
    int weightUnitIndex = jComboBox4.getSelectedIndex();

    //if pounds
    if (weightUnitIndex == 0) {
        weightfactor = 6.23;
    } //else if kilos
    else if (weightUnitIndex == 1) {
        weightfactor = 13.7;
    }
    System.out.println("Selected weight unit: " + weightUnitIndex);

    //feet or meters
    int heightUnitIndex = jComboBox5.getSelectedIndex();
    //if feet
    if (heightUnitIndex == 0) {
        heightfactor = 12.7;
    } //else if meters
    else if (heightUnitIndex == 1) {
        heightfactor = 5;
    }
    System.out.println("Selected weight unit: " + weightUnitIndex);
```

```
        }
        //get weight

        //if weight not certified
        if (weightString == null) {

           System.out.println("Weight not given");
        }
        double weight = Double.parseDouble(weightString);
        System.out.println("Weight: " + weight);

        //get height

        //if height not certified
        if (heightString == null) {
           System.out.println("Height not given");
        }
        double height = Double.parseDouble(heightString) * 100;
        System.out.println("Height: " + height);



        BMR = genderfactor + (weightfactor * weight) + (heightfactor * height) - (agefactor * age);
        BMI = weight / (Double.parseDouble(heightString) * Double.parseDouble(heightString));

        jLabel108.setText(Double.toString(RoundedValue(BMI)));
        double weight1 = 18.5 * Double.parseDouble(heightString) *
Double.parseDouble(heightString);
        double weight2 = 25 * Double.parseDouble(heightString) * Double.parseDouble(heightString);

        if (BMI < 15.0) {
           jLabel4.setText("Very severely underweight");
        } else if (BMI > 15.0 && BMI < 16.0) {
```

```
            jLabel4.setText("Severely underweight");
        } else if (BMI > 16.0 && BMI < 18.5) {
            jLabel4.setText("Underweight");
        } else if (BMI > 18.5 && BMI < 25.0) {
            jLabel4.setText("Normal (healthy weight)");
        } else if (BMI > 25.0 && BMI < 30.0) {
            jLabel4.setText("Overweight");
        } else if (BMI > 30.0 && BMI < 35.0) {
            jLabel4.setText("Obese Class I (Moderately obese)");
        } else if (BMI > 35.0 && BMI < 40.0) {
            jLabel4.setText("Obese Class II (Severely obese)");
        } else if (BMI > 40.0) {
            jLabel4.setText("Obese Class III (Very severely obese)");
        }

        jLabel111.setText(Double.toString(RoundedValue(weight1)));
        jLabel159.setText(Double.toString(RoundedValue(weight2)));

        System.out.println("BMR= " + genderfactor + "+(" + weightfactor + "*" + weight + ")+(" +
heightfactor + "*" + height + ")-(" + agefactor + "*" + age + ")= " + BMR);
        //button settings for next tab
        jButton16.setEnabled(true);
        jButton14.setEnabled(false);
        //go to next tab
        jTabbedPane1.setSelectedComponent(jPanel4);
      } else {
        if (ageString.equals("")) {
            jLabel15.setText("*Required Field");
        } else {
            jLabel15.setText(" ");
        }
        if (weightString.equals("")) {
            jLabel16.setText("*Required Field");
        } else {
```

```
            jLabel16.setText(" ");
        }
        if (heightString.equals("")) {
            jLabel17.setText("*Required Field");
        } else {
            jLabel17.setText(" ");
        }
    }
 double exercisefactor = 0.0;

    //if sedentary
    if (jRadioButton6.isSelected()) {
        exercisefactor = 1.2;
        User_Exercise = jRadioButton6.getText();
    } //if lightly active
    else if (jRadioButton7.isSelected()) {
        exercisefactor = 1.375;
        User_Exercise = jRadioButton7.getText();
    } //if moderatetely active
    else if (jRadioButton8.isSelected()) {
        exercisefactor = 1.55;
        User_Exercise = jRadioButton8.getText();
    } //if active
    else if (jRadioButton9.isSelected()) {
        exercisefactor = 1.725;
        User_Exercise = jRadioButton9.getText();
    } //if very active
    else if (jRadioButton10.isSelected()) {
        exercisefactor = 1.9;
        User_Exercise = jRadioButton10.getText();
    }
    MaintainCalories = BMR * exercisefactor;
    System.out.println("Your exercise factor is: " + exercisefactor + " so you should consume: " +
MaintainCalories + " Calories Daily.");
```

```
        //button settings for next tab
        jButton18.setEnabled(false);

        jLabel152.setText(User_Exercise);
        //go to next tab
        jTabbedPane1.setSelectedComponent(jPanel5);

if (jRadioButton3.isSelected()) {
           Calories = MaintainCalories - 500;

        } //if maintain
        else if (jRadioButton4.isSelected()) {
           Calories = MaintainCalories;
        } //if gain
        else if (jRadioButton5.isSelected()) {
           Calories = MaintainCalories + 500;
        }

        tempcal = (int) Calories;
        tempcal = tempcal / 100;
        tempcal = tempcal * 100;
        String Cal = Integer.toString(tempcal);

        label12.setText(Cal);
        //button settings for next tab
        jButton23.setEnabled(false);
        //go to next tab
        jTabbedPane1.setSelectedComponent(jPanel6);
String name = jTextField5.getText();
        String surname = jTextField7.getText();
        String gender = null;
        if (jRadioButton1.isSelected()) {
           gender = "F";
        } else {
```

```
    gender = "M";
}
int age = Integer.parseInt(jTextField6.getText());
Double weight = Double.parseDouble(jTextField3.getText());
String w_unit = null;
if (jComboBox4.getSelectedIndex() == 0) {
    w_unit = "L";
} else {
    w_unit = "K";
}
AGE=age;
Double height = Double.parseDouble(jTextField4.getText());
String h_unit = null;
if (jComboBox5.getSelectedIndex() == 0) {
    h_unit = "F";
} else {
    h_unit = "M";
}

Double Target_W = Double.parseDouble(jTextField9.getText());
String target_unit = null;
if (jComboBox6.getSelectedIndex() == 0) {
    target_unit = "L";
} else {
    target_unit = "K";
}

String Goal = null;
if (jRadioButton3.isSelected()) {
    Goal = "Loose";
} else if (jRadioButton4.isSelected()) {
    Goal = "Maintain";
} else if (jRadioButton5.isSelected()) {
    Goal = "Gain";
```

```
        }


        Double BMR = MaintainCalories;

        jLabel156.setText(Double.toString(tempcal));
        try {
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
            //getConnection();
            Statement stmt = (Statement) con.createStatement();

            int rs = stmt.executeUpdate("insert into `nutrient_data`.`user_details` (`Username`, `Password`, `Name`, `Surname`, `Age`, `Gender`, `Weight`, `Weight_unit`, `Height`, `Height_unit`, `Target_Weight`, `Target_unit`, `Goal`, `BMR`, `Suggested_Calories`, `Exercise`) "
                    + "values(\"" + username + "\",\"" + password + "\",\"" + name + "\",\"" + surname + "\"," + AGE + ", \"" + gender + "\", " + weight + ", \"" + w_unit + "\", " + height + ", \"" + h_unit + "\", " + Target_W + ", \"" + target_unit + "\", \"" + Goal + "\", " + BMR + "," + tempcal + ", \"" + User_Exercise + "\");");


        } catch (SQLException e) {
            System.err.println(e);
        }

//      jLabel156.setText(Double.toString(BMR));

        //close frame
        jInternalFrame3.setVisible(false);

        //load data in user profile frame

        jLabel60.setText(name + " " + surname);
```

```
jLabel61.setText(Integer.toString(AGE) + " years");
if (gender.equals("F")) {
   jLabel62.setText("Female");
} else {
   jLabel62.setText("Male");
}


jLabel65.setText(Double.toString(height));
jLabel66.setText(Double.toString(weight));

if (Goal.equals("Loose")) {
   jLabel68.setText("To Loose Weight");
} else if (Goal.equals("Maintain")) {
   jLabel68.setText("To Maintain Weight");
} else if (Goal.equals("Gain")) {
   jLabel68.setText("To Gain Weight");
}
jLabel69.setText(Double.toString(Target_W));
//show user profile
jInternalFrame8.setVisible(true);
```

Δ.2 Κώδικας Log In

```
username = jTextField12.getText();
password = jPasswordField1.getText();
//System.out.println("Password: "+password);

//fields filled
if (!username.equals("") && !password.equals("")) {
   int count = 0;
   String name = null;
   String surname = null;
   int age = 0;
```

```
String Goal = null;
String Exercise = null;

Double height = 0.0, weight = 0.0, Target_W = 0.0;
try {
  Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
  //getConnection();
  Statement stmt = (Statement) con.createStatement();

  ResultSet rs = stmt.executeQuery("select * from `nutrient_data`.`user_details` where `Username`='" + username + "' and `Password`='" + password + "';");

  while (rs.next()) {
    ID = rs.getInt("id_user");

    name = rs.getString("Name");
    surname = rs.getString("Surname");
    AGE = rs.getInt("Age");
    gender = rs.getString("Gender");
    height = rs.getDouble("Height");
    weight = rs.getDouble("Weight");
    Goal = rs.getString("Goal");
    Target_W = rs.getDouble("Target_Weight");
    Exercise = rs.getString("Exercise");

    BMR = rs.getDouble("Suggested_Calories");
    count++;
  }
} catch (SQLException e) {
  System.err.println(e);
}
if (count != 0) {
```

```
            System.out.println("User Found");
            jLabel156.setText(Double.toString(BMR));
            //load data to profile
            jLabel60.setText(name + " " + surname);

            jLabel61.setText(Integer.toString(AGE) + " years");
            if (gender.equals("F")) {
               jLabel62.setText("Female");
            } else {
               jLabel62.setText("Male");
            }

            jLabel152.setText(Exercise);
            jLabel65.setText(Double.toString(height));
            jLabel66.setText(Double.toString(weight));
            jLabel68.setText(Goal);

            jLabel69.setText(Double.toString(Target_W));
            jInternalFrame9.setVisible(false);
            jTextField12.setText("");
            jPasswordField1.setText("");
            if (flag == 1) {
               jInternalFrame8.setVisible(true);
            } else if (flag == 2) {   //launch quick nutrition tool

               int id_no = 0;
               try {
                  Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

                  Statement stmt = (Statement) con.createStatement();

                  ResultSet rs = stmt.executeQuery("select * from nutrient_data.candidate_categories
where (min_age<'" + AGE + "' and max_age>'" + AGE + "') and sex='" + gender + "';");
```

```
            int count3 = 0;

            while (rs.next()) {

               id_no = rs.getInt("id_candidate");
               count3++;
            }
            System.out.println("Id: " + id_no);

         } catch (SQLException e) {
            System.err.println(e);
         }
         int nutr_code = 0;
         try {
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
            //getConnection();
            Statement stmt = (Statement) con.createStatement();

            ResultSet rs = stmt.executeQuery("select * from nutrient_data.nutrient_requirements
where candidate_id='" + id_no + "';");
            int count4 = 0;

            while (rs.next()) {

               nutr_code = rs.getInt("nutr_code");

               //natrio
               if (nutr_code == 404) {
                  Thiamin = rs.getDouble("nutr_value");
               } //kalio
               else if (nutr_code == 328 || nutr_code == 325 || nutr_code == 326) {
                  VitaminD = rs.getDouble("nutr_value");
               } else if (nutr_code == 401) {
```

```
                    VitaminC = rs.getDouble("nutr_value");
                } else if (nutr_code == 301) {
                    Calcium = rs.getDouble("nutr_value");
                } else if (nutr_code == 303) {
                    Iron = rs.getDouble("nutr_value");
                }
                count4++;
            }
            System.out.println("Thiamin: " + Thiamin + " VitaminD: " + VitaminD + " Vitamin A: " +
VitaminC + " Vitamin C: " + VitaminC + " Calcium: " + Calcium + " Iron: " + Iron);

        } catch (SQLException e) {
            System.err.println(e);
        }

         QuickNutrFrame quickpanel=new QuickNutrFrame(username, password, ID,BMR,
Thiamin , VitaminD, VitaminC, Calcium, Iron);

          quickpanel.setLocationRelativeTo(null);
          quickpanel.setVisible(true);

      } else if (flag == 3) {
         jLabel46.setVisible(false);


            NutritionPlanFrame mynutritionplan=new NutritionPlanFrame(ID, BMR
,Iron,Calcium,VitaminC,VitaminD,Thiamin,prefNutrient, 0);
      mynutritionplan.setLocationRelativeTo(null);
            mynutritionplan.setVisible(true);

      } else if (flag == 4) {
         DefaultTableModel tableModel;

           String [] prefered_Name=new String[100];
```

```
int count1 = 0;
System.out.println("ID: " + ID);
try {
   Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
   //getConnection();
   Statement stmt = (Statement) con.createStatement();

   ResultSet rs = stmt.executeQuery("select idnutrition_plans,Nutrition_Date,PreferedId,PreferedName,Prefered_Limit from nutrient_data.nutrition_plans where user_id=" + ID + ";");

   while (rs.next()) {
      Date[count1] = rs.getString("Nutrition_Date");
      idnutritions[count1] = rs.getInt("idnutrition_plans");
      Prefered_Limit[count] = rs.getDouble("Prefered_Limit");
      prefered_id[count] = rs.getInt("PreferedId");
      prefered_Name[count]=rs.getString("PreferedName");
      count1++;
   }
   System.out.println(Date[0]);

} catch (SQLException e) {
   System.err.println(e);
}
if (count1 == 0) {
   jInternalFrame16.setVisible(true);
} else {
   tableModel = (DefaultTableModel) jTable5.getModel();
   String[] data = new String[3];
   for (int i = 0; i < count1; i++) {
      //data[2]=
      data[0] = Date[i];
```

```
                    data[1] = prefered_Name[i];
                    data[2] = "";
                    tableModel.addRow(data);
                    jTable5.setModel(tableModel);
                }

                jInternalFrame10.setVisible(true);
            }
        }
                jLabel124.setText("[Log Out]");
            jLabel126.setText(name + " " + surname);
            status = 1;

        } else {
            System.out.println("User not Found!");
            jTextField12.setText("");
            jPasswordField1.setText("");
            jLabel35.setText("*User Not Found! Please try again!");
        }
```

Δ.3 Κώδικας Αναζήτησης Φαγητού

```
 for (int i = 0; i < measures.length; i++) {
        measures[i] = null;

    }
    String searchWord = jTextField14.getText();
    String[] splitted = searchWord.split(" ");
    for (int i = 0; i < foods.length; i++) {
        foods[i] = null;
    }

    jList4.setModel(new javax.swing.AbstractListModel() {

        public int getSize() {
```

```
            return 0;
        }

        public Object getElementAt(int i) {
            return null;
        }
    });
    try {
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
        //getConnection();
        Statement stmt = (Statement) con.createStatement();

        ResultSet rs = stmt.executeQuery(q1.searchQuery(splitted));
        int count = 0;
        while (rs.next()) {
            //pairnw ta dedomena me vasi to onoma tis stilis
            String s = rs.getString("long_desc");
            System.out.println("food: " + s);
            foods[count] = s;
            count++;
        }
    } catch (SQLException e) {
        System.err.println(e);
    }
    jList4.setModel(new javax.swing.AbstractListModel() {

        public int getSize() {
            return foods.length;
        }

        public Object getElementAt(int i) {
            return foods[i];
        }  });
```

Δ.4 Κώδικας Πρόσθεσης Φαγητού στο πρόγραμμα διατροφής

```
String Carbs = null;
        String Fat = null;
        String Protein = null;
        double Sugars = 0;
        double VitD = 0;
        double VitC = 0;
        double Calcium = 0;
        double Iron = 0;
        double Choles = 0;
        double Thiamin = 0;
        String Calories = null;

        // TODO add your handling code here:
        System.out.println("mpikeeee!!! " + db_no);
        try {
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
            //getConnection();
            Statement stmt = (Statement) con.createStatement();

            ResultSet rs = stmt.executeQuery("select `Nutr_No`, `Nutr_Val` from `nutrient_data`.`nutrient_data` where `nutrient_data`.`NDB_No`=" + Integer.toString(db_no) + ";");

            int count = 0;
            while (rs.next()) {
                //pairnw ta dedomena me vasi to onoma tis stilis

                int i = rs.getInt("Nutr_No");

                NutrCode[count] = i;

                Double w = rs.getDouble("Nutr_Val");
```

```
            System.out.println(NutrCode[count] + " with weight: " + w);
            Nutrients[count] = w;

            count++;
          }
        } catch (SQLException e) {
          System.err.println(e);
        }
        //  double amount=Double.parseDouble(jTextField11.getText());
        int selected = jComboBox2.getSelectedIndex();
        //System.out.println("weight " + weights[selected] + " measure " + measures[selected] + " " +
(String) jList1.getSelectedValue());

        //ipologismos thermidwn gia kathe merida kai sta 100 grams
        int ingrmeasure = jComboBox2.getSelectedIndex();
        System.out.println(weights[ingrmeasure]);
        double weight = (weights[ingrmeasure] * Double.parseDouble(jTextField15.getText())) /
Amounts[ingrmeasure];
        System.out.println("WEIGHT FOUND: " + weight);
        for (int k = 0; NutrCode[k] != null; k++) {
          //  System.out.println(NutrCode[k]);
          //calories
          if (NutrCode[k].equals(208)) {

            double temp1 = RoundedValue((weight * Nutrients[k]) / 100);
            Calories = Double.toString(temp1);
          }
          //carbs
          if (NutrCode[k].equals(205)) {
            double temp2 = RoundedValue((weight * Nutrients[k]) / 100);
            Carbs = Double.toString(temp2);
             }
          //fat
```

```
 if (NutrCode[k].equals(204)) {
    double temp3 = RoundedValue((weight * Nutrients[k]) / 100);
    Fat = Double.toString(temp3);
 }
  //protein
 if (NutrCode[k].equals(203)) {
    double temp4 =RoundedValue((weight * Nutrients[k]) / 100);

    Protein = Double.toString(temp4);
  }
  //sugars
 if (NutrCode[k].equals(269)) {
    double temp5 = RoundedValue((weight * Nutrients[k]) / 100);
    Sugars = temp5;
 }
   //Calcium
 if (NutrCode[k].equals(301)) {
    double temp6 = RoundedValue((weight * Nutrients[k]) / 100);
    Calcium = temp6;
}
   //Iron
 if (NutrCode[k].equals(303)) {
    double temp7 =RoundedValue( (weight * Nutrients[k]) / 100);
    Iron = temp7;
}
   //Thiamin
 if (NutrCode[k].equals(404)) {
    double temp8 = RoundedValue((weight * Nutrients[k]) / 100);
    Thiamin =temp8;

   }
   //VitaminD
 if (NutrCode[k].equals(328)) {
    double temp9 = RoundedValue((weight * Nutrients[k]) / 100);
```

```
        VitD = temp9;
    }
        //VitaminC
      if (NutrCode[k].equals(269)) {
        double temp10 = RoundedValue((weight * Nutrients[k]) / 100);
        VitC = temp10;
    }
        //Cholesterol
      if (NutrCode[k].equals(269)) {
        double temp11 = RoundedValue((weight * Nutrients[k]) / 100);
        Choles =temp11;
     }
    }
   //GET SELECTED INDEX
    int selectedmeal = jComboBox3.getSelectedIndex();

    //add data in Breakfast
    if (selectedmeal == 0) {
      DefaultTableModel tableModel = (DefaultTableModel) jTable2.getModel();

      String[] data;
      if(Prefered!=0)
        data= new String[7];
      else
        data= new String[6];
      data[0] = jList4.getSelectedValue().toString();
      data[1] = jTextField15.getText() + " " + jComboBox2.getSelectedItem().toString();
      data[2] = Carbs;
      data[3] = Fat;
      data[4] = Protein;
      data[5] = Calories;

      BreakfastNutr[0] += Double.parseDouble(Carbs);
      BreakfastNutr[1] += Double.parseDouble(Fat);
```

```
BreakfastNutr[2] += Double.parseDouble(Protein);
BreakfastNutr[3] += Double.parseDouble(Calories);
BreakfastNutr[4] += Sugars;
BreakfastNutr[5] += Calcium;
BreakfastNutr[6] += Iron;
BreakfastNutr[7] += Thiamin;
BreakfastNutr[8] += Choles;
BreakfastNutr[9] += VitC;
BreakfastNutr[10] += VitD;

if(Prefered==269){
   data[6] = Double.toString(Sugars);
   jLabel1.setText(Double.toString(BreakfastNutr[4]));
 }
else if (Prefered==301){
   data[6] =Double.toString( Calcium);
jLabel1.setText(Double.toString(BreakfastNutr[5]));}
else if (Prefered==303){
   data[6] = Double.toString(Iron);
jLabel1.setText(Double.toString(BreakfastNutr[6]));
}
else if (Prefered==404){
   data[6] = Double.toString(Thiamin);
jLabel1.setText(Double.toString(BreakfastNutr[7]));
}
else if (Prefered==601){
   data[6] = Double.toString(Choles);
jLabel1.setText(Double.toString(BreakfastNutr[8]));
}
else if (Prefered==401){
   data[6] = Double.toString(VitC);
   jLabel1.setText(Double.toString(BreakfastNutr[9]));
}
else if (Prefered==328){
```

```
        data[6] = Double.toString(VitD);
        jLabel1.setText(Double.toString(BreakfastNutr[10]));
    }
    else {
     jLabel1.setText("");
    }

    tableModel.addRow(data);
    jTable2.setModel(tableModel);

    jLabel42.setText(Double.toString(BreakfastNutr[0]));
    jLabel43.setText(Double.toString(BreakfastNutr[1]));
    jLabel45.setText(Double.toString(BreakfastNutr[2]));
    jLabel47.setText(Double.toString(BreakfastNutr[3]));

} //add data in Lunch
else if (selectedmeal == 1) {
    DefaultTableModel tableModel = (DefaultTableModel) jTable3.getModel();

    String[] data;
    if(Prefered!=0)
        data= new String[7];
    else
        data= new String[6];
    data[0] = jList4.getSelectedValue().toString();
    data[1] = jTextField15.getText() + " " + jComboBox2.getSelectedItem().toString();
    data[2] = Carbs;
    data[3] = Fat;
    data[4] = Protein;
    data[5] = Calories;

    LunchNutr[0] += Double.parseDouble(Carbs);
    LunchNutr[1] += Double.parseDouble(Fat);
    LunchNutr[2] += Double.parseDouble(Protein);
```

```
LunchNutr[3] += Double.parseDouble(Calories);
LunchNutr[4] += Sugars;
LunchNutr[5] += Calcium;
LunchNutr[6] += Iron;
LunchNutr[7] += Thiamin;
LunchNutr[8] += Choles;
LunchNutr[9] += VitC;
LunchNutr[10] += VitD;

 if(Prefered==269){
   data[6] = Double.toString(Sugars);
   jLabel2.setText(Double.toString(LunchNutr[4]));
 }
else if (Prefered==301){
   data[6] =Double.toString( Calcium);
jLabel2.setText(Double.toString(LunchNutr[5]));}
else if (Prefered==303){
   data[6] = Double.toString(Iron);
jLabel2.setText(Double.toString(LunchNutr[6]));
}
else if (Prefered==404){
   data[6] = Double.toString(Thiamin);
jLabel2.setText(Double.toString(LunchNutr[7]));
}
else if (Prefered==601){
   data[6] = Double.toString(Choles);
jLabel2.setText(Double.toString(LunchNutr[8]));
}
else if (Prefered==401){
   data[6] = Double.toString(VitC);
   jLabel2.setText(Double.toString(LunchNutr[9]));
}
else if (Prefered==328){
   data[6] = Double.toString(VitD);
```

```
        jLabel2.setText(Double.toString(LunchNutr[10]));
    }
    else {
     jLabel2.setText("");
    }

    tableModel.addRow(data);
    jTable3.setModel(tableModel);
    jLabel80.setText(Double.toString(LunchNutr[0]));
    jLabel82.setText(Double.toString(LunchNutr[1]));
    jLabel84.setText(Double.toString(LunchNutr[2]));
    jLabel86.setText(Double.toString(LunchNutr[3]));

} //add data in Dinner
else if (selectedmeal == 2) {
    DefaultTableModel tableModel = (DefaultTableModel) jTable1.getModel();
    String[] data;
    if(Prefered!=0)
        data= new String[7];
    else
        data= new String[6];
    data[0] = jList4.getSelectedValue().toString();
    data[1] = jTextField15.getText() + " " + jComboBox2.getSelectedItem().toString();
    data[2] = Carbs;
    data[3] = Fat;
    data[4] = Protein;
    data[5] = Calories;

    DinnerNutr[0] += Double.parseDouble(Carbs);
    DinnerNutr[1] += Double.parseDouble(Fat);
    DinnerNutr[2] += Double.parseDouble(Protein);
    DinnerNutr[3] += Double.parseDouble(Calories);
    DinnerNutr[4] += Sugars;
    DinnerNutr[5] += Calcium;
```

```
DinnerNutr[6] += Iron;
DinnerNutr[7] += Thiamin;
DinnerNutr[8] += Choles;
DinnerNutr[9] += VitC;
DinnerNutr[10] += VitD;

 if(Prefered==269){
    data[6] = Double.toString(Sugars);
    jLabel3.setText(Double.toString(DinnerNutr[4]));
  }
else if (Prefered==301){
    data[6] =Double.toString( Calcium);
jLabel3.setText(Double.toString(DinnerNutr[5]));}
else if (Prefered==303){
    data[6] = Double.toString(Iron);
jLabel3.setText(Double.toString(DinnerNutr[6]));
}
else if (Prefered==404){
    data[6] = Double.toString(Thiamin);
jLabel3.setText(Double.toString(DinnerNutr[7]));
}
else if (Prefered==601){
    data[6] = Double.toString(Choles);
jLabel3.setText(Double.toString(DinnerNutr[8]));
}
else if (Prefered==401){
    data[6] = Double.toString(VitC);
    jLabel3.setText(Double.toString(DinnerNutr[9]));
}
else if (Prefered==328){
    data[6] = Double.toString(VitD);
    jLabel3.setText(Double.toString(DinnerNutr[10]));
}
else {
```

```
    jLabel3.setText("");
    }
    tableModel.addRow(data);
    jTable1.setModel(tableModel);


    jLabel88.setText(Double.toString(DinnerNutr[0]));
    jLabel90.setText(Double.toString(DinnerNutr[1]));
    jLabel91.setText(Double.toString(DinnerNutr[2]));
    jLabel94.setText(Double.toString(DinnerNutr[3]));

} //add data in Snacks
else if (selectedmeal == 3) {
    DefaultTableModel tableModel = (DefaultTableModel) jTable4.getModel();
    String[] data;
    if(Prefered!=0)
        data= new String[7];
    else
        data= new String[6];
    data[0] = jList4.getSelectedValue().toString();
    data[1] = jTextField15.getText() + " " + jComboBox2.getSelectedItem().toString();
    data[2] = Carbs;
    data[3] = Fat;
    data[4] = Protein;
    data[5] = Calories;

    SnacksNutr[0] += Double.parseDouble(Carbs);
    SnacksNutr[1] += Double.parseDouble(Fat);
    SnacksNutr[2] += Double.parseDouble(Protein);
    SnacksNutr[3] += Double.parseDouble(Calories);
    SnacksNutr[4] += Sugars;
    SnacksNutr[5] += Calcium;
    SnacksNutr[6] += Iron;
    SnacksNutr[7] += Thiamin;
```

```
SnacksNutr[8] += Choles;
SnacksNutr[9] += VitC;
SnacksNutr[10] += VitD;

if(Prefered==269){
   data[6] = Double.toString(Sugars);
   jLabel4.setText(Double.toString(SnacksNutr[4]));
 }
else if (Prefered==301){
   data[6] =Double.toString( Calcium);
jLabel4.setText(Double.toString(SnacksNutr[5]));}
else if (Prefered==303){
   data[6] = Double.toString(Iron);
jLabel4.setText(Double.toString(SnacksNutr[6]));
}
else if (Prefered==404){
   data[6] = Double.toString(Thiamin);
jLabel4.setText(Double.toString(SnacksNutr[7]));
}
else if (Prefered==601){
   data[6] = Double.toString(Choles);
jLabel4.setText(Double.toString(SnacksNutr[8]));
}
else if (Prefered==401){
   data[6] = Double.toString(VitC);
   jLabel4.setText(Double.toString(SnacksNutr[9]));
}
else if (Prefered==328){
   data[6] = Double.toString(VitD);
   jLabel4.setText(Double.toString(SnacksNutr[10]));
}
else {
 jLabel4.setText("");
}
```

```
        tableModel.addRow(data);
        jTable4.setModel(tableModel);



        jLabel97.setText(Double.toString(SnacksNutr[0]));
        jLabel98.setText(Double.toString(SnacksNutr[1]));
        jLabel99.setText(Double.toString(SnacksNutr[2]));
        jLabel101.setText(Double.toString(SnacksNutr[3]));
    }

    SetNewTotalsForNutrition(BreakfastNutr, LunchNutr, DinnerNutr, SnacksNutr);
    jButton28.setEnabled(true);
```

Δ.5 Κώδικας Αποθήκευσης Διατροφής ή Ανανέωσης (Update)

```
  String Food = null;
    String Amount = null;
    String Carbs = null;
    String Fat = null;
    String Protein = null;
    String Calories = null;
    String Meal = null;
    String PreferedValue=null;
    String Nutrition_Date = null;
    double PreferedLimit=0;

    if (updateFlag == 0) {

        if(Prefered==269){
          PreferedLimit=Sugar;
          PreferedName="Sugars";
         }
        else if (Prefered==301){
           PreferedLimit=Calcium;
```

```
   PreferedName="Calcium";
}
else if (Prefered==303){
   PreferedLimit=Iron;
   PreferedName="Iron";

}
else if (Prefered==404){
   PreferedLimit=Thiamin;
    PreferedName="Thiamin";

}
else if (Prefered==601){
   PreferedLimit=Cholesterol;
    PreferedName="Cholesterol";

}
else if (Prefered==401){
   PreferedLimit=VitaminC;
     PreferedName="VitaminC";

}
else if (Prefered==328){
   PreferedLimit=VitaminD;
    PreferedName="VitaminD";

}
else {
 PreferedName="";
}


//save breakfast
DefaultTableModel tableModel = (DefaultTableModel) jTable2.getModel();
```

```
        System.out.println("TABLE ROW COUNT: " + tableModel.getRowCount());

        System.out.println("Food: " + tableModel.getValueAt(0, 0));
        Nutrition_Date = jDateChooser1.getDate().toString();

        try {
           Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
           //getConnection();
           Statement stmt = (Statement) con.createStatement();

           int rs = stmt.executeUpdate("insert into `nutrient_data`.`nutrition_plans` ( `user_id`, `Nutrition_Date`, `date_created`, `PreferedId`, `PreferedName`, `Prefered_Limit`) values(" + ID + ", \"" + Nutrition_Date + "\", NOW(), "+Prefered+", \""+PreferedName+"\", "+PreferedLimit+");");


        } catch (SQLException e) {
           System.err.println(e);
        }

        for (int rows = 0; rows < tableModel.getRowCount(); rows++) {

           Food = tableModel.getValueAt(rows, 0).toString();
           Amount = tableModel.getValueAt(rows, 1).toString();
           Carbs = tableModel.getValueAt(rows, 2).toString();
           Fat = tableModel.getValueAt(rows, 3).toString();
           Protein = tableModel.getValueAt(rows, 4).toString();
           Calories = tableModel.getValueAt(rows, 5).toString();
           PreferedValue=tableModel.getValueAt(rows, 6).toString();
           Meal = "Breakfast";

           try {
              Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
```

```
            //getConnection();
            Statement stmt = (Statement) con.createStatement();

            int rs = stmt.executeUpdate("insert into `nutrient_data`.`nutrition_plans_food` ( `Nutrition_Plans_id`,`Food`, `Amount`, `Carbohydrate`, `Fat`, `Protein`, `Calories`,`Prefered`, `Meal`) "
                    + "values((select idnutrition_plans from nutrient_data.nutrition_plans where date_created=("
                    + "select max(date_created) from nutrient_data.nutrition_plans where user_id=" + ID + ")), \"" + Food + "\",\"" + Amount + "\",\"" + Carbs + "\",\"" + Fat + "\", \"" + Protein + "\", \"" + Calories + "\", \"" + PreferedValue + "\", \"" + Meal + "\");");


        } catch (SQLException e) {
            System.err.println(e);
        }
    }
    //save LUNCH
    DefaultTableModel tableModel2 = (DefaultTableModel) jTable3.getModel();
    System.out.println("TABLE ROW COUNT: " + tableModel2.getRowCount());

    for (int rows = 0; rows < tableModel2.getRowCount(); rows++) {

        Food = tableModel2.getValueAt(rows, 0).toString();
        Amount = tableModel2.getValueAt(rows, 1).toString();
        Carbs = tableModel2.getValueAt(rows, 2).toString();
        Fat = tableModel2.getValueAt(rows, 3).toString();
        Protein = tableModel2.getValueAt(rows, 4).toString();
        Calories = tableModel2.getValueAt(rows, 5).toString();
        PreferedValue=tableModel2.getValueAt(rows, 6).toString();
        Meal = "Lunch";

        try {
```

```
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
            //getConnection();
            Statement stmt = (Statement) con.createStatement();

            int rs = stmt.executeUpdate("insert into `nutrient_data`.`nutrition_plans_food` (
`Nutrition_Plans_id`,`Food`, `Amount`, `Carbohydrate`, `Fat`, `Protein`, `Calories`,`Prefered`, `Meal`)
"
                    + "values((select idnutrition_plans from nutrient_data.nutrition_plans where
date_created=("
                    + "select max(date_created) from nutrient_data.nutrition_plans where user_id=" +
ID + ")), \"" + Food + "\",\"" + Amount + "\",\"" + Carbs + "\",\"" + Fat + "\", \"" + Protein + "\", \"" +
Calories + "\", \"" + PreferedValue + "\", \"" + Meal + "\");");

            } catch (SQLException e) {
                System.err.println(e);
            }
        }

        //save Dinner
        DefaultTableModel tableModel3 = (DefaultTableModel) jTable1.getModel();
        System.out.println("TABLE ROW COUNT: " + tableModel3.getRowCount());


        for (int rows = 0; rows < tableModel3.getRowCount(); rows++) {

            Food = tableModel3.getValueAt(rows, 0).toString();
            Amount = tableModel3.getValueAt(rows, 1).toString();
            Carbs = tableModel3.getValueAt(rows, 2).toString();
            Fat = tableModel3.getValueAt(rows, 3).toString();
            Protein = tableModel3.getValueAt(rows, 4).toString();
            Calories = tableModel3.getValueAt(rows, 5).toString();
            PreferedValue=tableModel3.getValueAt(rows, 6).toString();
            Meal = "Dinner";
```

```
            try {
               Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
               //getConnection();
               Statement stmt = (Statement) con.createStatement();

               int rs = stmt.executeUpdate("insert into `nutrient_data`.`nutrition_plans_food` ( `Nutrition_Plans_id`,`Food`, `Amount`, `Carbohydrate`, `Fat`, `Protein`, `Calories`,`Prefered`, `Meal`) "
                     + "values((select idnutrition_plans from nutrient_data.nutrition_plans where date_created=("
                     + "select max(date_created) from nutrient_data.nutrition_plans where user_id=" + ID + ")), \"" + Food + "\",\"" + Amount + "\",\"" + Carbs + "\",\"" + Fat + "\", \"" + Protein + "\", \"" + Calories + "\", \"" + PreferedValue + "\", \"" + Meal + "\");");

            } catch (SQLException e) {
               System.err.println(e);
            }
         }

         //save Snacks
         DefaultTableModel tableModel4 = (DefaultTableModel) jTable4.getModel();
         System.out.println("TABLE ROW COUNT: " + tableModel4.getRowCount());


         for (int rows = 0; rows < tableModel4.getRowCount(); rows++) {

            Food = tableModel4.getValueAt(rows, 0).toString();
            Amount = tableModel4.getValueAt(rows, 1).toString();
            Carbs = tableModel4.getValueAt(rows, 2).toString();
            Fat = tableModel4.getValueAt(rows, 3).toString();
            Protein = tableModel4.getValueAt(rows, 4).toString();
            Calories = tableModel4.getValueAt(rows, 5).toString();
```

```
            PreferedValue=tableModel4.getValueAt(rows, 6).toString();
            Meal = "Snacks";

            try {
               Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

               Statement stmt = (Statement) con.createStatement();

               int rs = stmt.executeUpdate("insert into `nutrient_data`.`nutrition_plans_food` ( `Nutrition_Plans_id`,`Food`, `Amount`, `Carbohydrate`, `Fat`, `Protein`, `Calories`,`Prefered`, `Meal`) "
                     + "values((select idnutrition_plans from nutrient_data.nutrition_plans where date_created=("
                     + "select max(date_created) from nutrient_data.nutrition_plans where user_id=" + ID + ")), \"" + Food + "\",\"" + Amount + "\",\"" + Carbs + "\",\"" + Fat + "\", \"" + Protein + "\", \"" + Calories + "\", \"" + PreferedValue + "\", \"" + Meal + "\");");

            } catch (SQLException e) {
               System.err.println(e);
            }
         }
      } else if (updateFlag == 1) {

         DefaultTableModel tableModel = (DefaultTableModel) jTable2.getModel();

         System.out.println("update Breakfast Table size: " + Food_Id_b.length + " table rows: " + tableModel.getRowCount());

         for (int i = 0, rows = 0; i < Food_Id_b.length && rows < tableModel.getRowCount(); i++, rows++) {

            Food = tableModel.getValueAt(rows, 0).toString();
            Amount = tableModel.getValueAt(rows, 1).toString();
```

```
            Carbs = tableModel.getValueAt(rows, 2).toString();
            Fat = tableModel.getValueAt(rows, 3).toString();
            Protein = tableModel.getValueAt(rows, 4).toString();
            Calories = tableModel.getValueAt(rows, 5).toString();
            PreferedValue=tableModel.getValueAt(rows, 6).toString();
            try {

              Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
              //getConnection();
              Statement stmt = (Statement) con.createStatement();

              int rs = stmt.executeUpdate("UPDATE nutrient_data.nutrition_plans_food "
                  + "SET Food=\"" + Food + "\","
                  + " Amount=\"" + Amount + "\","
                  + " Carbohydrate=\"" + Carbs + "\","
                  + " Fat=\"" + Fat + "\","
                  + " Protein=\"" + Protein + "\","
                  + " Calories=\"" + Calories + "\","
                  + " Prefered=\"" + PreferedValue + "\" "
                  + "WHERE id=" + Food_Id_b[i]);


            } catch (SQLException e) {
              System.err.println(e);
            }
          }
          if (Food_Id_b.length > tableModel.getRowCount()) {

            for (int i = tableModel.getRowCount(); i < Food_Id_b.length; i++) {
              //delete the rest
              try {
                Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
```

```
                //getConnection();
                Statement stmt = (Statement) con.createStatement();

                int rs = stmt.executeUpdate("DELETE FROM nutrient_data.nutrition_plans_food "
                    + "WHERE id=" + Food_Id_b[i] + ";");

            } catch (SQLException e) {
                System.err.println(e);
            }
          }
        } else if (Food_Id_b.length < tableModel.getRowCount()) {

          for (int rows = Food_Id_b.length; rows < tableModel.getRowCount(); rows++) {
            //insert the rest
            Food = tableModel.getValueAt(rows, 0).toString();
            Amount = tableModel.getValueAt(rows, 1).toString();
            Carbs = tableModel.getValueAt(rows, 2).toString();
            Fat = tableModel.getValueAt(rows, 3).toString();
            Protein = tableModel.getValueAt(rows, 4).toString();
            Calories = tableModel.getValueAt(rows, 5).toString();
            PreferedValue=tableModel.getValueAt(rows, 6).toString();
            Meal = "Breakfast";

            try {
                Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
                //getConnection();
                Statement stmt = (Statement) con.createStatement();

            int rs = stmt.executeUpdate("insert into `nutrient_data`.`nutrition_plans_food` (
`Nutrition_Plans_id`,`Food`, `Amount`, `Carbohydrate`, `Fat`, `Protein`, `Calories`,`Prefered`, `Meal`)
"
                + "values((select idnutrition_plans from nutrient_data.nutrition_plans where
date_created=("
```

```
                + "select max(date_created) from nutrient_data.nutrition_plans where user_id=" +
ID + ")), \"" + Food + "\",\"" + Amount + "\",\"" + Carbs + "\",\"" + Fat + "\", \"" + Protein + "\", \"" +
Calories + "\", \"" + PreferedValue + "\", \"" + Meal + "\");");


            } catch (SQLException e) {
               System.err.println(e);
            }
          }
        }
        DefaultTableModel tableModel2 = (DefaultTableModel) jTable3.getModel();
        System.out.println("update Breakfast Table size: " + Food_Id_l.length + " table rows: " +
tableModel2.getRowCount());

        for (int i = 0, rows = 0; i < Food_Id_l.length && rows < tableModel2.getRowCount(); i++,
rows++) {

            Food = tableModel2.getValueAt(rows, 0).toString();
            Amount = tableModel2.getValueAt(rows, 1).toString();
            Carbs = tableModel2.getValueAt(rows, 2).toString();
            Fat = tableModel2.getValueAt(rows, 3).toString();
            Protein = tableModel2.getValueAt(rows, 4).toString();
            Calories = tableModel2.getValueAt(rows, 5).toString();
            PreferedValue=tableModel2.getValueAt(rows, 6).toString();
            try {

              Connection                                    con                                    =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
              //getConnection();
              Statement stmt = (Statement) con.createStatement();

             int rs = stmt.executeUpdate("UPDATE nutrient_data.nutrition_plans_food "
                    + "SET Food=\"" + Food + "\","
                    + " Amount=\"" + Amount + "\","
```

```
                + " Carbohydrate=\"" + Carbs + "\","
                + " Fat=\"" + Fat + "\","
                + " Protein=\"" + Protein + "\","
                + " Calories=\"" + Calories + "\","
                + " Prefered=\"" + PreferedValue + "\" "
                + "WHERE id=" + Food_Id_b[i]);
          } catch (SQLException e) {
            System.err.println(e);
          }
        }
        if (Food_Id_l.length > tableModel2.getRowCount()) {

          for (int i = tableModel2.getRowCount(); i < Food_Id_l.length; i++) {
            //delete the rest
            try {
              Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
              //getConnection();
              Statement stmt = (Statement) con.createStatement();

              int rs = stmt.executeUpdate("DELETE FROM nutrient_data.nutrition_plans_food "
                  + "WHERE id=" + Food_Id_l[i] + ";");

            } catch (SQLException e) {
              System.err.println(e);
            }
          }
        } else if (Food_Id_l.length < tableModel2.getRowCount()) {

          for (int rows = Food_Id_l.length; rows < tableModel2.getRowCount(); rows++) {
            //insert the rest
            Food = tableModel2.getValueAt(rows, 0).toString();
            Amount = tableModel2.getValueAt(rows, 1).toString();
            Carbs = tableModel2.getValueAt(rows, 2).toString();
```

```
            Fat = tableModel2.getValueAt(rows, 3).toString();
            Protein = tableModel2.getValueAt(rows, 4).toString();
            Calories = tableModel2.getValueAt(rows, 5).toString();
            PreferedValue=tableModel2.getValueAt(rows, 6).toString();
            Meal = "Lunch";
            try {
              Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
              //getConnection();
              Statement stmt = (Statement) con.createStatement();

              int rs = stmt.executeUpdate("insert into `nutrient_data`.`nutrition_plans_food` (
`Nutrition_Plans_id`,`Food`, `Amount`, `Carbohydrate`, `Fat`, `Protein`, `Calories`,`Prefered`, `Meal`)
"
                + "values((select idnutrition_plans from nutrient_data.nutrition_plans where
date_created=("
                + "select max(date_created) from nutrient_data.nutrition_plans where user_id=" +
ID + ")), \"" + Food + "\",\"" + Amount + "\",\"" + Carbs + "\",\"" + Fat + "\", \"" + Protein + "\", \"" +
Calories + "\", \"" + PreferedValue + "\", \"" + Meal + "\");");

            } catch (SQLException e) {
              System.err.println(e);
            }
          }

}
        //dinner
        DefaultTableModel tableModel3 = (DefaultTableModel) jTable1.getModel();

        System.out.println("update Breakfast Table size: " + Food_Id_d.length + " table rows: " +
tableModel3.getRowCount());

        for (int i = 0, rows = 0; i < Food_Id_d.length && rows < tableModel3.getRowCount(); i++,
rows++) {
```

```
            Food = tableModel3.getValueAt(rows, 0).toString();
            Amount = tableModel3.getValueAt(rows, 1).toString();
            Carbs = tableModel3.getValueAt(rows, 2).toString();
            Fat = tableModel3.getValueAt(rows, 3).toString();
            Protein = tableModel3.getValueAt(rows, 4).toString();
            Calories = tableModel3.getValueAt(rows, 5).toString();
            PreferedValue=tableModel3.getValueAt(rows, 6).toString();
            try {

              Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
              //getConnection();
              Statement stmt = (Statement) con.createStatement();

              int rs = stmt.executeUpdate("UPDATE nutrient_data.nutrition_plans_food "
                  + "SET Food=\"" + Food + "\","
                  + " Amount=\"" + Amount + "\","
                  + " Carbohydrate=\"" + Carbs + "\","
                  + " Fat=\"" + Fat + "\","
                  + " Protein=\"" + Protein + "\","
                  + " Calories=\"" + Calories + "\","
                  + " Prefered=\"" + PreferedValue + "\" "
                  + "WHERE id=" + Food_Id_b[i]);


            } catch (SQLException e) {
              System.err.println(e);
            }
          }
          if (Food_Id_d.length > tableModel3.getRowCount()) {

            for (int i = tableModel3.getRowCount(); i < Food_Id_d.length; i++) {
              //delete the rest
```

```
            try {
               Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
               //getConnection();
               Statement stmt = (Statement) con.createStatement();

               int rs = stmt.executeUpdate("DELETE FROM nutrient_data.nutrition_plans_food "
                     + "WHERE id=" + Food_Id_d[i] + ";");

            } catch (SQLException e) {
               System.err.println(e);
            }
         }
      } else if (Food_Id_d.length < tableModel3.getRowCount()) {
         for (int rows = Food_Id_d.length; rows < tableModel3.getRowCount(); rows++) {
            //insert the rest
            Food = tableModel3.getValueAt(rows, 0).toString();
            Amount = tableModel3.getValueAt(rows, 1).toString();
            Carbs = tableModel3.getValueAt(rows, 2).toString();
            Fat = tableModel3.getValueAt(rows, 3).toString();
            Protein = tableModel3.getValueAt(rows, 4).toString();
            Calories = tableModel3.getValueAt(rows, 5).toString();
            PreferedValue=tableModel3.getValueAt(rows, 6).toString();
            Meal = "Dinner";

            try {
               Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
               //getConnection();
               Statement stmt = (Statement) con.createStatement();

             int rs = stmt.executeUpdate("insert into `nutrient_data`.`nutrition_plans_food` (
`Nutrition_Plans_id`,`Food`, `Amount`, `Carbohydrate`, `Fat`, `Protein`, `Calories`,`Prefered`, `Meal`)
"
```

```
                + "values((select idnutrition_plans from nutrient_data.nutrition_plans where date_created=("
                + "select max(date_created) from nutrient_data.nutrition_plans where user_id=" + ID + ")), \"" + Food + "\",\"" + Amount + "\",\"" + Carbs + "\",\"" + Fat + "\", \"" + Protein + "\", \"" + Calories + "\", \"" + PreferedValue + "\", \"" + Meal + "\");");


            } catch (SQLException e) {
               System.err.println(e);
            }

          }

        }
        DefaultTableModel tableModel4 = (DefaultTableModel) jTable4.getModel();

        System.out.println("update Breakfast Table size: " + Food_Id_s.length + " table rows: " + tableModel4.getRowCount());

        for (int i = 0, rows = 0; i < Food_Id_s.length && rows < tableModel4.getRowCount(); i++, rows++) {

            Food = tableModel4.getValueAt(rows, 0).toString();
            Amount = tableModel4.getValueAt(rows, 1).toString();
            Carbs = tableModel4.getValueAt(rows, 2).toString();
            Fat = tableModel4.getValueAt(rows, 3).toString();
            Protein = tableModel4.getValueAt(rows, 4).toString();
            Calories = tableModel4.getValueAt(rows, 5).toString();
            PreferedValue=tableModel4.getValueAt(rows, 6).toString();
            try {

              Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
              Statement stmt = (Statement) con.createStatement();
```

```
            int rs = stmt.executeUpdate("UPDATE nutrient_data.nutrition_plans_food "
                    + "SET Food=\"" + Food + "\","
                    + " Amount=\"" + Amount + "\","
                    + " Carbohydrate=\"" + Carbs + "\","
                    + " Fat=\"" + Fat + "\","
                    + " Protein=\"" + Protein + "\","
                    + " Calories=\"" + Calories + "\","
                    + " Prefered=\"" + PreferedValue + "\" "
                    + "WHERE id=" + Food_Id_b[i]);
            } catch (SQLException e) {
                System.err.println(e);
            }
        }
        if (Food_Id_s.length > tableModel4.getRowCount()) {

            for (int i = tableModel4.getRowCount(); i < Food_Id_s.length; i++) {
                //delete the rest
                try {
                    Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

                    Statement stmt = (Statement) con.createStatement();

                    int rs = stmt.executeUpdate("DELETE FROM nutrient_data.nutrition_plans_food "
                            + "WHERE id=" + Food_Id_s[i] + ";");

                } catch (SQLException e) {
                    System.err.println(e);
                }
            }
        } else if (Food_Id_s.length < tableModel4.getRowCount()) {

            for (int rows = Food_Id_s.length; rows < tableModel4.getRowCount(); rows++) {
```

```
            //insert the rest
            Food = tableModel4.getValueAt(rows, 0).toString();
            Amount = tableModel4.getValueAt(rows, 1).toString();
            Carbs = tableModel4.getValueAt(rows, 2).toString();
            Fat = tableModel4.getValueAt(rows, 3).toString();
            Protein = tableModel4.getValueAt(rows, 4).toString();
            Calories = tableModel4.getValueAt(rows, 5).toString();
            PreferedValue=tableModel4.getValueAt(rows, 6).toString();
            Meal = "Snacks";

            try {
              Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
              //getConnection();
              Statement stmt = (Statement) con.createStatement();

             int rs = stmt.executeUpdate("insert into `nutrient_data`.`nutrition_plans_food` (
`Nutrition_Plans_id`,`Food`, `Amount`, `Carbohydrate`, `Fat`, `Protein`, `Calories`,`Prefered`, `Meal`)
"
                + "values((select idnutrition_plans from nutrient_data.nutrition_plans where
date_created=("
                + "select max(date_created) from nutrient_data.nutrition_plans where user_id=" +
ID + ")), \"" + Food + "\",\"" + Amount + "\",\"" + Carbs + "\",\"" + Fat + "\", \"" + Protein + "\", \"" +
Calories + "\", \"" + PreferedValue + "\", \"" + Meal + "\");");

            } catch (SQLException e) {
              System.err.println(e);
            }
          }
        }
      }
      munutr.dispose();
```

Δ.6 Κώδικας Ανανέωσης Προφίλ Χρήστη

```
String[] Name = jTextField23.getText().split(" ");
    String[] a = jTextField24.getText().split(" ");
    String g = null;
    if (jRadioButton92.isSelected()) {
      g = "M";
    } else if (jRadioButton93.isSelected()) {
      g = "F";
    }

    Double bmr = calculateBMR(Integer.parseInt(a[0]), Double.parseDouble(jTextField26.getText()),
Double.parseDouble(jTextField27.getText()));
    double cal = 0.0;
    if (jRadioButton84.isSelected()) {
      User_Exercise = "Sedentary (Little or No Exercise)";
      cal = bmr * 1.2;
    } else if (jRadioButton85.isSelected()) {
      User_Exercise = "Lightly Active (1-3 Days a week)";
      cal = bmr * 1.375;
    } else if (jRadioButton86.isSelected()) {
      User_Exercise = "Moderatetely Active (3-5 Days a week)";
      cal = bmr * 1.55;
    } else if (jRadioButton87.isSelected()) {
      User_Exercise = "Active (6-7 Days a week)";
      cal = bmr * 1.725;
    } else if (jRadioButton88.isSelected()) {
      User_Exercise = "Extra Active (Very Hard Exercise or 2x Training)";
      cal = bmr * 1.9;
    }

    String goal = null;
    Double sug_cal = 0.0;
    if (jRadioButton89.isSelected()) {
      goal = "Loose";
```

```
            sug_cal = cal - 500;
        } else if (jRadioButton90.isSelected()) {
            goal = "Maintain";
            sug_cal = cal;
        } else if (jRadioButton91.isSelected()) {
            goal = "Gain";
            sug_cal = cal + 500;
        }
        System.out.println("suggested calories: " + sug_cal);
        //calculate new Goals
        try {
          Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data",
"root", "MyNewPass");
            //getConnection();
            Statement stmt = (Statement) con.createStatement();

            int rs = stmt.executeUpdate("UPDATE nutrient_data.user_details "
                    + "SET Username='" + jTextField25.getText() + "',"
                    + " Name=" + "'" + Name[0] + "',"
                    + " Surname=" + "'" + Name[1] + "',"
                    + " Age=" + a[0] + ","
                    + " Gender=" + "'" + g + "',"
                    + " Weight=" + jTextField27.getText() + ","
                    + " Height=" + jTextField26.getText() + ","
                    + " Target_Weight=" + jTextField28.getText() + ","
                    + " Goal='" + goal + "', "
                    + " BMR=" + bmr + ","
                    + " Suggested_Calories=" + sug_cal + ","
                    + " Exercise=" + "'" + User_Exercise + "'"
                    + "WHERE id_user=" + ID);

        } catch (SQLException e) {
            System.err.println(e);
        }
```

```
//update profile
AGE=Integer.parseInt(a[0]);
 jLabel60.setText(Name[0] + " " + Name[1]);

jLabel61.setText(Integer.toString(AGE) + " years");
if (gender.equals("F")) {
   jLabel62.setText("Female");
} else {
   jLabel62.setText("Male");
}

jLabel65.setText(jTextField26.getText());
jLabel66.setText(jTextField27.getText() );
jLabel156.setText(Double.toString(sug_cal));
jLabel152.setText(User_Exercise);
   jLabel68.setText(goal);

jLabel69.setText(jTextField28.getText());
jInternalFrame14.setVisible(false);
jInternalFrame8.setVisible(true);
```

Δ.7 Κώδικας Προβολής Διατροφών Χρήστη

```
jButton41.setEnabled(false);
jButton42.setEnabled(false);

flag = 4;
if (username == null && password == null) {
   jInternalFrame9.setVisible(true);
} else {

   DefaultTableModel tableModel = (DefaultTableModel) jTable5.getModel();
   jTable5.setRowHeight(30);

   if (tableModel.getRowCount() == 0) {
```

```
            String []prefered_Name=new String[100];
           int count = 0;
            System.out.println("ID: " + ID);
            try {
               Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
               //getConnection();
               Statement stmt = (Statement) con.createStatement();

               ResultSet rs = stmt.executeQuery("select
idnutrition_plans,Nutrition_Date,PreferedId,PreferedName,Prefered_Limit from
nutrient_data.nutrition_plans where user_id=" + ID + ";");
               while (rs.next()) {
                  Date[count] = rs.getString("Nutrition_Date");
                  idnutritions[count] = rs.getInt("idnutrition_plans");
                  Prefered_Limit[count] = rs.getDouble("Prefered_Limit");
                  prefered_id[count] = rs.getInt("PreferedId");
                  prefered_Name[count]=rs.getString("PreferedName");
                  count++;
               }
               System.out.println(Date[0]);

            } catch (SQLException e) {
               System.err.println(e);
            }

            if (count == 0) {
               jInternalFrame16.setVisible(true);
            } else {
               String[] data = new String[3];
               for (int i = 0; i < count; i++) {
                  //data[2]=
                  data[0] = Date[i];
                  data[1] = prefered_Name[i];
```

```
                data[2] = "";
                tableModel.addRow(data);
                jTable5.setModel(tableModel);
              }
              jInternalFrame10.setVisible(true);
            }
          }
        }
```

Δ.8 Κώδικας Εισαγωγής Νέου Φαγητού στην Βάση

```
 if (!jTextField8.getText().equals("") && !jTextField16.getText().equals("") && !jTextField17.getText().equals("") && !jTextField18.getText().equals("") && !jTextField19.getText().equals("") && !jTextField20.getText().equals("")) {

        String FoodName_short = jTextField8.getText();
        String FoodName_long = jTextField16.getText();
        Double cal = Double.parseDouble(jTextField17.getText());
        Double carb = Double.parseDouble(jTextField18.getText());
        Double fat = Double.parseDouble(jTextField19.getText());
        Double prot = Double.parseDouble(jTextField20.getText());

          String fdg_no = q1.returnFoodGroup(jComboBox7.getSelectedIndex());
        Double fibre;
        Double sod;
        if (jTextField22.getText().equals("")) {
          sod = 0.0;
        } else {
          sod = Double.parseDouble(jTextField22.getText());
        }
        if (jTextField21.getText().equals("")) {
          fibre = 0.0;
        } else {
          fibre = Double.parseDouble(jTextField21.getText());
        }
```

```
        Integer fcd = 0;
        //find max ndb_no in db
        try {
           Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
           Statement stmt = (Statement) con.createStatement();

           ResultSet rs = stmt.executeQuery("select max(`ndb_no`) from `nutrient_data`.`food_description`;");

           int count = 0;
           while (rs.next()) {
              //pairnw ta dedomena me vasi to onoma tis stilis

              String s = rs.getString("max(`ndb_no`)");
              System.out.println("max food code: " + s);
              fcd = Integer.parseInt(s);

              count++;
           }

        } catch (SQLException e) {
           System.err.println(e);
        }
        fcd = fcd + 1;

        try {
           Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
           //getConnection();
           Statement stmt = (Statement) con.createStatement();
```

```
            int rs = stmt.executeUpdate("insert into `nutrient_data`.`food_description` (`ndb_no`, `fdgrp_cd`, `long_desc`, `shrt_desc`) "
                    + "values(\"" + fcd + "\",\"" + fdg_no + "\",\"" + FoodName_long + "\", \"" + FoodName_short + "\");");

        } catch (SQLException e) {
            System.err.println(e);
        }

        //SAVE IN NUTRIENT DATA
       try {
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

            Statement stmt = (Statement) con.createStatement();

            int rs = stmt.executeUpdate("insert into `nutrient_data`.`nutrient_data`(NDB_No, Nutr_No, Nutr_Val, Num_Data_Pts, Src_Cd) "
                    + "values(\"" + fcd + "\",\"208\"," + cal + ",0, \"91\"),(\"" + fcd + "\",\"205\"," + carb + ",0, \"91\"),"
                    + "(\"" + fcd + "\",\"204\"," + fat + ",0, \"91\"),(\"" + fcd + "\",\"203\"," + prot + ",0, \"91\"),"
                    + "(\"" + fcd + "\",\"291\"," + fibre + ",0, \"91\"),(\"" + fcd + "\",\"307\"," + sod + ",0, \"91\");");

        } catch (SQLException e) {
            System.err.println(e);
        }

//          //SAVE IN WEIGHT FILE
        try {
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
```

```
            Statement stmt = (Statement) con.createStatement();

            int rs = stmt.executeUpdate("insert into `nutrient_data`.`weight_file`(NDB_No, Seq,
Amount, Msre_Desc, GM_Wgt) "
                    + "values(\"" + fcd + "\",\"1\", 100.0, \"grams\", " + 100 + ");");

        } catch (SQLException e) {
            System.err.println(e);
        }

        jInternalFrame12.setVisible(false);

    }
    else {

        if(jTextField8.getText().equals("") )
        {
            jLabel96.setVisible(true);
        }
        else if(jTextField16.getText().equals("") )
        {

            jLabel95.setVisible(true);
        }
        else if (jTextField17.getText().equals(""))
        {
            jLabel85.setVisible(true);
        }
        else if(!jTextField18.getText().equals(""))
        {
            jLabel89.setVisible(true);
        }
        else if( !jTextField19.getText().equals("") )
        {
```

```
            jLabel92.setVisible(true);
        }
        else if(!jTextField20.getText().equals(""))
        {
            jLabel93.setVisible(true);
        }
    }
```

Δ.9 Κώδικας Αλλαγής Password

```
String oldPass = jPasswordField4.getText();
    String newPass1 = jPasswordField5.getText();
    String newPass2 = jPasswordField6.getText();

    if (oldPass.equals(password)) {

        if (newPass1.equals(newPass2)) {
            try {
                Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
                //getConnection();
                Statement stmt = (Statement) con.createStatement();

                int rs = stmt.executeUpdate("UPDATE nutrient_data.user_details "
                    + "SET Password='" + newPass1 + "' "
                    + "WHERE id_user=" + ID);

            } catch (SQLException e) {
                System.err.println(e);
            }
            jInternalFrame17.setVisible(false);
            jLabel73.setVisible(true);
        } else {
            jPasswordField5.setText("");
            jPasswordField6.setText("");
```

```
            jLabel164.setText("The New Passwords do not match!");
        }

    } else {
        jPasswordField4.setText("");
        jLabel164.setText("Please enter your current password correctly!");
    }
    jLabel164.setText("");
    jPasswordField4.setText("");
    jPasswordField5.setText("");
    jPasswordField6.setText("");
```

Δ.10 Κώδικας Εισαγωγής Νέου Φαγητού από διάφορα Υλικά στην Βάση

```
if(jList2.getModel().getSize()==0)
    {
    jLabel81.setVisible(true);
    }
    else
        if(jTextField1.getText().length()==0 && jTextField2.getText().length()==0)
    {
        jLabel83.setVisible(true);
        jLabel87.setVisible(true);
    }
    else if(jTextField1.getText().length()==0)
    {
        jLabel83.setVisible(true);
    }
   else if(jTextField2.getText().length()==0)
    {
        jLabel87.setVisible(true);
    }
   else
    {
```

```
        Integer fcd = 0;
        //find max ndb_no in db
        try {
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
            //getConnection();
            Statement stmt = (Statement) con.createStatement();

            ResultSet rs = stmt.executeQuery("select max(`ndb_no`) from `nutrient_data`.`food_description`;");

            int count = 0;
            while (rs.next()) {
                //pairnw ta dedomena me vasi to onoma tis stilis

                String s = rs.getString("max(`ndb_no`)");
                System.out.println("max food code: " + s);
                fcd = Integer.parseInt(s);

                count++;
            }

        } catch (SQLException e) {
            System.err.println(e);
        }
        fcd = fcd + 1;

        //save recipe in database

        //SAVE IN FOOD DESCRIPTION
        String ndb_no = Integer.toString(fcd);

        String comname;
```

```
String long_desc = null, short_desc = null;
if (!jTextField1.equals("")) {

   comname = jTextField1.getText();
   if (!jTextArea3.equals("")) {
      short_desc = jTextField1.getText().toUpperCase();

   } else {

      short_desc = jTextArea3.getText().toUpperCase();
   }

   if (!jTextArea2.equals("")) {
      long_desc = jTextField1.getText();

   } else {

      long_desc = jTextArea2.getText();
   }
}
String fdgrp_cd = q1.returnFoodGroup(jComboBox8.getSelectedIndex());
try {
   Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
   //getConnection();
   Statement stmt = (Statement) con.createStatement();

   int rs = stmt.executeUpdate("insert into `nutrient_data`.`food_description` (`ndb_no`,
`fdgrp_cd`, `long_desc`, `shrt_desc`) "
         + "values(\"" + ndb_no + "\",\"" + fdgrp_cd + "\",\"" + long_desc + "\", \"" + short_desc +
"\");");

} catch (SQLException e) {
   System.err.println(e);
```

```
    }
    System.out.println("Ingredients Length: " + IngrCount);

    if (IngrCount > 0) {
      Double CalPer100gr = TotalCal / IngrCount;

      Double CarbsPer100gr = TotalCarbs / IngrCount;
      Double FatPer100gr = TotalFat / IngrCount;
      Double ProteinPer100gr = TotalProtein / IngrCount;
      System.out.println("Values Per 100gr: (Calories): " + CalPer100gr + " (Carbs): " +
CarbsPer100gr + " (Fat): " + FatPer100gr + " (Protein): " + ProteinPer100gr);

      try {
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

        Statement stmt = (Statement) con.createStatement();

        int rs = stmt.executeUpdate("insert into `nutrient_data`.`nutrient_data`(NDB_No, Nutr_No,
Nutr_Val, Num_Data_Pts, Src_Cd) "
            + "values(\"" + fcd + "\",\"208\"," + CalPer100gr + ",0, \"91\"),(\"" + fcd + "\",\"205\","
+ CarbsPer100gr + ",0, \"91\"),"
            + "(\"" + fcd + "\",\"204\"," + FatPer100gr + ",0, \"91\"),(\"" + fcd + "\",\"203\"," +
ProteinPer100gr + ",0, \"91\");");

      } catch (SQLException e) {
        System.err.println(e);
      }
    }

    //SAVE IN WEIGHT FILE
    try {
      Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
```

```
        Statement stmt = (Statement) con.createStatement();

        int rs = stmt.executeUpdate("insert into `nutrient_data`.`weight_file`(NDB_No, Seq, Amount,
Msre_Desc, GM_Wgt) "
                + "values(\"" + ndb_no + "\",\"1\", 1.0, \"Serving\", " + WeightPerSer + ");");

      } catch (SQLException e) {
        System.err.println(e);
      }
      jInternalFrame2.setVisible(false);

      }
```

Δ.11 Κώδικας Προβολής Θρεπτικών Συστατικών Φαγητού

```
  if (jTextField31.getText().length()!=0) {
        Double number = Double.parseDouble(jTextField31.getText());
        System.out.println("mpikeeee!!! " + db_no);
        try {
          Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
          //getConnection();
          Statement stmt = (Statement) con.createStatement();

          ResultSet rs = stmt.executeQuery("select `Nutr_No`, `Nutr_Val` from
`nutrient_data`.`nutrient_data` where `nutrient_data`.`NDB_No`=" + Integer.toString(db_no) + ";");

          int count = 0;
          while (rs.next()) {
            //pairnw ta dedomena me vasi to onoma tis stilis

            //String s = rs.getString("shrt_desc");
            int i = rs.getInt("Nutr_No");
            //String md=rs.getString("Msre_Desc");
            NutrCode[count] = i;
```

```
            Double w = rs.getDouble("Nutr_Val");
            // short_desc=s;
            // float n = rs.getFloat("Age");
            System.out.println(NutrCode[count] + " with weight: " + w);
            Nutrients[count] = w;
            //measures[count]=md;
            count++;
          }
        } catch (SQLException e) {
          System.err.println(e);
        }
        //  double amount=Double.parseDouble(jTextField11.getText());
        int selected = jComboBox10.getSelectedIndex();
System.out.println(selected);

        System.out.println("number: " + number + " weight " + weights[selected] + " measure " +
measures[selected] + " ");

         double weight = (weights[selected] * Double.parseDouble(jTextField31.getText())) /
Amounts[selected];

        for (int k = 0; NutrCode[k] != null; k++) {
        //  System.out.println(NutrCode[k]);
        //calories
        if (NutrCode[k].equals(208)) {

          double temp1 = (weight * Nutrients[k]) / 100;
          jLabel191.setText(Double.toString(temp1));

        }
        //carbs
        if (NutrCode[k].equals(205)) {
          double temp2 = (weight * Nutrients[k]) / 100;
```

```
            jLabel192.setText(Double.toString(RoundedValue(temp2)));

        }
        //fat
        if (NutrCode[k].equals(204)) {
            double temp3 = (weight * Nutrients[k]) / 100;
            jLabel193.setText(Double.toString(RoundedValue(temp3)));
        }
        //protein
        if (NutrCode[k].equals(203)) {
            double temp4 = (weight * Nutrients[k]) / 100;
             jLabel194.setText(Double.toString(RoundedValue(temp4)));
        }
        //fibre
        if (NutrCode[k].equals(291)) {
            double temp5 = (weight * Nutrients[k]) / 100;
             jLabel195.setText(Double.toString(RoundedValue(temp5)));
        }
        //sodium
        if (NutrCode[k].equals(307)) {
            double temp6 = (weight * Nutrients[k]) / 100;
             jLabel196.setText(Double.toString(RoundedValue(temp6)));
        }
}
    } else {

        jLabel197.setText("Please Choose an amount and Measurement to calculate.");

    }
```

Δ.12 Κώδικας Αρχικοποίησης Προτιμήσεων Χρήστη για Αυτόματη Διατροφή

```
    int minmax=0;
        if(jRadioButton94.isSelected())
    {
        minmax=0;
```

```
}
else if(jRadioButton95.isSelected())
{
   minmax=1;
}
int sel = jComboBox9.getSelectedIndex();
if (sel == 0)//none
{
   preferenceNutrient = 0;
   nutrientValue = 0;
} else {

   if(!jTextField29.getText().equals("")){

   nutrientValue = Double.parseDouble(jTextField29.getText());
   nutrValue = (int)(nutrientValue * 1000);
   if (sel == 1)//sugars
   {
      minmax=1;
      preferenceNutrient = 269;
   } else if (sel == 2)//calcium
   {

      preferenceNutrient = 301;
   } else if (sel == 3)//iron
   {
      preferenceNutrient = 303;
   } else if (sel == 4)//thiamin
   {
      minmax=0;
      preferenceNutrient = 404;
   } else if (sel == 5)//cholesterol
   {
      minmax=1;
```

```
            preferenceNutrient = 601;
        } else if (sel == 6)//vitamin C
        {
            preferenceNutrient = 401;
        } else if (sel == 7)//vitamin D
        {
            preferenceNutrient = 328;

        }

        }
        else
        {

            jLabel81.setText("* Please Enter amount for Nutrient to Control!");

        }

    }

    ///breakfast panel
    if ((jRadioButton24.isSelected() || jRadioButton25.isSelected() || jRadioButton26.isSelected())
&& (jRadioButton14.isSelected() || jRadioButton15.isSelected() || jRadioButton23.isSelected()))
    {
        //cereal
        if (jRadioButton24.isSelected()) {
            b1_choice = 1;
            } //bread
        else if (jRadioButton25.isSelected()) {
            b1_choice = 2;
         } //cakes etc.
        else if (jRadioButton26.isSelected()) {
            b1_choice = 3;
```

```
        }
        else {
         b1_choice = 0;
        }

        //fruit
        if (jRadioButton28.isSelected()) {
           b3_choice = 1;
             } else if (jRadioButton29.isSelected()) {
           b3_choice = 0;
        }
      B3QueryNutr=query.getB3QueryNutr(b3_choice, prefNutrient);
      B3QueryWeight=query.getB3QueryWeight(b3_choice);

      //sausage
        if (jRadioButton31.isSelected()) {
           b4_choice = 1;

}
        else if(jRadioButton96.isSelected())
        {
         b4_choice = 0;
        }
        B4QueryNutr=query.getB4QueryNutr(b4_choice, prefNutrient);
        B4QueryWeight=query.getB4QueryWeight(b4_choice);
        //eggs
        if (jRadioButton32.isSelected()) {
             b5_choice = 1;
                }
        else if(jRadioButton97.isSelected())
        {
           b5_choice = 0;
        }
        B5QueryNutr=query.getB5QueryNutr(b5_choice, prefNutrient);
```

```
   B5QueryWeight=query.getB5QueryWeight(b5_choice);
    //dairy milk
   if (jRadioButton14.isSelected()) {
      b2_choice = 1;

   } //cheese
   else if (jRadioButton15.isSelected()) {
      b2_choice = 2;

   } //yogurt
   else if (jRadioButton23.isSelected()) {
      b2_choice = 3;

   }

   B2QueryNutr=query.getB2QueryNutr(b2_choice, prefNutrient);
   B2QueryWeight=query.getB2QueryWeight(b2_choice);

B1QueryNutr=query.getB1QueryNutr(b1_choice, prefNutrient);
B1QueryWeight=query.getB1QueryWeight(b1_choice);

} else {
   //Error Messages
   jLabel81.setText("* Please make a choice at Least from the Grain and Dairy Groups!");
}
//lunch  panel
if (jRadioButton16.isSelected() || jRadioButton17.isSelected() || jRadioButton18.isSelected()) {
   //if nutrition
   if (jRadioButton18.isSelected() )
   {
      //nutrition--meat
    if (jRadioButton98.isSelected()) {
         l1_choice = 0;
         }
```

```
        //chicken
         else if (jRadioButton13.isSelected()) {
            l1_choice = 7;

         } //pork
         else if (jRadioButton30.isSelected()) {
            l1_choice = 8;

         } //beef
         else if (jRadioButton33.isSelected()) {
            l1_choice = 9;
        } //lamb
         else if (jRadioButton34.isSelected()) {
            l1_choice = 10;

         }
         else if (jRadioButton4.isSelected()) {
            l1_choice = 11;

}
         if (jRadioButton100.isSelected()) {
             l2_choice = 0;
         }
         //corn or potatoes
         if (jRadioButton35.isSelected()) {
            l2_choice = 1;

         } //Grains & Pasta
         else if (jRadioButton36.isSelected()) {
            l2_choice = 2;

         } //bread
         else if (jRadioButton38.isSelected()) {
            l2_choice = 3;
```

```
}
  L2QueryNutr=query.getL2QueryNutr(l2_choice, prefNutrient);
   L2QueryWeight=query.getL2QueryWeight(l2_choice);

   //legumes
if (jRadioButton39.isSelected()) {
   l3_choice = 1;

} else if (jRadioButton40.isSelected()) {
   l3_choice = 0;
}
L3QueryNutr=query.getL3QueryNutr(l3_choice, prefNutrient);
   L3QueryWeight=query.getL3QueryWeight(l3_choice);
//vegetables
if (jRadioButton37.isSelected()) {
   l4_choice = 1;

} else if (jRadioButton54.isSelected()) {
   l4_choice = 0;
}
L4QueryNutr=query.getL4QueryNutr(l4_choice, prefNutrient);
   L4QueryWeight=query.getL4QueryWeight(l4_choice);

   //dairy
if (jRadioButton99.isSelected()) {
   l5_choice = 0;

}//milk
else if (jRadioButton11.isSelected()) {
   l5_choice = 1;
}
else if (jRadioButton12.isSelected()) {
   l5_choice = 2;
```

```
            }
            else if (jRadioButton27.isSelected()) {
               l5_choice = 3;
            }
            L5QueryNutr=query.getB2QueryNutr(l5_choice, prefNutrient);
               L5QueryWeight=query.getB2QueryWeight(l5_choice);
         } //cuisine
         else if (jRadioButton16.isSelected() && (jRadioButton47.isSelected() ||
jRadioButton57.isSelected() || jRadioButton59.isSelected() || jRadioButton58.isSelected() ||
jRadioButton60.isSelected())) {
            //greek
            if (jRadioButton47.isSelected()) {
               l1_choice = 1;



            } //italian
            else if (jRadioButton57.isSelected()) {
               l1_choice = 2;

            } //french
            else if (jRadioButton58.isSelected()) {
               l1_choice = 3;
          } //japanese
            else if (jRadioButton59.isSelected()) {
               l1_choice = 4;
   } //chinese
            else if (jRadioButton60.isSelected()) {
               l1_choice = 5;

            }
         } //restaurant
         else if (jRadioButton17.isSelected()) {
            l1_choice = 6;
```

```
  } else {
            jLabel81.setText("* Please make a choice in every Category in Cuisine!");
          }
      L1QueryNutr=query.getL1QueryNutr(l1_choice, prefNutrient);
      L1QueryWeight=query.getL1QueryWeight(l1_choice);

      } else {
        jLabel81.setText("* Please make a choice in every Category for Lunch!");
      }
      //dinner panel
      if (jRadioButton19.isSelected() || jRadioButton20.isSelected() || jRadioButton21.isSelected() ||
jRadioButton22.isSelected()) {
        //cuisine
        if (jRadioButton19.isSelected() && (jRadioButton61.isSelected() ||
jRadioButton62.isSelected() || jRadioButton63.isSelected() || jRadioButton64.isSelected() ||
jRadioButton65.isSelected()))
        {
          //greek
          if (jRadioButton61.isSelected()) {
            d1_choice = 1;
         } //italian
          else if (jRadioButton62.isSelected()) {
            d1_choice = 2;

          } //french
          else if (jRadioButton63.isSelected()) {
            d1_choice = 3;

          } //japanese
          else if (jRadioButton64.isSelected()) {
            d1_choice = 4;

          } //chinese
          else if (jRadioButton65.isSelected()) {
```

```
            d1_choice = 5;

        }
    } //restaurant food
    else if (jRadioButton20.isSelected()) {
        d1_choice = 6;

} else if (jRadioButton21.isSelected() ) {

        if (jRadioButton2.isSelected()) {
            d1_choice = 0;

        }
        //chicken
        else if (jRadioButton41.isSelected()) {
            d1_choice = 7;

        } //pork
        else if (jRadioButton42.isSelected()) {
            d1_choice = 8;


        } //beef
        else if (jRadioButton43.isSelected()) {
            d1_choice = 9;

        } //lamb
        else if (jRadioButton44.isSelected()) {
            d1_choice = 10;

       }//fish
         else if (jRadioButton5.isSelected()) {
            d1_choice = 11;
```

```
 }
//dairy
    if (jRadioButton1.isSelected()) {
       d2_choice = 0;
     }//milk
    else if (jRadioButton51.isSelected()) {
       d2_choice = 1;
    }
    else if (jRadioButton52.isSelected()) {
       d2_choice = 2;

           }
    else if (jRadioButton53.isSelected()) {
       d2_choice = 3;
    }
     D2QueryNutr=query.getB2QueryNutr(d2_choice, prefNutrient);
       D2QueryWeight=query.getB2QueryWeight(d2_choice);

 if (jRadioButton3.isSelected()) {
  d3_choice = 0;
}
 //Corn or Potato
 else if (jRadioButton45.isSelected()) {
    d3_choice = 1;
  }
 //Grains or Pasta
 else if (jRadioButton46.isSelected()) {
    d3_choice = 2;

 } //Bread
 else if (jRadioButton48.isSelected()) {
    d3_choice = 3;

 }
```

```
        else if(jRadioButton22.isSelected())
        {
           d3_choice = 4;
        }

D3QueryNutr=query.getL2QueryNutr(d3_choice, prefNutrient);
D3QueryWeight=query.getL2QueryWeight(d3_choice);

//legumes
        if (jRadioButton49.isSelected()) {
           d4_choice = 1;
        } else {
           d4_choice = 0;
        }
D4QueryNutr=query.getL3QueryNutr(d4_choice, prefNutrient);
D4QueryWeight=query.getL3QueryWeight(d4_choice);
        //vegetables
        if (jRadioButton55.isSelected()) {
           d5_choice = 1;

        } else {
           d5_choice = 0;
        }

D5QueryNutr=query.getL4QueryNutr(d5_choice, prefNutrient);
D5QueryWeight=query.getL4QueryWeight(d5_choice);
    } //cereal

    D1QueryNutr=query.getL1QueryNutr(d1_choice, prefNutrient);
    D1QueryWeight=query.getL1QueryWeight(d1_choice);

}
```

```
    //Snack Panel
    if ((jRadioButton66.isSelected() || jRadioButton67.isSelected() || jRadioButton68.isSelected() ||
jRadioButton69.isSelected() || jRadioButton70.isSelected() || jRadioButton76.isSelected())
        && (jRadioButton71.isSelected() || jRadioButton72.isSelected() ||
jRadioButton73.isSelected() || jRadioButton74.isSelected() || jRadioButton75.isSelected() ||
jRadioButton77.isSelected())
        && (jRadioButton78.isSelected() || jRadioButton79.isSelected() ||
jRadioButton80.isSelected() || jRadioButton81.isSelected() || jRadioButton82.isSelected() ||
jRadioButton83.isSelected()))
    {
      jLabel81.setText(" ");
      //Snack 1

      //Snack Bars
      if (jRadioButton66.isSelected()) {
        s1_choice = 1;



      } //Sweets
      else if (jRadioButton67.isSelected()) {
        s1_choice = 2;

      } //Fruits
      else if (jRadioButton68.isSelected()) {
        s1_choice = 3;



      } //Beverages
      else if (jRadioButton69.isSelected()) {
        s1_choice = 4;



      } //Cakes, Pies
      else if (jRadioButton70.isSelected()) {
```

```
   s1_choice = 5;
  } //Nuts
else if (jRadioButton76.isSelected()) {
   s1_choice = 6;
}
S1QueryNutr=query.getSQueryNutr(s1_choice, prefNutrient);
S1QueryWeight=query.getSQueryWeight(s1_choice);

//Snack 2
//snack bar
if (jRadioButton71.isSelected()) {
   s2_choice = 1;
   } //sweets
else if (jRadioButton72.isSelected()) {

   s2_choice = 2;

} //fruits
else if (jRadioButton73.isSelected()) {

   s2_choice = 3;

} //beverages
else if (jRadioButton74.isSelected()) {
   s2_choice = 4;

} //cakes
else if (jRadioButton75.isSelected()) {
   s2_choice = 5;

} //nuts
else if (jRadioButton77.isSelected()) {
   s2_choice = 6;
```

```
    }
    S2QueryNutr=query.getSQueryNutr(s2_choice, prefNutrient);
    S2QueryWeight=query.getSQueryWeight(s2_choice);
    //Snack 3
    //Snack Bars
    if (jRadioButton78.isSelected()) {
       s3_choice = 1;


    } //sweets
    else if (jRadioButton79.isSelected()) {
       s3_choice = 2;

    } //fruits
    else if (jRadioButton80.isSelected()) {
       s3_choice = 3;

    } //beverages
    else if (jRadioButton81.isSelected()) {
       s3_choice = 4;

    } //cakes, pies
    else if (jRadioButton82.isSelected()) {
       s3_choice = 5;

    } //nuts
    else if (jRadioButton83.isSelected()) {
       s3_choice = 6;

    }
S3QueryNutr=query.getSQueryNutr(s3_choice, prefNutrient);
S3QueryWeight=query.getSQueryWeight(s3_choice);
```

```
QuickNutrition newQuickNutr = new QuickNutrition(username, password, b1_choice, b2_choice, b3_choice, b4_choice, b5_choice, l1_choice, l2_choice, l3_choice, l4_choice, l5_choice, d1_choice, d2_choice, d3_choice, d4_choice, d5_choice, s1_choice, s2_choice, s3_choice, B1QueryNutr, B1QueryWeight, B2QueryNutr, B2QueryWeight, B3QueryNutr, B3QueryWeight, B4QueryNutr, B4QueryWeight,
        B5QueryNutr, B5QueryWeight,
        L1QueryNutr, L1QueryWeight, L2QueryNutr, L2QueryWeight, L3QueryNutr, L3QueryWeight, L4QueryNutr, L4QueryWeight, L5QueryNutr, L5QueryWeight,
        D1QueryNutr, D1QueryWeight, D3QueryNutr, D3QueryWeight, D4QueryNutr, D4QueryWeight, D5QueryNutr, D5QueryWeight, D2QueryNutr, D2QueryWeight, S1QueryNutr, S1QueryWeight, S2QueryNutr, S2QueryWeight, S3QueryNutr, S3QueryWeight, preferenceNutrient, nutrientValue, minmax);

    String outputstr = newQuickNutr.ExecuteConstraintProgram(b1_choice, b2_choice, b3_choice, b4_choice,b5_choice, l1_choice, l2_choice, l3_choice, l4_choice,l5_choice, d1_choice, d2_choice, d3_choice, d4_choice, d5_choice, s1_choice, s2_choice, s3_choice, nutrValue);
    System.out.println("the out put is: " + outputstr);
    long start = System.currentTimeMillis();
    long end = start + 10*1000; // 60 seconds * 1000 ms/sec


    while (outputstr.equals("UNSATISFIABLE") && System.currentTimeMillis() < end) {
      outputstr = newQuickNutr.ExecuteConstraintProgram(b1_choice, b2_choice, b3_choice, b4_choice,b5_choice, l1_choice, l2_choice, l3_choice, l4_choice,l5_choice, d1_choice, d2_choice, d3_choice, d4_choice, d5_choice, s1_choice, s2_choice, s3_choice, nutrValue);
      System.out.println("the out put is: " + outputstr);

      if(outputstr==null)
      {
      break;
      }
    }

    if (!outputstr.equals("UNSATISFIABLE")) {
```

```
            Date date = new Date();

            String[][] foodIDs = newQuickNutr.getFoodIDs(outputstr);
            Integer[] amounts = newQuickNutr.getFoodAmount(outputstr);
           NutritionPlanFrame mynutritionplan=new NutritionPlanFrame(ID, BMR ,foodIDs,
amounts,prefNutrient, 0, preferenceNutrient,nutrientValue);
            mynutritionplan.setLocationRelativeTo(null);
                    mynutritionplan.setVisible(true);

        }
        else{
        System.out.println("No solution");
         AboutDialog msg=new AboutDialog(username,password,ID,BMR,Thiamin , VitaminD,
VitaminC, Calcium, Iron);
          msg.setVisible(true);
        }
      myquickframe.dispose();
      } else {

        jLabel81.setText("* Please make a choice in every Category!");
      }
```

Δ.13 Κώδικας Υπολογισμού Βέλτιστης Διατροφής

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package my.nutritionapp;

import choco.kernel.model.variables.integer.IntegerVariable;
import choco.kernel.model.variables.real.RealVariable;
import java.io.*;
import java.sql.*;
import java.util.ArrayList;
```

```
import java.util.Collections;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * Class for Constraint Programming
 * Using CHOCO Library
 *
 * @author Snowflake
 */
public class QuickNutrition {
//Variable Declaration
   //Personal info
   private int age=0;
   private String gender=null;
   private double nutr_value=0;
   private int nutr_code=0;
   int Prefered=0;

   //Combination of Foods

    private int EggCount=0;
     Double [][]EggAmount;
     Double [][]EggWeight;
     String [][]EggMeasures;

   private Integer[] Dairy1;
   int Dairy1Count=0;
     Double [][]DairyAmount;
     Double [][]DairyWeight;
     String [][]DairyMeasures;

     private Integer[] BCereal;
```

```
int B1Count=0;
String [] B1IDs;
Double [][]B1Amount;
Double [][]B1Weight;
String [][]B1Measures;


int B2Count=0;
String [] B2IDs;
Double [][]B2Amount;
Double [][]B2Weight;
String [][]B2Measures;

int B3Count=0;
String [] B3IDs;
Double [][]B3Amount;
Double [][]B3Weight;
String [][]B3Measures;

int B4Count=0;
String [] B4IDs;
Double [][]B4Amount;
Double [][]B4Weight;
String [][]B4Measures;

  int B5Count=0;
String [] B5IDs;
Double [][]B5Amount;
Double [][]B5Weight;
String [][]B5Measures;

int L1Count=0;
String [] L1IDs;
Double [][]L1Amount;
```

```
  Double [][]L1Weight;
  String [][]L1Measures;

int L2Count=0;
String [] L2IDs;
  Double [][]L2Amount;
  Double [][]L2Weight;
  String [][]L2Measures;

  int L3Count=0;
  String [] L3IDs;
  Double [][]L3Amount;
  Double [][]L3Weight;
  String [][]L3Measures;

  int L4Count=0;
  String [] L4IDs;
  Double [][]L4Amount;
  Double [][]L4Weight;
  String [][]L4Measures;

  int L5Count=0;
  String [] L5IDs;
  Double [][]L5Amount;
  Double [][]L5Weight;
  String [][]L5Measures;

  int D1Count=0;
  String [] D1IDs;
  Double [][]D1Amount;
  Double [][]D1Weight;
  String [][]D1Measures;
```

```
int D2Count=0;
String [] D2IDs;
Double [][]D2Amount;
Double [][]D2Weight;
String [][]D2Measures;

int D3Count=0;
String [] D3IDs;
Double [][]D3Amount;
Double [][]D3Weight;
String [][]D3Measures;


int D4Count=0;
String [] D4IDs;
Double [][]D4Amount;
Double [][]D4Weight;
String [][]D4Measures;

int D5Count=0;
String [] D5IDs;
Double [][]D5Amount;
Double [][]D5Weight;
String [][]D5Measures;

  int S1Count=0;
  String [] S1IDs;
Double [][]S1Amount;
Double [][]S1Weight;
String [][]S1Measures;

  int S2Count=0;
  String [] S2IDs;
Double [][]S2Amount;
```

```
 Double [][]S2Weight;
 String [][]S2Measures;

  int S3Count=0;
  String [] S3IDs;
 Double [][]S3Amount;
 Double [][]S3Weight;
 String [][]S3Measures;

String fileString=new String();
String constraintString=new String();
 String tempConstraint1=new String();
  String tempConstraint2=new String();
   String tempConstraint3=new String();
   String tempConstraint4=new String();
   String tempConstraint5=new String();
   String tempConstraint6=new String();
   String tempConstraint7=new String();
   String tempConstraint8=new String();
   String tempConstraint9=new String();
   String tempConstraint10=new String();
   String tempConstraint11=new String();
String programString=new String();
String outputString=new String();

private double Calories;
private double Carbohydrate;
private double Fat;
private double Protein;
private double Cholesterol;
private double Sodium;//Natrio
private double Potassium;//Kallio
private double Fiber;
private double VitaminA;
```

```
private double VitaminC;
private double Calcium;
private double Iron;
private double Magnesium;


int calories;
int carbs;
int fat;
int protein;
int sodium;//Natrio
int potassium;//Kallio
int vitaminA;
int  vitaminC;
int calcium;
int  iron;
int magnesium;

private String StringCalories;
private String StringCarbohydrate;
private String StringFat;
private String StringProtein;
private String StringCholesterol;
private String StringSodium;//Natrio
private String StringPotassium;//Kallio
private String StringFiber;
private String StringVitaminA;
private String StringVitaminC;
private String StringCalcium;
private String StringIron;
private String StringMagnesium;


 int [][]B1IntNutr;
```

```
int [][]B2IntNutr;
int [][]B3IntNutr;
int [][]B4IntNutr;
 int [][]B5IntNutr;
int [][]L1IntNutr;
int [][]L2IntNutr;
int [][]L3IntNutr;
int [][]L4IntNutr;
int [][]L5IntNutr;
int [][]D1IntNutr;
int [][]D2IntNutr;
int [][]D3IntNutr;
int [][]D4IntNutr;
int [][]D5IntNutr;
int [][]S1IntNutr;
int [][]S2IntNutr;
int [][]S3IntNutr;

int []B1randomIndex=new int[5];
int []B2randomIndex=new int[5];
int []B3randomIndex=new int[5];
int []B4randomIndex=new int[5];
int []B5randomIndex=new int[5];
int []L1randomIndex=new int[5];
int []L2randomIndex=new int[5];
int []L3randomIndex=new int[5];
int []L4randomIndex=new int[5];
int []L5randomIndex=new int[5];
int []D1randomIndex=new int[5];
int []D2randomIndex=new int[5];
int []D3randomIndex=new int[5];
int []D4randomIndex=new int[5];
int []D5randomIndex=new int[5];
int []S1randomIndex=new int[5];
```

```
 int []S2randomIndex=new int[5];
 int []S3randomIndex=new int[5];

private IntegerVariable[] BreakFast;
private IntegerVariable[] Lunch;
private IntegerVariable[] Dinner;
private IntegerVariable[] Snacks;

private IntegerVariable []Measure;


private RealVariable[] BreakFastAmount;
private RealVariable[] LunchAmount;
private RealVariable[] DinnerAmount;
private RealVariable[] SnacksAmount;
double []preferredTable=new double[16];
String ConstraintBuilder(String name)
{
   String constraint=null;


   return constraint;
}

String CreateDataString1(int Count, int IntNutr[][], String name, int min, int max, int prefered)
{
   String Calories=null;
   String Carbohydrate=null;
   String Fat=null;
   String Protein=null;
   String Preference=null;

   String file;
```

```
        file=name+"Count = "+Count+";\n"+"min"+name+"="+min+";\n"
              +"max"+name+"="+max+";\n";
        Calories=name+"Calories=[";
        Carbohydrate=name+"Carbohydrate=[";
        Fat=name+"Fat=[";
        Protein=name+"Protein=[";
//          if(prefered!=0)
        Preference=name+"Preference=[";
//          Potassium=name+"Potassium=[";//Kallio
//          VitaminA=name+"VitaminA=[";
//          VitaminC=name+"VitaminC=[";
//          Calcium=name+"Calcium=[";
//          Iron=name+"Iron=[";
//          Magnesium=name+"Magnesium=[";

        for(int i=0; i<Count; i++){

            if(i==Count-1){
               Calories+=IntNutr[i][0];
               Carbohydrate+=IntNutr[i][1];
               Fat+=IntNutr[i][2];
               Protein+=IntNutr[i][3];
//                 if(prefered!=0)
               Preference+=IntNutr[i][4];//Natrio
//                Potassium+=IntNutr[i][5];//Kallio
//                VitaminA+=IntNutr[i][6];
//                VitaminC+=IntNutr[i][7];
//                Calcium+=IntNutr[i][8];
//                Iron+=IntNutr[i][9];
//                Magnesium+=IntNutr[i][10];
             }
            else
            {
                Calories+=IntNutr[i][0]+",";
```

```
                Carbohydrate+=IntNutr[i][1]+",";
                Fat+=IntNutr[i][2]+",";
                Protein+=IntNutr[i][3]+",";
              //                  if(prefered!=0)
                Preference+=IntNutr[i][4]+",";//Natrio
//                  Potassium+=IntNutr[i][5]+",";//Kallio
//                  VitaminA+=IntNutr[i][6]+",";
//                  VitaminC+=IntNutr[i][7]+",";
//                  Calcium+=IntNutr[i][8]+",";
//                  Iron+=IntNutr[i][9]+",";
//                  Magnesium+=IntNutr[i][10]+",";
             }


          }


          file+=Calories+"];\n"
                + Carbohydrate+"];\n"
                + Fat+"];\n"
                + Protein+"];\n"
                + Preference+"];\n";


   return file;
   }

   int [] returnAmount (int choice, String meal)
   {
      int[] amount={0,0};

      if(meal.equals("B1"))
      {
         //cereal and cakes
```

```
        if(choice==1 || choice==3)
        {
            amount[0]=1;
            amount[1]=3;
        }
        //bread
        else if(choice==2)
        {
            amount[0]=1;
            amount[1]=3;
        }

    }

    else if (meal.equals("B2"))
    {
        if(choice==1 || choice==3)
        {
            amount[0]=8;
            amount[1]=10;
        }
//          cheese
        else if(choice==2)
        {
            amount[0]=1;
            amount[1]=2;
        }

    }
    else if(meal.equals("B3")&& choice==1)

    {
            amount[0]=3;
            amount[1]=6;
```

```
}
else if(meal.equals("B4")&& choice==1)

{
   //sausages


      amount[0]=1;
      amount[1]=2;
}
 //eggs
else if(meal.equals("B5")&& choice==1)
   {
      amount[0]=2;
      amount[1]=5;
   }


else if(meal.equals("L1")|| meal.equals("D1"))
{
   if(choice==1 || choice==2 || choice==3 || choice==4 || choice==5)
   {
      amount[0]=9;
      amount[1]=16;
   }
   else if(choice==6)
   {
      amount[0]=7;
      amount[1]=10;
   }
   else if(choice==7 || choice ==8 || choice==9 || choice==10 || choice==11)
   {
      amount[0]=3;
      amount[1]=7;
```

```
    }

}
else if(meal.equals("L2") || meal.equals("D3"))
{
   if(choice==1)
   {
      amount[0]=6;
      amount[1]=8;
   }
   else if(choice==2)
   {
      amount[0]=7;
      amount[1]=9;
   }
   else if(choice==3)
   {
      amount[0]=1;
      amount[1]=3;
   }
   else if(choice==4)
   {
      amount[0]=1;
      amount[1]=3;
   }
}
else if(meal.equals("L3") || meal.equals("D4"))
{
    if(choice==1)
   {
      amount[0]=8;
      amount[1]=10;
   }
}
```

```
        else if(meal.equals("L4") || meal.equals("D5"))
        {
           if(choice==1)
           {
              amount[0]=6;
              amount[1]=10;
           }
        }
        else if(meal.equals("L5") || meal.equals("D2"))
        {
          if(choice==1 || choice==3)
           {
              amount[0]=8;
              amount[1]=10;
           }
//          cheese
           else if(choice==2)
           {
              amount[0]=1;
              amount[1]=2;
           }
        }


        else if(meal.equals("S1") || meal.equals("S2") || meal.equals("S3"))
        {
           if(choice==1 || choice==2 || choice==5)
           {
              amount[0]=1;
              amount[1]=3;
           }
           else if(choice==3)
           {
              amount[0]=3;
```

```
            amount[1]=6;
        }
        else if(choice==4)
        {
            amount[0]=6;
            amount[1]=8;
        }
        else if(choice==6)
        {
            amount[0]=1;
            amount[1]=2;
        }
    }
    return amount;

}


int []getMedIndex(int count){

  ArrayList<Integer> numbers = new ArrayList<Integer>(count);
        for(int i = 0; i < count; i++)
        {
            numbers.add(i);
        }
    int min=(count/2)-2;
    int max=(count/2)+2;

    int counter=0;
    for(int j =min; j < max; j++)
    {
        counter++;
    }
    if((counter%2)==0){
```

```
        counter++;
        max++;
    }
    System.out.println("Counter for Indexes: "+counter);
 int[] array=new int[counter];

     for(int j =min; j < max; j++)
    {

        array[j]=numbers.get(j);
    }

 return array;
}

int []getRandomIndexes(int count)
{
  int[] array=new int[5];

    ArrayList<Integer> numbers = new ArrayList<Integer>(count);
        for(int i = 0; i < count; i++)
        {
            numbers.add(i);
        }

    Collections.shuffle(numbers);

    for(int j =0; j < 5; j++)
    {

        array[j]=numbers.get(j);
    }
```

```
 return array;
}

public double PrefVal=0;

//CONSTRUCTOR
public QuickNutrition(String username, String password, int b1_choice, int b2_choice, int b3_choice, int b4_choice,int b5_choice,int l1_choice,int l2_choice,int l3_choice,int l4_choice,int l5_choice,int d1_choice, int d2_choice,int  d3_choice, int d4_choice, int d5_choice,int s1_choice, int s2_choice, int s3_choice,
                        String B1QueryNutr, String B1QueryWeight, String B2QueryNutr, String B2QueryWeight, String B3QueryNutr, String B3QueryWeight, String B4QueryNutr, String B4QueryWeight, String B5QueryNutr, String B5QueryWeight,
                String L1QueryNutr, String L1QueryWeight, String L2QueryNutr, String L2QueryWeight, String L3QueryNutr, String L3QueryWeight, String L4QueryNutr, String L4QueryWeight,String L5QueryNutr,String L5QueryWeight,
 String D1QueryNutr, String D1QueryWeight, String D3QueryNutr, String D3QueryWeight, String D4QueryNutr, String D4QueryWeight, String D5QueryNutr, String D5QueryWeight, String D2QueryNutr, String D2QueryWeight, String S1QueryNutr, String S1QueryWeight, String S2QueryNutr, String S2QueryWeight, String S3QueryNutr, String S3QueryWeight, int prefered, double nutrientValue, int minmax)
{
   Prefered=prefered;
   for(int i=0; i<16; i++)
      preferredTable  [i]=0.0;
   PrefVal=nutrientValue;
   System.out.println("B1:  "+b1_choice+"  B2:  "+b2_choice+"  B3:  "+b3_choice+"  B4:  "+b4_choice+"\nL1:  "+l1_choice+"  L2:  "+ l2_choice+"  L3:  "+l3_choice+"  L4:  "+l4_choice+"\nD1:  "+d1_choice+"  D2:  "+d2_choice+"  D3:  "+d3_choice+"  D4:  "+d4_choice+"  D5:  "+d5_choice+"\nS1:  "+s1_choice+ " S2:  "+s2_choice +" S3:  "+s3_choice);


   //*********************************************//
   //*******************BREAKFAST******************//
```

```
//************************************************//


   ///Initialize Breakfast Grains
B1IDs = new String[2000];
double [][]B1Nutr=new double[2000][5];

  for(int i=0;i< B1Count+1; i++){
     for(int j=0; j<5; j++)
        B1Nutr[i][j]=0.0;
   }

   try {
   Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

   Statement stmt = (Statement) con.createStatement();

   ResultSet rs = stmt.executeQuery(B1QueryNutr);

     String[] strings = new String[5000];

    String temp=new String();
     int count = 0, count1=0;
     int counter=0;

     while (rs.next()) {
     //pairnw ta dedomena me vasi to onoma tis stilis
     String s = rs.getString("ndb_no");
          strings[count]=s;

        if(count==0)
        {
           temp=strings[count];
```

```
            B1IDs[count1]=temp;
             count1++;
           }
        if (!temp.equals(strings[count])){
          temp=strings[count];
          B1IDs[count1]=temp;
             count1++;
          counter++;
        }

        if(rs.getString("Nutr_no").equals("208"))//calories
        {
           B1Nutr[counter][0]=rs.getDouble("Nutr_Val");

        }
        else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
        {
           if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              B1Nutr[counter][0]=0.0;
        }

        if (rs.getString("Nutr_no").equals("205"))//carbs
        {

             B1Nutr[counter][1]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              B1Nutr[counter][1]=0.0;
        }
        if (rs.getString("Nutr_no").equals("204"))//fat
```

```
{

      B1Nutr[counter][2]=rs.getDouble("Nutr_Val");
}
else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
{
    if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
      B1Nutr[counter][2]=0.0;
}
if (rs.getString("Nutr_no").equals("203"))//protein
{
   B1Nutr[counter][3]=rs.getDouble("Nutr_Val");
}
else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
{
    if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
      B1Nutr[counter][3]=0.0;
}

 if(prefered!=0)
{
   if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
   {
       B1Nutr[counter][4]=rs.getDouble("Nutr_Val");
   }
    else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
{
   B1Nutr[counter][4]=0.0;
}

}
```

```
                count++;
            }
            System.out.println("Breakfast 1 Products found: "+counter);


           B1Count=counter;



        } catch (SQLException e) {
            System.err.println(e);
        }

            //Get Weight_file Info for Dairy1
            B1Amount=new Double[B1Count+1][20];
            B1Weight=new Double[B1Count+1][20];
            B1Measures=new String[B1Count+1][20];

            Double [][] tempWeights =new Double[B1Count+1][20];
            try {
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

            Statement stmt = (Statement) con.createStatement();

            ResultSet rs = stmt.executeQuery(B1QueryWeight);

            int count=0;
            int counter1=0, counter2=0;

            String temp=new String();
```

```
while (rs.next()) {
   //pairnw ta dedomena me vasi to onoma tis stilis
   String s = rs.getString("ndb_no");
      if(count==0)
      {
         temp=s;


      }


   if (!temp.equals(s)){

     temp=s;

     counter2=0;
   }
   for(int i=0; B1IDs[i]!=null; i++)
   {
    if(B1IDs[i].equals(temp)){
      counter1=i;
      break;
    }
   }


   B1Amount[counter1][counter2]=rs.getDouble("Amount");
   B1Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
   B1Measures[counter1][counter2]=rs.getString("Msre_Desc");

   counter2++;
   count++;
```

```
    }


} catch (SQLException e) {
   System.err.println(e);
}



   //ipologismos nutrients ana grammario

   double[][] B1NutrValPerGram=new double [B1Count+1][5];
   for(int i=0; i<B1Count+1; i++)
      for(int j=0; j<5; j++)
      {
         if(B1Nutr[i][j]!=0)
            B1NutrValPerGram[i][j]=B1Nutr[i][j]/100;
         else
            B1NutrValPerGram[i][j]=0.0;
      }

   //metatropi timwn nutrients se megala integers gia skopous algorithmou
   //ikanopoiisis periorismwn

   B1IntNutr=new int[B1Count+1][5];



   for(int i=0; i<B1Count+1; i++){
      for(int j=0; j<5; j++){
         B1IntNutr[i][j]=(int)(B1NutrValPerGram[i][j]*1000);

         }
```

```
        }


        programString="%constant declaration"
           +"\nint: nutrCount=5;"

           +"\n%%%define arrays for Breakfast length%%%"
           +"\nint: B1Count;"
           +"\nint: minB1;"
           +"\nint: maxB1;"
           +"\nset of int: B1Items= 1..B1Count;"
           +"\narray[B1Items] of int: B1Calories;"
           +"\narray[B1Items] of int: B1Carbohydrate;"
           +"\narray[B1Items] of int: B1Fat;"
           +"\narray[B1Items] of int: B1Protein;"
           +"\narray[B1Items] of int: B1Preference;"
           +"\n\nvar minB1..maxB1: AmountB1;"
           + "\n\nvar 1..B1Count: b1Amount;"
              + "\nconstraint (B1Calories[b1Amount]!=0);";


           tempConstraint1="temp[1]= (B1Calories[b1Amount]*AmountB1*30)";
           tempConstraint2="temp[2]= (B1Carbohydrate[b1Amount]*AmountB1*30)";
           tempConstraint3="temp[3]= (B1Fat[b1Amount]*AmountB1*30)";
           tempConstraint4="temp[4]= (B1Protein[b1Amount]*AmountB1*30)";
           if(prefered!=0)
           tempConstraint5="temp[5]= (B1Preference[b1Amount]*AmountB1*30)";

           outputString="\"B1 Amount: \", show(AmountB1), \" in pos: \",show(b1Amount), \"\\n\"\n++";

           //Dairy Products
      B2IDs = new String[2000];
```

```
double [][]B2Nutr=new double[2000][5];

 for(int i=0;i< B2Count+1; i++){
     for(int j=0; j<5; j++)
        B2Nutr[i][j]=0.0;
  }


  try {

  Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

  Statement stmt = (Statement) con.createStatement();

  ResultSet rs = stmt.executeQuery(B2QueryNutr);

      String[] strings = new String[5000];
      String temp=new String();
      int count = 0, count1=0;
      int counter=0;

    while (rs.next()) {
    //pairnw ta dedomena me vasi to onoma tis stilis
    String s = rs.getString("ndb_no");


    strings[count]=s;

      if(count==0)
      {
        temp=strings[count];
        B2IDs[count1]=temp;
        count1++;
```

```
            }
          if (!temp.equals(strings[count])){
            temp=strings[count];
            B2IDs[count1]=temp;
            count1++;
            counter++;
          }
          if(rs.getString("Nutr_no").equals("208"))//calories
          {
             B2Nutr[counter][0]=rs.getDouble("Nutr_Val");
             System.out.println("calories "+B2Nutr[counter][0]);
          }
          else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
          {
             if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                B2Nutr[counter][0]=0.0;
          }

          if (rs.getString("Nutr_no").equals("205"))//carbs
          {

                B2Nutr[counter][1]=rs.getDouble("Nutr_Val");
          }
          else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
          {
              if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                B2Nutr[counter][1]=0.0;
          }
          if (rs.getString("Nutr_no").equals("204"))//fat
          {

                B2Nutr[counter][2]=rs.getDouble("Nutr_Val");
```

```
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              B2Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
           B2Nutr[counter][3]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              B2Nutr[counter][3]=0.0;
        }

         if(prefered!=0)
        {
           if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
           {
               B2Nutr[counter][4]=rs.getDouble("Nutr_Val");
           }
            else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
           {
              B2Nutr[counter][4]=0.0;
           }

        }

        count++;
       }
```

```
            B2Count=counter;
            System.out.println("Breakfast 2 Products found: "+counter);

             for(int i=0;i< B2Count; i++){
             for(int j=0; j<5; j++)
                System.out.print(B2Nutr[i][j]+" ");
             System.out.println();
          }

       } catch (SQLException e) {
          System.err.println(e);
       }


          //Get Weight_file Info for Dairy1
          B2Amount=new Double [B2Count+1][15];
          B2Weight=new Double [B2Count+1][15];
          B2Measures=new String [B2Count+1][15];


              try {
          Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

          Statement stmt = (Statement) con.createStatement();
          //
          ResultSet rs = stmt.executeQuery(B2QueryWeight);

          int count=0;
          int counter1=0, counter2=0;

          String temp=new String();

          //
```

```
while (rs.next()) {
   //pairnw ta dedomena me vasi to onoma tis stilis
   String s = rs.getString("ndb_no");
      if(count==0)
      {
         temp=s;



      }


   if (!temp.equals(s)){

     temp=s;

     counter2=0;
   }

   for(int i=0; B2IDs[i]!=null; i++)
   {
    if(B2IDs[i].equals(temp)){
      counter1=i;
      break;
    }
   }

   B2Amount[counter1][counter2]=rs.getDouble("Amount");
   B2Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
   B2Measures[counter1][counter2]=rs.getString("Msre_Desc");

   counter2++;
   count++;
```

```
    }


} catch (SQLException e) {
   System.err.println(e);
}

      //ipologismos nutrients ana grammario

   double[][] B2NutrValPerGram=new double [B2Count+1][5];
   for(int i=0; i<B2Count+1; i++)
      for(int j=0; j<5; j++)
      {
         if(B2Nutr[i][j]!=0)
            B2NutrValPerGram[i][j]=B2Nutr[i][j]/100;
         else
            B2NutrValPerGram[i][j]=0.0;
      }

   //metatropi timwn nutrients se megala integers gia skopous algorithmou
   //ikanopoiisis periorismwn

   B2IntNutr=new int[B2Count+1][5];


            for(int i=0; i<B2Count+1; i++){
      for(int j=0; j<5; j++){
         B2IntNutr[i][j]=(int)(B2NutrValPerGram[i][j]*1000);

      }


   }
```

```
 programString+="\n\nint: B2Count;"
     +"\nint: minB2;"
     +"\nint: maxB2;"
     +"\nset of int: B2Items= 1..B2Count;"
     +"\narray[B2Items] of int: B2Calories;"
     +"\narray[B2Items] of int: B2Carbohydrate;"
     +"\narray[B2Items] of int: B2Fat;"
     +"\narray[B2Items] of int: B2Protein;"
     +"\narray[B2Items] of int: B2Preference;"

     +"\n\nvar minB2..maxB2: AmountB2;"
     +"\n\nvar 1..B2Count: b2Amount;"
     + "\nconstraint (B2Calories[b2Amount]!=0);";


   tempConstraint1+="+(B2Calories[b2Amount]*AmountB2*30)";
     tempConstraint2+="+(B2Carbohydrate[b2Amount]*AmountB2*30)";
     tempConstraint3+="+(B2Fat[b2Amount]*AmountB2*30)";
     tempConstraint4+="+(B2Protein[b2Amount]*AmountB2*30)";
     if(prefered!=0)
     tempConstraint5+="+(B2Preference[b2Amount]*AmountB2*30)";

     outputString+="\"B2 Amount: \", show(AmountB2), \" in pos: \",show(b2Amount),
\"\\n\"\n++";
   //************************************************//
   //***************LUNCH & DINNER******************//
   //************************************************//

  //******************Lunch 1*****************************//
 if(l1_choice!=0){
   L1IDs = new String[2000];
   double [][]L1Nutr=new double[2000][5];

    for(int i=0;i< L1Count+1; i++){
```

```
    for(int j=0; j<5; j++)
        L1Nutr[i][j]=0.0;
    }

    try {

    Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

    Statement stmt = (Statement) con.createStatement();
    ResultSet rs = stmt.executeQuery(L1QueryNutr);

        String[] strings = new String[5000];
        String temp=new String();
        int count = 0, count1=0;
        int counter=0;

    while (rs.next()) {
    //pairnw ta dedomena me vasi to onoma tis stilis
    String s = rs.getString("ndb_no");

    strings[count]=s;


    if(count==0)
      {
        temp=strings[count];
        L1IDs[count1]=temp;
        count1++;
      }
    if (!temp.equals(strings[count])){
      temp=strings[count];
      L1IDs[count1]=temp;
      count1++;
```

```
            counter++;
          }
          if(rs.getString("Nutr_no").equals("208"))//calories
          {
             L1Nutr[counter][0]=rs.getDouble("Nutr_Val");

          }
          else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
          {
             if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                L1Nutr[counter][0]=0.0;
          }

          if (rs.getString("Nutr_no").equals("205"))//carbs
          {

                L1Nutr[counter][1]=rs.getDouble("Nutr_Val");
          }
          else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
          {
              if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                L1Nutr[counter][1]=0.0;
          }
          if (rs.getString("Nutr_no").equals("204"))//fat
          {

                L1Nutr[counter][2]=rs.getDouble("Nutr_Val");
          }
          else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
          {
              if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
```

```
            L1Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
           L1Nutr[counter][3]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
            L1Nutr[counter][3]=0.0;
        }

         if(prefered!=0)
        {
           if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
           {
               L1Nutr[counter][4]=rs.getDouble("Nutr_Val");
           }
            else  if(!rs.getString("Nutr_no").equals("208")  &&  !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
           {
              L1Nutr[counter][4]=0.0;
           }

        }



        count++;
        }
        System.out.println("Lunch 1 Products found: "+counter);
        L1Count=counter;
```

```
            for(int i=0;i< counter; i++){
            for(int j=0; j<5; j++)
               System.out.print(L1Nutr[i][j]+" ");
            System.out.println();
         }
      } catch (SQLException e) {
         System.err.println(e);
      }



         L1Amount=new Double [L1Count+1][15];
         L1Weight=new Double [L1Count+1][15];
         L1Measures=new String [L1Count+1][15];


             try {
         Connection                                    con                                    =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

         Statement stmt = (Statement) con.createStatement();
         //
         ResultSet rs = stmt.executeQuery(L1QueryWeight);

         int count=0;
         int counter1=0, counter2=0;

         String temp=new String();


         while (rs.next()) {
            //pairnw ta dedomena me vasi to onoma tis stilis
```

```
  String s = rs.getString("ndb_no");
     if(count==0)
     {
        temp=s;


     }


  if (!temp.equals(s)){

    temp=s;

    counter2=0;
  }

  for(int i=0; L1IDs[i]!=null; i++)
  {
   if(L1IDs[i].equals(temp)){
     counter1=i;
     break;
   }
  }

  if(counter1<L1Count){
     L1Amount[counter1][counter2]=rs.getDouble("Amount");
     L1Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
     L1Measures[counter1][counter2]=rs.getString("Msre_Desc");
  }
  counter2++;
  count++;


}
```

```
} catch (SQLException e) {
   System.err.println(e);
}

          //ipologismos nutrients ana grammario

   double[][] L1NutrValPerGram=new double [L1Count+1][5];
   for(int i=0; i<L1Count+1; i++)
      for(int j=0; j<5; j++)
      {
         if(L1Nutr[i][j]!=0)
            L1NutrValPerGram[i][j]=L1Nutr[i][j]/100;
         else
            L1NutrValPerGram[i][j]=0.0;
      }

   //metatropi timwn nutrients se megala integers gia skopous algorithmou
   //ikanopoiisis periorismwn

   L1IntNutr=new int[L1Count+1][5];


   for(int i=0; i<L1Count+1; i++){
      for(int j=0; j<5; j++){
         L1IntNutr[i][j]=(int)(L1NutrValPerGram[i][j]*1000);


      }

   }
```

```
        programString+="\n\nint: L1Count;"
           +"\nint: minL1;"
           +"\nint: maxL1;"
           +"\nset of int: L1Items= 1..L1Count;"
           +"\narray[L1Items] of int: L1Calories;"
           +"\narray[L1Items] of int: L1Carbohydrate;"
           +"\narray[L1Items] of int: L1Fat;"
           +"\narray[L1Items] of int: L1Protein;"
           +"\narray[L1Items] of int: L1Preference;"

              + "\n\nvar minL1..maxL1: AmountL1;"
              + "\n\nvar 1..L1Count: l1Amount;"
              + "\nconstraint (L1Calories[l1Amount]!=0);";

               tempConstraint1+="+(L1Calories[l1Amount]*AmountL1*30)";
              tempConstraint2+="+(L1Carbohydrate[l1Amount]*AmountL1*30)";
              tempConstraint3+="+(L1Fat[l1Amount]*AmountL1*30)";
              tempConstraint4+="+(L1Protein[l1Amount]*AmountL1*30)";
               if(prefered!=0)
              tempConstraint5+="+(L1Preference[l1Amount]*AmountL1*30)";


          outputString+="\"L1 Amount: \", show(AmountL1), \" in pos: \",show(l1Amount),
\"\\n\"\n++";
   }

   if(d1_choice!=0){
         D1IDs = new String[4000];
         double [][]D1Nutr=new double[4000][5];

           for(int i=0;i< D1Count+1; i++){
            for(int j=0; j<5; j++)
               D1Nutr[i][j]=0.0;
         }
```

```
        //Initialize Vegetables and Vegetable Products
      try {

        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

        Statement stmt = (Statement) con.createStatement();
        ResultSet rs = stmt.executeQuery(D1QueryNutr);

            String[] strings = new String[10000];
            String temp=new String();
            int count = 0, count1=0;
            int counter=0;

        while (rs.next()) {
        //pairnw ta dedomena me vasi to onoma tis stilis
        String s = rs.getString("ndb_no");

        strings[count]=s;
          if(count==0)
            {
               temp=strings[count];
               D1IDs[count1]=temp;
               count1++;
            }
        if (!temp.equals(strings[count])){
          temp=strings[count];
          D1IDs[count1]=temp;
          count1++;
          counter++;
        }
        if(rs.getString("Nutr_no").equals("208"))//calories
        {
```

```
            D1Nutr[counter][0]=rs.getDouble("Nutr_Val");

        }
        else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                D1Nutr[counter][0]=0.0;
        }

        if (rs.getString("Nutr_no").equals("205"))//carbs
        {

            D1Nutr[counter][1]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                D1Nutr[counter][1]=0.0;
        }
        if (rs.getString("Nutr_no").equals("204"))//fat
        {

            D1Nutr[counter][2]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                D1Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
```

```
            D1Nutr[counter][3]=rs.getDouble("Nutr_Val");
          }
          else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
          {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              D1Nutr[counter][3]=0.0;
          }

           if(prefered!=0)
          {
            if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
            {
               D1Nutr[counter][4]=rs.getDouble("Nutr_Val");
            }
             else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
            {
              D1Nutr[counter][4]=0.0;
            }

          }
          count++;
          }
          D1Count=counter;

          System.out.println("Dinner 1 Products found: "+counter);


      } catch (SQLException e) {
         System.err.println(e);
      }
```

```
                    //Get Weight_file Info for Dairy Milk

        D1Amount=new Double [D1Count+1][15];
        D1Weight=new Double [D1Count+1][15];
        D1Measures=new String [D1Count+1][15];


            try {
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

        Statement stmt = (Statement) con.createStatement();

        ResultSet rs = stmt.executeQuery(D1QueryWeight);

        int count=0;
        int counter1=0, counter2=0;

        String temp=new String();


        while (rs.next()) {
           //pairnw ta dedomena me vasi to onoma tis stilis
           String s = rs.getString("ndb_no");
              if(count==0)
              {
                 temp=s;


              }


           if (!temp.equals(s)){
```

```
          temp=s;

          counter2=0;
        }

        for(int i=0; D1IDs[i]!=null; i++)
        {
         if(D1IDs[i].equals(temp)){
           counter1=i;
           break;
         }
        }


        if(counter1< D1Count){
           D1Amount[counter1][counter2]=rs.getDouble("Amount");
           D1Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
           D1Measures[counter1][counter2]=rs.getString("Msre_Desc");
        }
        counter2++;
        count++;


      }


} catch (SQLException e) {
   System.err.println(e);
}


   //ipologismos nutrients ana grammario
```

```
double[][] D1NutrValPerGram=new double [D1Count+1][5];
for(int i=0; i<D1Count+1; i++)
   for(int j=0; j<5; j++)
   {
      if(D1Nutr[i][j]!=0)
         D1NutrValPerGram[i][j]=D1Nutr[i][j]/100;
      else
         D1NutrValPerGram[i][j]=0.0;
   }

//metatropi timwn nutrients se megala integers gia skopous algorithmou
//ikanopoiisis periorismwn

D1IntNutr=new int[D1Count+1][5];


for(int i=0; i<D1Count+1; i++){
   for(int j=0; j<5; j++){
      D1IntNutr[i][j]=(int)(D1NutrValPerGram[i][j]*1000);

   }

}


programString+="\n\nint: D1Count;"
      +"\nint: minD1;"
      +"\nint: maxD1;"
      +"\nset of int: D1Items= 1..D1Count;"
      +"\narray[D1Items] of int: D1Calories;"
      +"\narray[D1Items] of int: D1Carbohydrate;"
      +"\narray[D1Items] of int: D1Fat;"
      +"\narray[D1Items] of int: D1Protein;"
```

```
            +"\narray[D1Items] of int: D1Preference;"

            + "\n\nvar minD1..maxD1: AmountD1;"
            + "\n\nvar 1..D1Count: d1Amount;"
            + "\nconstraint (D1Calories[d1Amount]!=0);";


          tempConstraint1+="+(D1Calories[d1Amount]*AmountD1*30)";
           tempConstraint2+="+(D1Carbohydrate[d1Amount]*AmountD1*30)";
           tempConstraint3+="+(D1Fat[d1Amount]*AmountD1*30)";
           tempConstraint4+="+(D1Protein[d1Amount]*AmountD1*30)";
           if(prefered!=0)
           tempConstraint5+="+(D1Preference[d1Amount]*AmountD1*30)";


           outputString+="\"D1 Amount: \", show(AmountD1), \" in pos: \",show(d1Amount), \"\\n\"\n++";

//*************************************************************************************//

//*****************************************SNACKS*************************************//

//*************************************************************************************//
   }
            S1IDs = new String[2000];
            double [][]S1Nutr=new double[2000][5];

             for(int i=0;i< S1Count+1; i++){
                for(int j=0; j<5; j++)
                    S1Nutr[i][j]=0.0;
             }
```

```
    //Initialize Snacks 1
        try {

        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

        Statement stmt = (Statement) con.createStatement();
        ResultSet rs = stmt.executeQuery(S1QueryNutr);

            String[] strings = new String[7000];
            String temp=new String();
            int count = 0, count1=0;
            int counter=0;

          while (rs.next()) {
          //pairnw ta dedomena me vasi to onoma tis stilis
          String s = rs.getString("ndb_no");

          strings[count]=s;
           if(count==0)
            {
               temp=strings[count];
               S1IDs[count1]=temp;
               count1++;
            }
          if (!temp.equals(strings[count])){
            temp=strings[count];
            S1IDs[count1]=temp;
            count1++;
            counter++;
          }
            if(rs.getString("Nutr_no").equals("208"))//calories
          {
```

```
            S1Nutr[counter][0]=rs.getDouble("Nutr_Val");

        }
        else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
        {
           if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              S1Nutr[counter][0]=0.0;
        }

        if (rs.getString("Nutr_no").equals("205"))//carbs
        {

              S1Nutr[counter][1]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              S1Nutr[counter][1]=0.0;
        }
        if (rs.getString("Nutr_no").equals("204"))//fat
        {

              S1Nutr[counter][2]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              S1Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
```

```
            S1Nutr[counter][3]=rs.getDouble("Nutr_Val");
          }
          else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
          {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              S1Nutr[counter][3]=0.0;
          }

           if(prefered!=0)
          {
            if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
            {
              S1Nutr[counter][4]=rs.getDouble("Nutr_Val");
            }
             else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
            {
              S1Nutr[counter][4]=0.0;
            }

          }
          count++;
          }
          S1Count=counter;
          System.out.println("Snack 1 Products found: "+counter);

          for(int i=0;i< counter; i++){
          for(int j=0; j<5; j++)
            System.out.print(S1Nutr[i][j]+" ");
          System.out.println();
        }
```

```
        } catch (SQLException e) {
        System.err.println(e);
    }

        //Get Weight_file Info for Beverages

        S1Amount=new Double [S1Count+1][15];
        S1Weight=new Double [S1Count+1][15];
        S1Measures=new String [S1Count+1][15];


            try {
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

        Statement stmt = (Statement) con.createStatement();

        ResultSet rs = stmt.executeQuery(S1QueryWeight);

        int count=0;
        int counter1=0, counter2=0;

        String temp=new String();


        while (rs.next()) {
          //pairnw ta dedomena me vasi to onoma tis stilis
          String s = rs.getString("ndb_no");
             if(count==0)
             {
                temp=s;


             }
```

```
        if (!temp.equals(s)){

          temp=s;

          counter2=0;
        }

        for(int i=0; S1IDs[i]!=null; i++)
        {
         if(S1IDs[i].equals(temp)){
            counter1=i;
            break;
         }
        }

        if(counter1<S1Count){
            S1Amount[counter1][counter2]=rs.getDouble("Amount");
            S1Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
            S1Measures[counter1][counter2]=rs.getString("Msre_Desc");
        }
        counter2++;
        count++;


      }


} catch (SQLException e) {
   System.err.println(e);
}
```

```
//ipologismos nutrients ana grammario

double[][] S1NutrValPerGram=new double [S1Count+1][5];
for(int i=0; i<S1Count+1; i++)
   for(int j=0; j<5; j++)
   {
      if(S1Nutr[i][j]!=0)
         S1NutrValPerGram[i][j]=S1Nutr[i][j]/100;
      else
         S1NutrValPerGram[i][j]=0.0;
   }

//metatropi timwn nutrients se megala integers gia skopous algorithmou
//ikanopoiisis periorismwn

S1IntNutr=new int[S1Count+1][5];


for(int i=0; i<S1Count+1; i++){
   for(int j=0; j<5; j++){
      S1IntNutr[i][j]=(int)(S1NutrValPerGram[i][j]*1000);


   }

}


 programString+="\n\nint: S1Count;"
      +"\nint: minS1;"
      +"\nint: maxS1;"
      +"\nset of int: S1Items= 1..S1Count;"
      +"\narray[S1Items] of int: S1Calories;"
```

```
            +"\narray[S1Items] of int: S1Carbohydrate;"
            +"\narray[S1Items] of int: S1Fat;"
            +"\narray[S1Items] of int: S1Protein;"
            +"\narray[S1Items] of int: S1Preference;"

             + "\n\nvar minS1..maxS1: AmountS1;"
             + "\n\nvar 1..S1Count: s1Amount;"
             + "\nconstraint (S1Calories[s1Amount]!=0);";


          tempConstraint1+="+(S1Calories[s1Amount]*AmountS1*30)";
          tempConstraint2+="+(S1Carbohydrate[s1Amount]*AmountS1*30)";
          tempConstraint3+="+(S1Fat[s1Amount]*AmountS1*30)";
          tempConstraint4+="+(S1Protein[s1Amount]*AmountS1*30)";
           if(prefered!=0)
          tempConstraint5+="+(S1Preference[s1Amount]*AmountS1*30)";


          outputString+="\"S1 Amount: \", show(AmountS1), \" in pos: \",show(s1Amount), \"\\n\"\n++";
        S2IDs = new String[2000];
        double [][]S2Nutr=new double[2000][5];

          for(int i=0;i< S2Count+1; i++){
           for(int j=0; j<5; j++)
              S2Nutr[i][j]=0.0;
        }

    //Initialize Snacks 2
        try {

       Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
```

```
Statement stmt = (Statement) con.createStatement();
ResultSet rs = stmt.executeQuery(S2QueryNutr);

    String[] strings = new String[7000];
    String temp=new String();
    int count = 0, count1=0;
    int counter=0;

  while (rs.next()) {
  //pairnw ta dedomena me vasi to onoma tis stilis
  String s = rs.getString("ndb_no");

  strings[count]=s;
   if(count==0)
    {
      temp=strings[count];
      S2IDs[count1]=temp;
      count1++;
    }
  if (!temp.equals(strings[count])){
    temp=strings[count];
    S2IDs[count1]=temp;
    count1++;
    counter++;
  }
   if(rs.getString("Nutr_no").equals("208"))//calories
  {
     S2Nutr[counter][0]=rs.getDouble("Nutr_Val");

  }
  else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
  {
     if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
```

```
            S2Nutr[counter][0]=0.0;
        }

        if (rs.getString("Nutr_no").equals("205"))//carbs
        {

            S2Nutr[counter][1]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
            S2Nutr[counter][1]=0.0;
        }
        if (rs.getString("Nutr_no").equals("204"))//fat
        {

            S2Nutr[counter][2]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
            S2Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
           S2Nutr[counter][3]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
            S2Nutr[counter][3]=0.0;
```

```
            }

             if(prefered!=0)
            {
               if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
               {
                   S2Nutr[counter][4]=rs.getDouble("Nutr_Val");
               }
                else  if(!rs.getString("Nutr_no").equals("208")  &&  !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
               {
                  S2Nutr[counter][4]=0.0;
               }

            }
            count++;
            }
            S2Count=counter;
            System.out.println("Snack 2 Products found: "+counter);

      } catch (SQLException e) {
         System.err.println(e);
      }
               //Get Weight_file Info for Beverages

         S2Amount=new Double [S2Count+1][15];
         S2Weight=new Double [S2Count+1][15];
         S2Measures=new String [S2Count+1][15];


             try {
         Connection                                    con                                    =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
```

```
Statement stmt = (Statement) con.createStatement();

ResultSet rs = stmt.executeQuery(S2QueryWeight);

int count=0;
int counter1=0, counter2=0;

String temp=new String();


while (rs.next()) {
   //pairnw ta dedomena me vasi to onoma tis stilis
   String s = rs.getString("ndb_no");
      if(count==0)
      {
         temp=s;


      }


   if (!temp.equals(s)){

     temp=s;

     counter2=0;
   }

   for(int i=0; S2IDs[i]!=null; i++)
   {
    if(S2IDs[i].equals(temp)){
      counter1=i;
      break;
    }
```

```
        }


        if(counter1<S2Count){
           S2Amount[counter1][counter2]=rs.getDouble("Amount");
           S2Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
           S2Measures[counter1][counter2]=rs.getString("Msre_Desc");
        }
        counter2++;
        count++;


     }


} catch (SQLException e) {
   System.err.println(e);
}

  //ipologismos nutrients ana grammario

   double[][] S2NutrValPerGram=new double [S2Count+1][5];
   for(int i=0; i<S2Count+1; i++)
      for(int j=0; j<5; j++)
      {
         if(S2Nutr[i][j]!=0)
            S2NutrValPerGram[i][j]=S2Nutr[i][j]/100;
         else
            S2NutrValPerGram[i][j]=0.0;
      }

   //metatropi timwn nutrients se megala integers gia skopous algorithmou
   //ikanopoiisis periorismwn
```

```
S2IntNutr=new int[S2Count+1][5];


for(int i=0; i<S2Count+1; i++){
   for(int j=0; j<5; j++){
      S2IntNutr[i][j]=(int)(S2NutrValPerGram[i][j]*1000);


   }

}



programString+="\n\nint: S2Count;"
      +"\nint: minS2;"
      +"\nint: maxS2;"
      +"\nset of int: S2Items= 1..S2Count;"
      +"\narray[S2Items] of int: S2Calories;"
      +"\narray[S2Items] of int: S2Carbohydrate;"
      +"\narray[S2Items] of int: S2Fat;"
      +"\narray[S2Items] of int: S2Protein;"
      +"\narray[S2Items] of int: S2Preference;"

      + "\n\nvar minS2..maxS2: AmountS2;"
      + "\n\nvar 1..S2Count: s2Amount;"
      + "\nconstraint (S2Calories[s2Amount]!=0);";


  tempConstraint1+="+(S2Calories[s2Amount]*AmountS2*30)";
   tempConstraint2+="+(S2Carbohydrate[s2Amount]*AmountS2*30)";
   tempConstraint3+="+(S2Fat[s2Amount]*AmountS2*30)";
   tempConstraint4+="+(S2Protein[s2Amount]*AmountS2*30)";
   if(prefered!=0)
```

```
tempConstraint5+="+(S2Preference[s2Amount]*AmountS2*30)";


outputString+="\"S2 Amount: \", show(AmountS2), \" in pos: \",show(s2Amount), \"\\n\"\n++";

S3IDs = new String[2000];
double [][]S3Nutr=new double[2000][5];

for(int i=0;i< S3Count+1; i++){
   for(int j=0; j<5; j++)
      S3Nutr[i][j]=0.0;
}

//Initialize Snacks 3
try {

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

Statement stmt = (Statement) con.createStatement();
ResultSet rs = stmt.executeQuery(S3QueryNutr);

String[] strings = new String[7000];
String temp=new String();
int count = 0, count1=0;
int counter=0;

while (rs.next()) {
//pairnw ta dedomena me vasi to onoma tis stilis
String s = rs.getString("ndb_no");

strings[count]=s;
 if(count==0)
```

```
            {
                temp=strings[count];
                S3IDs[count1]=temp;
                count1++;
            }
          if (!temp.equals(strings[count])){
            temp=strings[count];
            S3IDs[count1]=temp;
            count1++;
            counter++;
          }
           if(rs.getString("Nutr_no").equals("208"))//calories
          {
             S3Nutr[counter][0]=rs.getDouble("Nutr_Val");

          }
          else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
          {
             if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                S3Nutr[counter][0]=0.0;
          }

          if (rs.getString("Nutr_no").equals("205"))//carbs
          {

                S3Nutr[counter][1]=rs.getDouble("Nutr_Val");
          }
          else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
          {
              if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                S3Nutr[counter][1]=0.0;
          }
```

```
        if (rs.getString("Nutr_no").equals("204"))//fat
        {

            S3Nutr[counter][2]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              S3Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
           S3Nutr[counter][3]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              S3Nutr[counter][3]=0.0;
        }

         if(prefered!=0)
        {
           if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
           {
               S3Nutr[counter][4]=rs.getDouble("Nutr_Val");
           }
            else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
           {
              S3Nutr[counter][4]=0.0;
           }
        }
```

```
            count++;
            }
            S3Count=counter;
            System.out.println("Snack 3 Products found: "+counter);

        } catch (SQLException e) {
            System.err.println(e);
        }


              //Get Weight_file Info for Nuts

            S3Amount=new Double [S3Count+1][15];
            S3Weight=new Double [S3Count+1][15];
            S3Measures=new String [S3Count+1][15];


              try {
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
            //getConnection();
            Statement stmt = (Statement) con.createStatement();
            //
            ResultSet rs = stmt.executeQuery(S3QueryWeight);

            int count=0;
            int counter1=0, counter2=0;

            String temp=new String();

            //
            while (rs.next()) {
              //pairnw ta dedomena me vasi to onoma tis stilis
              String s = rs.getString("ndb_no");
```

```
    if(count==0)
    {
       temp=s;

    }


  if (!temp.equals(s)){

    temp=s;

    counter2=0;
  }

  for(int i=0; S3IDs[i]!=null; i++)
  {
   if(S3IDs[i].equals(temp)){
     counter1=i;
     break;
   }
  }


  if(counter1<S3Count){
     S3Amount[counter1][counter2]=rs.getDouble("Amount");
     S3Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
     S3Measures[counter1][counter2]=rs.getString("Msre_Desc");
  }
  counter2++;
  count++;


}
```

```
} catch (SQLException e) {
   System.err.println(e);
}



   //ipologismos nutrients ana grammario

   double[][] S3NutrValPerGram=new double [S3Count+1][5];
   for(int i=0; i<S3Count+1; i++)
      for(int j=0; j<5; j++)
      {
         if(S3Nutr[i][j]!=0)
            S3NutrValPerGram[i][j]=S3Nutr[i][j]/100;
         else
            S3NutrValPerGram[i][j]=0.0;
      }

   //metatropi timwn nutrients se megala integers gia skopous algorithmou
   //ikanopoiisis periorismwn

   S3IntNutr=new int[S3Count+1][5];



   for(int i=0; i<S3Count+1; i++){
      for(int j=0; j<5; j++){
         S3IntNutr[i][j]=(int)(S3NutrValPerGram[i][j]*1000);



      }

   }
```

```
programString+="\n\nint: S3Count;"
      +"\nint: minS3;"
      +"\nint: maxS3;"
      +"\nset of int: S3Items= 1..S3Count;"
      +"\narray[S3Items] of int: S3Calories;"
      +"\narray[S3Items] of int: S3Carbohydrate;"
      +"\narray[S3Items] of int: S3Fat;"
      +"\narray[S3Items] of int: S3Protein;"
      +"\narray[S3Items] of int: S3Preference;"

      + "\n\nvar minS3..maxS3: AmountS3;"
      + "\n\nvar 1..S3Count: s3Amount;"
      + "\nconstraint (S3Calories[s3Amount]!=0);";


   tempConstraint1+="+(S3Calories[s3Amount]*AmountS3*30)";
   tempConstraint2+="+(S3Carbohydrate[s3Amount]*AmountS3*30)";
   tempConstraint3+="+(S3Fat[s3Amount]*AmountS3*30)";
   tempConstraint4+="+(S3Protein[s3Amount]*AmountS3*30)";
   if(prefered!=0)
   tempConstraint5+="+(S3Preference[s3Amount]*AmountS3)";


outputString+="\"S3 Amount: \", show(AmountS3), \" in pos: \",show(s3Amount), \"\\n\"";


/**************************************************************************
*************/
      //Initialize Nutrient Constants

      // Get Calories From User in Database
      try {
```

```
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

        Statement stmt = (Statement) con.createStatement();

        ResultSet rs = stmt.executeQuery("select * from `nutrient_data`.`user_details` where `Username`='"+username+"' and `Password`='"+password+"';");
           int count=0;

        while (rs.next()) {

           age = rs.getInt("Age");
           gender = rs.getString("Gender");

              Calories = rs.getDouble("Suggested_Calories");

           count++;
        }
        System.out.println("Calories: "+Calories);

     } catch (SQLException e) {
        System.err.println(e);
     }


        Carbohydrate=((Calories*55)/100)/4;
        System.out.println("Carbs: "+Carbohydrate);

        Fat=((Calories*30)/100)/9;
        System.out.println("Fat: "+Fat);

        Protein=((Calories*15)/100)/4;
        System.out.println("Protein: "+Protein);
```

```
        int id_no=0;
        try {
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
        //getConnection();
        Statement stmt = (Statement) con.createStatement();

        ResultSet rs = stmt.executeQuery("select * from nutrient_data.candidate_categories where (min_age<'"+age+"' and max_age>'"+age+"') and sex='"+gender+"';");
           int count=0;

        while (rs.next()) {

           id_no = rs.getInt("id_candidate");
           count++;
        }
        System.out.println("Id: "+id_no);

     } catch (SQLException e) {
        System.err.println(e);
     }

        try {
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

        Statement stmt = (Statement) con.createStatement();

        ResultSet rs = stmt.executeQuery("select * from nutrient_data.nutrient_requirements where candidate_id='"+id_no+"';");
           int count=0;

        while (rs.next()) {
```

```
            nutr_code = rs.getInt("nutr_code");

            //natrio
            if(nutr_code==307)
               Sodium = rs.getDouble("nutr_value");
            //kalio
            else if(nutr_code==306)
               Potassium = rs.getDouble("nutr_value");
            else if(nutr_code==320)
               VitaminA = rs.getDouble("nutr_value");
            else if(nutr_code==401)
               VitaminC = rs.getDouble("nutr_value");
            else if(nutr_code==301)
               Calcium = rs.getDouble("nutr_value");
             else if(nutr_code==303)
               Iron = rs.getDouble("nutr_value");
             else if(nutr_code==304)
               Magnesium = rs.getDouble("nutr_value");

            count++;
          }
          System.out.println("Sodium: "+Sodium+" Potassium: "+Potassium+" Vitamin A: "+VitaminA+"
Vitamin C: "+VitaminC+" Calcium: "+Calcium+" Iron: "+Iron+" Magnesium: "+Magnesium);

       } catch (SQLException e) {
          System.err.println(e);
       }

       //******************Breakfast 3*****************************//
          if(b3_choice==1){

              ///Initialize Fruits and Fruit Juices
      B3IDs = new String[2000];
      double [][]B3Nutr=new double[2000][5];
```

```
    for(int i=0;i< B3Count+1; i++){
        for(int j=0; j<5; j++)
           B3Nutr[i][j]=0.0;
        }


        try {

        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

        Statement stmt = (Statement) con.createStatement();
        ResultSet rs = stmt.executeQuery(B3QueryNutr);

            String[] strings = new String[5000];
            String temp=new String();
            int count = 0, count1=0;
            int counter=0;

          while (rs.next()) {
          //pairnw ta dedomena me vasi to onoma tis stilis
          String s = rs.getString("ndb_no");


          strings[count]=s;

          if(count==0)
            {
              temp=strings[count];
              B3IDs[count1]=temp;
              count1++;
            }
```

```
            if (!temp.equals(strings[count])){
              temp=strings[count];
              B3IDs[count1]=temp;
              count1++;
              counter++;
            }
             if(rs.getString("Nutr_no").equals("208"))//calories
            {
               B3Nutr[counter][0]=rs.getDouble("Nutr_Val");

            }
            else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
            {
               if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                  B3Nutr[counter][0]=0.0;
            }

            if (rs.getString("Nutr_no").equals("205"))//carbs
            {

                  B3Nutr[counter][1]=rs.getDouble("Nutr_Val");
            }
            else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
            {
                if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                  B3Nutr[counter][1]=0.0;
            }
            if (rs.getString("Nutr_no").equals("204"))//fat
            {

                  B3Nutr[counter][2]=rs.getDouble("Nutr_Val");
            }
```

```
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              B3Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
           B3Nutr[counter][3]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              B3Nutr[counter][3]=0.0;
        }

         if(prefered!=0)
        {
           if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
           {
               B3Nutr[counter][4]=rs.getDouble("Nutr_Val");
           }
            else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
           {
              B3Nutr[counter][4]=0.0;
           }
        }

        count++;
        }

        System.out.println("Breakfast 3 Products found: "+counter);
```

```
            B3Count=counter;


        } catch (SQLException e) {
            System.err.println(e);
        }

            //Get Weight_file Info for Dairy1

            B3Amount=new Double [B3Count+1][15];
            B3Weight=new Double [B3Count+1][15];
            B3Measures=new String [B3Count+1][15];


                try {
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

            Statement stmt = (Statement) con.createStatement();

            ResultSet rs = stmt.executeQuery(B3QueryWeight);

            int count=0;
            int counter1=0, counter2=0;

            String temp=new String();


            while (rs.next()) {
               //pairnw ta dedomena me vasi to onoma tis stilis
               String s = rs.getString("ndb_no");
                  if(count==0)
                  {
                     temp=s;
```

```
        }


      if (!temp.equals(s)){

        temp=s;

        counter2=0;
      }

      for(int i=0; B3IDs[i]!=null; i++)
      {
       if(B3IDs[i].equals(temp)){
          counter1=i;
          break;
       }
      }



      B3Amount[counter1][counter2]=rs.getDouble("Amount");
      B3Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
      B3Measures[counter1][counter2]=rs.getString("Msre_Desc");

      counter2++;
      count++;


    }


} catch (SQLException e) {
```

```
    System.err.println(e);
}


    //ipologismos nutrients ana grammario

    double[][] B3NutrValPerGram=new double [B3Count+1][5];
    for(int i=0; i<B3Count+1; i++)
        for(int j=0; j<5; j++)
        {
            if(B3Nutr[i][j]!=0)
                B3NutrValPerGram[i][j]=B3Nutr[i][j]/100;
            else
                B3NutrValPerGram[i][j]=0.0;
        }

    //metatropi timwn nutrients se megala integers gia skopous algorithmou
    //ikanopoiisis periorismwn

    B3IntNutr=new int[B3Count+1][5];



    for(int i=0; i<B3Count+1; i++){
        for(int j=0; j<5; j++){
            B3IntNutr[i][j]=(int)(B3NutrValPerGram[i][j]*1000);

        }

    }


    programString+="\n\nint: B3Count;"
            +"\nint: minB3;"
```

```
        +"\nint: maxB3;"
        +"\nset of int: B3Items= 1..B3Count;"
        +"\narray[B3Items] of int: B3Calories;"
        +"\narray[B3Items] of int: B3Carbohydrate;"
        +"\narray[B3Items] of int: B3Fat;"
        +"\narray[B3Items] of int: B3Protein;"
        +"\narray[B3Items] of int: B3Preference;"

        + "\n\nvar minB3..maxB3: AmountB3;"
        + "\n\nvar 1..B3Count: b3Amount;"
        + "\nconstraint (B3Calories[b3Amount]!=0);";

          tempConstraint1+="+(B3Calories[b3Amount]*AmountB3*30)";
       tempConstraint2+="+(B3Carbohydrate[b3Amount]*AmountB3*30)";
       tempConstraint3+="+(B3Fat[b3Amount]*AmountB3*30)";
       tempConstraint4+="+(B3Protein[b3Amount]*AmountB3*30)";
       if(prefered!=0)
       tempConstraint5+="+(B3Preference[b3Amount]*AmountB3*30)";

       outputString+="\n++\"B3 Amount: \", show(AmountB3), \" in pos: \",show(b3Amount),
\"\\n\"";
    }

    //*******************Breakfast 4*****************************//
     //Protein Breakfast
    if(b4_choice!=0){
    B4IDs = new String[2000];
    double [][]B4Nutr=new double[2000][5];

      for(int i=0;i< B4Count+1; i++){
       for(int j=0; j<5; j++)
          B4Nutr[i][j]=0.0;
    }
```

```
try {

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

Statement stmt = (Statement) con.createStatement();
ResultSet rs = stmt.executeQuery(B4QueryNutr);

    String[] strings = new String[1000];
    String temp=new String();
    int count = 0, count1=0;
    int counter=0;

  while (rs.next()) {
  //pairnw ta dedomena me vasi to onoma tis stilis
  String s = rs.getString("ndb_no");
  strings[count]=s;
  if(count==0)
    {
      temp=strings[count];
      B4IDs[count1]=temp;
      count1++;
    }
  if (!temp.equals(strings[count])){
    temp=strings[count];
    B4IDs[count1]=temp;
    count1++;
    counter++;
  }
   if(rs.getString("Nutr_no").equals("208"))//calories
  {
     B4Nutr[counter][0]=rs.getDouble("Nutr_Val");

  }
```

```
        else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
        {
           if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              B4Nutr[counter][0]=0.0;
        }

        if (rs.getString("Nutr_no").equals("205"))//carbs
        {

              B4Nutr[counter][1]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              B4Nutr[counter][1]=0.0;
        }
        if (rs.getString("Nutr_no").equals("204"))//fat
        {

              B4Nutr[counter][2]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              B4Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
           B4Nutr[counter][3]=rs.getDouble("Nutr_Val");
        }
```

```
            else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
            {
                if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                  B4Nutr[counter][3]=0.0;
            }

             if(prefered!=0)
            {
               if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
               {
                   B4Nutr[counter][4]=rs.getDouble("Nutr_Val");
               }
                else  if(!rs.getString("Nutr_no").equals("208")  &&  !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
               {
                  B4Nutr[counter][4]=0.0;
               }
            }

            count++;
            }
            System.out.println("Breakfast 4 Products found: "+counter);

            B4Count=counter;


        } catch (SQLException e) {
           System.err.println(e);
        }


          //Get Weight_file Info for Sausages
```

```
B4Amount=new Double [B4Count+1][30];
B4Weight=new Double [B4Count+1][30];
B4Measures=new String [B4Count+1][30];


    try {
Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

Statement stmt = (Statement) con.createStatement();

ResultSet rs = stmt.executeQuery(B4QueryWeight);

int count=0;
int counter1=0, counter2=0;

String temp=new String();


while (rs.next()) {
  //pairnw ta dedomena me vasi to onoma tis stilis
  String s = rs.getString("ndb_no");
    if(count==0)
    {
      temp=s;


    }


  if (!temp.equals(s)){

    temp=s;
```

```
          counter1++;
          counter2=0;
        }

        for(int i=0; B4IDs[i]!=null; i++)
        {
         if(B4IDs[i].equals(temp)){
           counter1=i;
           break;
         }
        }



        if(counter1<B4Count){

        B4Amount[counter1][counter2]=rs.getDouble("Amount");
        B4Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
        B4Measures[counter1][counter2]=rs.getString("Msre_Desc");
        }
        counter2++;
        count++;


      }


} catch (SQLException e) {
   System.err.println(e);
}

         //ipologismos nutrients ana grammario

   double[][] B4NutrValPerGram=new double [B4Count+1][5];
```

```
for(int i=0; i<B4Count+1; i++)
   for(int j=0; j<5; j++)
   {
      if(B4Nutr[i][j]!=0)
         B4NutrValPerGram[i][j]=B4Nutr[i][j]/100;
      else
         B4NutrValPerGram[i][j]=0.0;
   }

//metatropi timwn nutrients se megala integers gia skopous algorithmou
//ikanopoiisis periorismwn

B4IntNutr=new int[B4Count+1][5];


for(int i=0; i<B4Count+1; i++){
   for(int j=0; j<5; j++){
      B4IntNutr[i][j]=(int)(B4NutrValPerGram[i][j]*1000);


   }

}


programString+="\n\nint: B4Count;"
      +"\nint: minB4;"
      +"\nint: maxB4;"
      +"\nset of int: B4Items= 1..B4Count;"
      +"\narray[B4Items] of int: B4Calories;"
      +"\narray[B4Items] of int: B4Carbohydrate;"
      +"\narray[B4Items] of int: B4Fat;"
      +"\narray[B4Items] of int: B4Protein;"
      +"\narray[B4Items] of int: B4Preference;"
```

```
            + "\n\nvar minB4..maxB4: AmountB4;"
               + "\n\nvar 1..B4Count: b4Amount;"
               + "\nconstraint (B4Calories[b4Amount]!=0);";


              tempConstraint1+="+(B4Calories[b4Amount]*AmountB4*30)";
            tempConstraint2+="+(B4Carbohydrate[b4Amount]*AmountB4*30)";
            tempConstraint3+="+(B4Fat[b4Amount]*AmountB4*30)";
            tempConstraint4+="+(B4Protein[b4Amount]*AmountB4*30)";
            if(prefered!=0)
            tempConstraint5+="+(B4Preference[b4Amount]*AmountB4*30)";


        outputString+="\n++\"B4 Amount: \", show(AmountB4), \" in pos: \",show(b4Amount),
\"\\n\"";

        }


    if(b5_choice==1){
        B5IDs = new String[2000];
        double [][]B5Nutr=new double[2000][5];


          for(int i=0;i< B5Count+1; i++){
           for(int j=0; j<5; j++)
              B5Nutr[i][j]=0.0;
        }

     //Initialize Pork Products
        try {
```

```
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

Statement stmt = (Statement) con.createStatement();
ResultSet rs = stmt.executeQuery(B5QueryNutr);

    String[] strings = new String[7000];
    String temp=new String();
    int count = 0, count1=0;
    int counter=0;

  while (rs.next()) {
  //pairnw ta dedomena me vasi to onoma tis stilis
  String s = rs.getString("ndb_no");

  strings[count]=s;
   if(count==0)
    {
      temp=strings[count];
      B5IDs[count1]=temp;
      count1++;
    }
  if (!temp.equals(strings[count])){
    temp=strings[count];
    B5IDs[count1]=temp;
    count1++;
    counter++;
  }
   if(rs.getString("Nutr_no").equals("208"))//calories
  {
     B5Nutr[counter][0]=rs.getDouble("Nutr_Val");

  }
```

```
        else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
        {
           if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              B5Nutr[counter][0]=0.0;
        }

        if (rs.getString("Nutr_no").equals("205"))//carbs
        {

              B5Nutr[counter][1]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              B5Nutr[counter][1]=0.0;
        }
        if (rs.getString("Nutr_no").equals("204"))//fat
        {

              B5Nutr[counter][2]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              B5Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
           B5Nutr[counter][3]=rs.getDouble("Nutr_Val");
        }
```

```
            else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
            {
                if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                  B5Nutr[counter][3]=0.0;
            }

             if(prefered!=0)
            {
              if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
              {
                  B5Nutr[counter][4]=rs.getDouble("Nutr_Val");
              }
               else  if(!rs.getString("Nutr_no").equals("208")  &&  !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
              {
                 B5Nutr[counter][4]=0.0;
              }
            }
            count++;
            }

            System.out.println("Lunch 4 Products found: "+counter);
            B5Count=counter;



      } catch (SQLException e) {
         System.err.println(e);
      }

                              //Get Weight_file Info for Pork

         B5Amount=new Double [B5Count+1][15];
```

```
B5Weight=new Double [B5Count+1][15];
B5Measures=new String [B5Count+1][15];


    try {
Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

Statement stmt = (Statement) con.createStatement();

ResultSet rs = stmt.executeQuery(B5QueryWeight);

int count=0;
int counter1=0, counter2=0;

String temp=new String();


while (rs.next()) {
  //pairnw ta dedomena me vasi to onoma tis stilis
  String s = rs.getString("ndb_no");
    if(count==0)
    {
      temp=s;


    }


  if (!temp.equals(s)){

    temp=s;

    counter2=0;
```

```
        }
        for(int i=0; B5IDs[i]!=null; i++)
        {
         if(B5IDs[i].equals(temp)){
            counter1=i;
            break;
         }
        }

        if(counter1<B5Count){
            B5Amount[counter1][counter2]=rs.getDouble("Amount");
            B5Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
            B5Measures[counter1][counter2]=rs.getString("Msre_Desc");
        }
        counter2++;
        count++;


     }


} catch (SQLException e) {
    System.err.println(e);
}


              //ipologismos nutrients ana grammario

    double[][] B5NutrValPerGram=new double [B5Count+1][5];
    for(int i=0; i<B5Count+1; i++)
        for(int j=0; j<5; j++)
        {
            if(B5Nutr[i][j]!=0)
                B5NutrValPerGram[i][j]=B5Nutr[i][j]/100;
```

```
        else
            B5NutrValPerGram[i][j]=0.0;
    }

//metatropi timwn nutrients se megala integers gia skopous algorithmou
//ikanopoiisis periorismwn

B5IntNutr=new int[B5Count+1][5];


for(int i=0; i<B5Count+1; i++){
    for(int j=0; j<5; j++){
       B5IntNutr[i][j]=(int)(B5NutrValPerGram[i][j]*1000);


    }

}


programString+="\n\nint: B5Count;"
        +"\nint: minB5;"
        +"\nint: maxB5;"
        +"\nset of int: B5Items= 1..B5Count;"
        +"\narray[B5Items] of int: B5Calories;"
        +"\narray[B5Items] of int: B5Carbohydrate;"
        +"\narray[B5Items] of int: B5Fat;"
        +"\narray[B5Items] of int: B5Protein;"
        +"\narray[B5Items] of int: B5Preference;"

        + "\n\nvar minB5..maxB5: AmountB5;"
        + "\n\nvar 1..B5Count: b5Amount;"
        + "\nconstraint (B5Calories[b5Amount]!=0);";
```

```
            tempConstraint1+="+(B5Calories[b5Amount]*AmountB5*30)";
            tempConstraint2+="+(B5Carbohydrate[b5Amount]*AmountB5*30)";
            tempConstraint3+="+(B5Fat[b5Amount]*AmountB5*30)";
            tempConstraint4+="+(B5Protein[b5Amount]*AmountB5*30)";
            if(prefered!=0)
            tempConstraint5+="+(B5Preference[b5Amount]*AmountB5*30)";


            outputString+="\n++\"B5 Amount: \", show(AmountB5), \" in pos: \",show(b5Amount),
\"\\n\"";
          }



      //elegxw ean tha stilw t l2_choice, l3_choice kai l3_choice
    if(l1_choice==7 || l1_choice==8 || l1_choice==9 || l1_choice==10 || l1_choice==0)
      {
      if(l2_choice!=0){
            System.out.println("lunch 2");
          L2IDs = new String[2000];
          double [][]L2Nutr=new double[2000][5];

           for(int i=0;i< L2Count+1; i++){
            for(int j=0; j<5; j++)
               L2Nutr[i][j]=0.0;
          }

          //Initialize DairyMilk
          try {

         Connection                                    con                                    =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
```

```
Statement stmt = (Statement) con.createStatement();
ResultSet rs = stmt.executeQuery(L2QueryNutr);

    String[] strings = new String[1000];
    String temp=new String();
    int count = 0, count1=0;
    int counter=0;

  while (rs.next()) {
  //pairnw ta dedomena me vasi to onoma tis stilis
  String s = rs.getString("ndb_no");

  strings[count]=s;



  if(count==0)
    {
      temp=strings[count];
      L2IDs[count1]=temp;
      count1++;
    }
  if (!temp.equals(strings[count])){
    temp=strings[count];
    L2IDs[count1]=temp;
    count1++;
    counter++;
  }
   if(rs.getString("Nutr_no").equals("208"))//calories
  {
     L2Nutr[counter][0]=rs.getDouble("Nutr_Val");

  }
```

```
        else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
        {
           if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              L2Nutr[counter][0]=0.0;
        }

        if (rs.getString("Nutr_no").equals("205"))//carbs
        {

              L2Nutr[counter][1]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              L2Nutr[counter][1]=0.0;
        }
        if (rs.getString("Nutr_no").equals("204"))//fat
        {

              L2Nutr[counter][2]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              L2Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
           L2Nutr[counter][3]=rs.getDouble("Nutr_Val");
        }
```

```
            else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
            {
                if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                  L2Nutr[counter][3]=0.0;
            }

             if(prefered!=0)
            {
               if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
               {
                   L2Nutr[counter][4]=rs.getDouble("Nutr_Val");
               }
                else  if(!rs.getString("Nutr_no").equals("208")  &&  !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
               {
                  L2Nutr[counter][4]=0.0;
               }
            }


            count++;
            }
            System.out.println("Lunch 2 Products found: "+counter);
            L2Count=counter;



        } catch (SQLException e) {
           System.err.println(e);
        }
```

```
            //Get Weight_file Info for Dairy Milk

        L2Amount=new Double [L2Count+1][15];
        L2Weight=new Double [L2Count+1][15];
        L2Measures=new String [L2Count+1][15];



            try {
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

        Statement stmt = (Statement) con.createStatement();
        //
        ResultSet rs = stmt.executeQuery(L2QueryWeight);

        int count=0;
        int counter1=0, counter2=0;

        String temp=new String();

        //
        while (rs.next()) {
           //pairnw ta dedomena me vasi to onoma tis stilis
           String s = rs.getString("ndb_no");
              if(count==0)
              {
                 temp=s;



              }


           if (!temp.equals(s)){
```

```
          temp=s;

          counter2=0;
        }
        for(int i=0; L2IDs[i]!=null; i++)
        {
         if(L2IDs[i].equals(temp)){
           counter1=i;
           break;
         }
        }



        if(counter1<L2Count){
           L2Amount[counter1][counter2]=rs.getDouble("Amount");
           L2Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
           L2Measures[counter1][counter2]=rs.getString("Msre_Desc");
        }
        counter2++;
        count++;


     }


} catch (SQLException e) {
   System.err.println(e);
}




   //ipologismos nutrients ana grammario
```

```
double[][] L2NutrValPerGram=new double [L2Count+1][5];
for(int i=0; i<L2Count+1; i++)
   for(int j=0; j<5; j++)
   {
      if(L2Nutr[i][j]!=0)
         L2NutrValPerGram[i][j]=L2Nutr[i][j]/100;
      else
         L2NutrValPerGram[i][j]=0.0;
   }

//metatropi timwn nutrients se megala integers gia skopous algorithmou
//ikanopoiisis periorismwn

L2IntNutr=new int[L2Count+1][5];

for(int i=0; i<L2Count+1; i++){
   for(int j=0; j<5; j++){
      L2IntNutr[i][j]=(int)(L2NutrValPerGram[i][j]*1000);

   }

}

programString+="\n\nint: L2Count;"
      +"\nint: minL2;"
      +"\nint: maxL2;"
      +"\nset of int: L2Items= 1..L2Count;"
      +"\narray[L2Items] of int: L2Calories;"
      +"\narray[L2Items] of int: L2Carbohydrate;"
      +"\narray[L2Items] of int: L2Fat;"
      +"\narray[L2Items] of int: L2Protein;"
      +"\narray[L2Items] of int: L2Preference;"
```

```
            + "\n\nvar minL2..maxL2: AmountL2;"
            + "\n\nvar 1..L2Count: l2Amount;"
            + "\nconstraint (L2Calories[l2Amount]!=0);";



          tempConstraint1+="+(L2Calories[l2Amount]*AmountL2*30)";
          tempConstraint2+="+(L2Carbohydrate[l2Amount]*AmountL2*30)";
          tempConstraint3+="+(L2Fat[l2Amount]*AmountL2*30)";
          tempConstraint4+="+(L2Protein[l2Amount]*AmountL2*30)";
          if(prefered!=0)
          tempConstraint5+="+(L2Preference[l2Amount]*AmountL2*30)";


          outputString+="\n++\"L2 Amount: \", show(AmountL2), \" in pos: \",show(l2Amount),
\"\\n\"";
      }
        if(l3_choice==1){
         L3IDs = new String[2000];
         double [][]L3Nutr=new double[2000][5];

          for(int i=0;i< L3Count+1; i++){
           for(int j=0; j<5; j++)
              L3Nutr[i][j]=0.0;
        }

         //Initialize Poultry Products
         try {

        Connection                                    con                                    =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

        Statement stmt = (Statement) con.createStatement();
        ResultSet rs = stmt.executeQuery(L3QueryNutr);
```

```
            String[] strings = new String[5000];
            String temp=new String();
            int count = 0, count1=0;
            int counter=0;

        while (rs.next()) {
        //pairnw ta dedomena me vasi to onoma tis stilis
        String s = rs.getString("ndb_no");

        strings[count]=s;
          if(count==0)
            {
               temp=strings[count];
               L3IDs[count1]=temp;
               count1++;
            }
        if (!temp.equals(strings[count])){
          temp=strings[count];
          L3IDs[count1]=temp;
          count1++;
          counter++;
        }
          if(rs.getString("Nutr_no").equals("208"))//calories
        {
            L3Nutr[counter][0]=rs.getDouble("Nutr_Val");

        }
        else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
               L3Nutr[counter][0]=0.0;
        }
```

```
        if (rs.getString("Nutr_no").equals("205"))//carbs
        {

            L3Nutr[counter][1]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
        {
           if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
             L3Nutr[counter][1]=0.0;
        }
        if (rs.getString("Nutr_no").equals("204"))//fat
        {

            L3Nutr[counter][2]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
           if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
             L3Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
          L3Nutr[counter][3]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
        {
           if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
             L3Nutr[counter][3]=0.0;
        }
```

```
            if(prefered!=0)
            {
               if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
               {
                   L3Nutr[counter][4]=rs.getDouble("Nutr_Val");
               }
                else  if(!rs.getString("Nutr_no").equals("208")  &&  !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
               {
                  L3Nutr[counter][4]=0.0;
               }
            }
            count++;
            }
            L3Count=counter;
            System.out.println("Lunch 3 Products found: "+counter);

      } catch (SQLException e) {
         System.err.println(e);
      }


                        //Get Weight_file Info for Poultry

         L3Amount=new Double [L3Count+1][15];
         L3Weight=new Double [L3Count+1][15];
         L3Measures=new String [L3Count+1][15];


             try {
         Connection                                    con                                    =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

         Statement stmt = (Statement) con.createStatement();
```

```
//
ResultSet rs = stmt.executeQuery(L3QueryWeight);

int count=0;
int counter1=0, counter2=0;

String temp=new String();

//
while (rs.next()) {
   //pairnw ta dedomena me vasi to onoma tis stilis
   String s = rs.getString("ndb_no");
      if(count==0)
      {
         temp=s;


      }


   if (!temp.equals(s)){

     temp=s;

     counter1++;
     counter2=0;
   }
   for(int i=0; L3IDs[i]!=null; i++)
   {
    if(L3IDs[i].equals(temp)){
       counter1=i;
       break;
    }
   }
```

```
        if(counter1<L3Count){
            L3Amount[counter1][counter2]=rs.getDouble("Amount");
            L3Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
            L3Measures[counter1][counter2]=rs.getString("Msre_Desc");
        }
        counter2++;
        count++;


    }


} catch (SQLException e) {
    System.err.println(e);
}


    //ipologismos nutrients ana grammario

    double[][] L3NutrValPerGram=new double [L3Count+1][5];
    for(int i=0; i<L3Count+1; i++)
        for(int j=0; j<5; j++)
        {
            if(L3Nutr[i][j]!=0)
                L3NutrValPerGram[i][j]=L3Nutr[i][j]/100;
            else
                L3NutrValPerGram[i][j]=0.0;
        }

    //metatropi timwn nutrients se megala integers gia skopous algorithmou
    //ikanopoiisis periorismwn

    L3IntNutr=new int[L3Count+1][5];
```

```
for(int i=0; i<L3Count+1; i++){
   for(int j=0; j<5; j++){
      L3IntNutr[i][j]=(int)(L3NutrValPerGram[i][j]*1000);

   }

}



programString+="\n\nint: L3Count;"
      +"\nint: minL3;"
      +"\nint: maxL3;"
      +"\nset of int: L3Items= 1..L3Count;"
      +"\narray[L3Items] of int: L3Calories;"
      +"\narray[L3Items] of int: L3Carbohydrate;"
      +"\narray[L3Items] of int: L3Fat;"
      +"\narray[L3Items] of int: L3Protein;"
      +"\narray[L3Items] of int: L3Preference;"

      + "\n\nvar minL3..maxL3: AmountL3;"
      + "\n\nvar 1..L3Count: l3Amount;"
      + "\nconstraint (L3Calories[l3Amount]!=0);";



   tempConstraint1+="+(L3Calories[l3Amount]*AmountL3*30)";
   tempConstraint2+="+(L3Carbohydrate[l3Amount]*AmountL3*30)";
   tempConstraint3+="+(L3Fat[l3Amount]*AmountL3*30)";
   tempConstraint4+="+(L3Protein[l3Amount]*AmountL3*30)";
   if(prefered!=0)
   tempConstraint5+="+(L3Preference[l3Amount]*AmountL3*30)";
```

```
            outputString+="\n++\"L3 Amount: \", show(AmountL3), \" in pos: \",show(l3Amount),
\"\\n\"";
          }

          if(l4_choice==1){
           L4IDs = new String[2000];
           double [][]L4Nutr=new double[2000][5];


            for(int i=0;i< L4Count+1; i++){
             for(int j=0; j<5; j++)
                L4Nutr[i][j]=0.0;
          }

       //Initialize Pork Products
           try {

          Connection                                    con                                    =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

          Statement stmt = (Statement) con.createStatement();
          ResultSet rs = stmt.executeQuery(L4QueryNutr);

                String[] strings = new String[7000];
                String temp=new String();
                int count = 0, count1=0;
                int counter=0;

             while (rs.next()) {
             //pairnw ta dedomena me vasi to onoma tis stilis
             String s = rs.getString("ndb_no");

             strings[count]=s;
              if(count==0)
```

```
            {
                temp=strings[count];
                L4IDs[count1]=temp;
                count1++;
            }
          if (!temp.equals(strings[count])){
            temp=strings[count];
            L4IDs[count1]=temp;
            count1++;
            counter++;
          }
           if(rs.getString("Nutr_no").equals("208"))//calories
          {
             L4Nutr[counter][0]=rs.getDouble("Nutr_Val");

          }
          else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
          {
             if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                L4Nutr[counter][0]=0.0;
          }

          if (rs.getString("Nutr_no").equals("205"))//carbs
          {

                L4Nutr[counter][1]=rs.getDouble("Nutr_Val");
          }
          else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
          {
              if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                L4Nutr[counter][1]=0.0;
          }
```

```
        if (rs.getString("Nutr_no").equals("204"))//fat
        {

            L4Nutr[counter][2]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              L4Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
           L4Nutr[counter][3]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              L4Nutr[counter][3]=0.0;
        }

         if(prefered!=0)
        {
           if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
           {
               L4Nutr[counter][4]=rs.getDouble("Nutr_Val");
           }
            else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
           {
              L4Nutr[counter][4]=0.0;
           }
        }
```

```
            count++;
            }

            System.out.println("Lunch 4 Products found: "+counter);
            L4Count=counter;


        } catch (SQLException e) {
           System.err.println(e);
        }

                              //Get Weight_file Info for Pork

           L4Amount=new Double [L4Count+1][15];
           L4Weight=new Double [L4Count+1][15];
           L4Measures=new String [L4Count+1][15];


              try {
           Connection                                    con                                    =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

           Statement stmt = (Statement) con.createStatement();

           ResultSet rs = stmt.executeQuery(L4QueryWeight);

           int count=0;
           int counter1=0, counter2=0;

           String temp=new String();


           while (rs.next()) {
              //pairnw ta dedomena me vasi to onoma tis stilis
```

```
  String s = rs.getString("ndb_no");
     if(count==0)
     {
        temp=s;


     }


  if (!temp.equals(s)){

    temp=s;

    counter2=0;
  }
  for(int i=0; L4IDs[i]!=null; i++)
  {
   if(L4IDs[i].equals(temp)){
     counter1=i;
     break;
   }
  }


  if(counter1<L4Count){
     L4Amount[counter1][counter2]=rs.getDouble("Amount");
     L4Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
     L4Measures[counter1][counter2]=rs.getString("Msre_Desc");
  }
  counter2++;
  count++;


}
```

```
} catch (SQLException e) {
   System.err.println(e);
}


            //ipologismos nutrients ana grammario

   double[][] L4NutrValPerGram=new double [L4Count+1][5];
   for(int i=0; i<L4Count+1; i++)
      for(int j=0; j<5; j++)
      {
         if(L4Nutr[i][j]!=0)
            L4NutrValPerGram[i][j]=L4Nutr[i][j]/100;
         else
            L4NutrValPerGram[i][j]=0.0;
      }

   //metatropi timwn nutrients se megala integers gia skopous algorithmou
   //ikanopoiisis periorismwn

   L4IntNutr=new int[L4Count+1][5];


   for(int i=0; i<L4Count+1; i++){
      for(int j=0; j<5; j++){
         L4IntNutr[i][j]=(int)(L4NutrValPerGram[i][j]*1000);


      }

   }
```

```
programString+="\n\nint: L4Count;"
      +"\nint: minL4;"
      +"\nint: maxL4;"
      +"\nset of int: L4Items= 1..L4Count;"
      +"\narray[L4Items] of int: L4Calories;"
      +"\narray[L4Items] of int: L4Carbohydrate;"
      +"\narray[L4Items] of int: L4Fat;"
      +"\narray[L4Items] of int: L4Protein;"
      +"\narray[L4Items] of int: L4Preference;"

      + "\n\nvar minL4..maxL4: AmountL4;"
      + "\n\nvar 1..L4Count: l4Amount;"
      + "\nconstraint (L4Calories[l4Amount]!=0);";



   tempConstraint1+="+(L4Calories[l4Amount]*AmountL4*30)";
   tempConstraint2+="+(L4Carbohydrate[l4Amount]*AmountL4*30)";
   tempConstraint3+="+(L4Fat[l4Amount]*AmountL4*30)";
   tempConstraint4+="+(L4Protein[l4Amount]*AmountL4*30)";
   if(prefered!=0)
      tempConstraint5+="+(L4Preference[l4Amount]*AmountL4*30)";


   outputString+="\n++\"L4 Amount: \", show(AmountL4), \" in pos: \",show(l4Amount),
\"\\n\"";
  }

 if(l5_choice!=0){
  L5IDs = new String[2000];
  double [][]L5Nutr=new double[2000][5];
```

```
          for(int i=0;i< L5Count+1; i++){
           for(int j=0; j<5; j++)
              L5Nutr[i][j]=0.0;
         }

     //Initialize Pork Products
         try {

        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

        Statement stmt = (Statement) con.createStatement();
        ResultSet rs = stmt.executeQuery(L5QueryNutr);

             String[] strings = new String[7000];
             String temp=new String();
             int count = 0, count1=0;
             int counter=0;

           while (rs.next()) {
           //pairnw ta dedomena me vasi to onoma tis stilis
           String s = rs.getString("ndb_no");

           strings[count]=s;
            if(count==0)
             {
                temp=strings[count];
                L5IDs[count1]=temp;
                count1++;
             }
           if (!temp.equals(strings[count])){
             temp=strings[count];
             L5IDs[count1]=temp;
             count1++;
```

```
            counter++;
          }
           if(rs.getString("Nutr_no").equals("208"))//calories
          {
             L5Nutr[counter][0]=rs.getDouble("Nutr_Val");

          }
          else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
          {
             if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                L5Nutr[counter][0]=0.0;
          }

          if (rs.getString("Nutr_no").equals("205"))//carbs
          {

                L5Nutr[counter][1]=rs.getDouble("Nutr_Val");
          }
          else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
          {
              if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                L5Nutr[counter][1]=0.0;
          }
          if (rs.getString("Nutr_no").equals("204"))//fat
          {

                L5Nutr[counter][2]=rs.getDouble("Nutr_Val");
          }
          else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
          {
              if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
```

```
                L5Nutr[counter][2]=0.0;
            }
            if (rs.getString("Nutr_no").equals("203"))//protein
            {
               L5Nutr[counter][3]=rs.getDouble("Nutr_Val");
            }
            else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
            {
                if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                L5Nutr[counter][3]=0.0;
            }

             if(prefered!=0)
            {
               if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
               {
                   L5Nutr[counter][4]=rs.getDouble("Nutr_Val");
               }
                else  if(!rs.getString("Nutr_no").equals("208")  &&  !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
               {
                  L5Nutr[counter][4]=0.0;
               }
            }
            count++;
            }

            System.out.println("Lunch 4 Products found: "+counter);
            L5Count=counter;



        } catch (SQLException e) {
```

```
            System.err.println(e);
        }

                                //Get Weight_file Info for Pork

            L5Amount=new Double [L5Count+1][15];
            L5Weight=new Double [L5Count+1][15];
            L5Measures=new String [L5Count+1][15];


                try {
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

            Statement stmt = (Statement) con.createStatement();

            ResultSet rs = stmt.executeQuery(L5QueryWeight);

            int count=0;
            int counter1=0, counter2=0;

            String temp=new String();


            while (rs.next()) {
               //pairnw ta dedomena me vasi to onoma tis stilis
               String s = rs.getString("ndb_no");
                  if(count==0)
                  {
                     temp=s;


                  }
```

```
        if (!temp.equals(s)){

          temp=s;

          counter2=0;
        }
        for(int i=0; L5IDs[i]!=null; i++)
        {
         if(L5IDs[i].equals(temp)){
            counter1=i;
            break;
         }
        }
       if(counter1<L5Count){
            L5Amount[counter1][counter2]=rs.getDouble("Amount");
            L5Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
            L5Measures[counter1][counter2]=rs.getString("Msre_Desc");
        }
        counter2++;
        count++;


      }


} catch (SQLException e) {
   System.err.println(e);
}

              //ipologismos nutrients ana grammario

   double[][] L5NutrValPerGram=new double [L5Count+1][5];
   for(int i=0; i<L5Count+1; i++)
```

```
   for(int j=0; j<5; j++)
   {
      if(L5Nutr[i][j]!=0)
         L5NutrValPerGram[i][j]=L5Nutr[i][j]/100;
      else
         L5NutrValPerGram[i][j]=0.0;
   }

//metatropi timwn nutrients se megala integers gia skopous algorithmou
//ikanopoiisis periorismwn

L5IntNutr=new int[L5Count+1][5];

for(int i=0; i<L5Count+1; i++){
   for(int j=0; j<5; j++){
      L5IntNutr[i][j]=(int)(L5NutrValPerGram[i][j]*1000);

   }

}



programString+="\n\nint: L5Count;"
      +"\nint: minL5;"
      +"\nint: maxL5;"
      +"\nset of int: L5Items= 1..L5Count;"
      +"\narray[L5Items] of int: L5Calories;"
      +"\narray[L5Items] of int: L5Carbohydrate;"
      +"\narray[L5Items] of int: L5Fat;"
      +"\narray[L5Items] of int: L5Protein;"
      +"\narray[L5Items] of int: L5Preference;"

      + "\n\nvar minL5..maxL5: AmountL5;"
      + "\n\nvar 1..L5Count: l5Amount;"
```

```
            + "\nconstraint (L5Calories[l5Amount]!=0);";

            tempConstraint1+="+(L5Calories[l5Amount]*AmountL5*30)";
            tempConstraint2+="+(L5Carbohydrate[l5Amount]*AmountL5*30)";
            tempConstraint3+="+(L5Fat[l5Amount]*AmountL5*30)";
            tempConstraint4+="+(L5Protein[l5Amount]*AmountL5*30)";
            if(prefered!=0)
            tempConstraint5+="+(L5Preference[l5Amount]*AmountL5*30)";

            outputString+="\n++\"L5 Amount: \", show(AmountL5), \" in pos: \",show(l5Amount),
\"\\n\"";
          }
        }


        //gia t dinner stelnw panta t d1_choice
        System.out.println("D1 Choice: "+d1_choice+"");


        //ean t d1={7..10} tote stelnw kai d3_choice, d4_choice, d5_choice
        if(d1_choice==7 || d1_choice==8 || d1_choice==9 || d1_choice==10 ||d1_choice==0 )
        {
         if(d2_choice!=0){
            D2IDs = new String[2000];
            double [][]D2Nutr=new double[2000][5];

             for(int i=0;i< D2Count+1; i++){
              for(int j=0; j<5; j++)
                 D2Nutr[i][j]=0.0;
            }

      //Initialize Beef Products
        try {
```

```
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

Statement stmt = (Statement) con.createStatement();
ResultSet rs = stmt.executeQuery(D2QueryNutr);

    String[] strings = new String[5000];
    String temp=new String();
    int count = 0, count1=0;
    int counter=0;

  while (rs.next()) {
  //pairnw ta dedomena me vasi to onoma tis stilis
  String s = rs.getString("ndb_no");

  strings[count]=s;
   if(count==0)
    {
      temp=strings[count];
      D2IDs[count1]=temp;
      count1++;
    }
  if (!temp.equals(strings[count])){
    temp=strings[count];
    D2IDs[count1]=temp;
    count1++;
    counter++;
  }
      if(rs.getString("Nutr_no").equals("208"))//calories
  {
     D2Nutr[counter][0]=rs.getDouble("Nutr_Val");

  }
```

```
        else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
        {
           if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              D2Nutr[counter][0]=0.0;
        }

        if (rs.getString("Nutr_no").equals("205"))//carbs
        {

              D2Nutr[counter][1]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              D2Nutr[counter][1]=0.0;
        }
        if (rs.getString("Nutr_no").equals("204"))//fat
        {

              D2Nutr[counter][2]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              D2Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
           D2Nutr[counter][3]=rs.getDouble("Nutr_Val");
        }
```

```
            else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
            {
                if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                  D2Nutr[counter][3]=0.0;
            }

             if(prefered!=0)
            {
               if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
               {
                   D2Nutr[counter][4]=rs.getDouble("Nutr_Val");
               }
                else  if(!rs.getString("Nutr_no").equals("208")  &&  !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
               {
                  D2Nutr[counter][4]=0.0;
               }
            }
            count++;
            }
            D2Count=counter;
            System.out.println("Dinner 1 Products found: "+counter);

        } catch (SQLException e) {
           System.err.println(e);
       }



                 //Get Weight_file Info for Beef

           D2Amount=new Double [D2Count+1][15];
           D2Weight=new Double [D2Count+1][15];
```

```
D2Measures=new String [D2Count+1][15];


    try {
Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

Statement stmt = (Statement) con.createStatement();

ResultSet rs = stmt.executeQuery(D2QueryWeight);

int count=0;
int counter1=0, counter2=0;

String temp=new String();

//
while (rs.next()) {
  //pairnw ta dedomena me vasi to onoma tis stilis
  String s = rs.getString("ndb_no");
    if(count==0)
    {
      temp=s;


    }


  if (!temp.equals(s)){

    temp=s;

    counter2=0;
  }
```

```
      for(int i=0; D2IDs[i]!=null; i++)
      {
       if(D2IDs[i].equals(temp)){
          counter1=i;
          break;
       }
      }



      if(counter1<D2Count){
          D2Amount[counter1][counter2]=rs.getDouble("Amount");
          D2Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
          D2Measures[counter1][counter2]=rs.getString("Msre_Desc");
      }
      counter2++;
      count++;


    }


} catch (SQLException e) {
   System.err.println(e);
}



   //ipologismos nutrients ana grammario

   double[][] D2NutrValPerGram=new double [D2Count+1][5];
   for(int i=0; i<D2Count+1; i++)
      for(int j=0; j<5; j++)
```

```
   {
      if(D2Nutr[i][j]!=0)
         D2NutrValPerGram[i][j]=D2Nutr[i][j]/100;
      else
         D2NutrValPerGram[i][j]=0.0;
   }

//metatropi timwn nutrients se megala integers gia skopous algorithmou
//ikanopoiisis periorismwn

D2IntNutr=new int[D2Count+1][5];


for(int i=0; i<D2Count+1; i++){
   for(int j=0; j<5; j++){
      D2IntNutr[i][j]=(int)(D2NutrValPerGram[i][j]*1000);

   }

}


programString+="\n\nint: D2Count;"
      +"\nint: minD2;"
      +"\nint: maxD2;"
      +"\nset of int: D2Items= 1..D2Count;"
      +"\narray[D2Items] of int: D2Calories;"
      +"\narray[D2Items] of int: D2Carbohydrate;"
      +"\narray[D2Items] of int: D2Fat;"
      +"\narray[D2Items] of int: D2Protein;"
      +"\narray[D2Items] of int: D2Preference;"

      + "\n\nvar minD2..maxD2: AmountD2;"
      + "\n\nvar 1..D2Count: d2Amount;"
```

```
            + "\nconstraint (D2Calories[d2Amount]!=0);";


            tempConstraint1+="+(D2Calories[d2Amount]*AmountD2*30)";
            tempConstraint2+="+(D2Carbohydrate[d2Amount]*AmountD2*30)";
            tempConstraint3+="+(D2Fat[d2Amount]*AmountD2*30)";
            tempConstraint4+="+(D2Protein[d2Amount]*AmountD2*30)";
            if(prefered!=0)
            tempConstraint5+="+(D2Preference[d2Amount]*AmountD2*30)";

            outputString+="\n++\"D2 Amount: \", show(AmountD2), \" in pos: \",show(d2Amount), \"\\n\"";

        }
          if(d3_choice!=0){
           D3IDs = new String[2000];
           double [][]D3Nutr=new double[2000][5];

            for(int i=0;i< D3Count+1; i++){
              for(int j=0; j<5; j++)
                 D3Nutr[i][j]=0.0;
           }

        //Initialize Finfish & Shellfish Products
        try {

           Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

           Statement stmt = (Statement) con.createStatement();
           ResultSet rs = stmt.executeQuery(D3QueryNutr);

                String[] strings = new String[5000];
                String temp=new String();
```

```
    int count = 0, count1=0;
    int counter=0;

while (rs.next()) {
//pairnw ta dedomena me vasi to onoma tis stilis
String s = rs.getString("ndb_no");

strings[count]=s;
 if(count==0)
   {
      temp=strings[count];
      D3IDs[count1]=temp;
      count1++;
   }
if (!temp.equals(strings[count])){
  temp=strings[count];
  D3IDs[count1]=temp;
  count1++;
  counter++;
}
  if(rs.getString("Nutr_no").equals("208"))//calories
{
   D3Nutr[counter][0]=rs.getDouble("Nutr_Val");

}
else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
{
   if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
      D3Nutr[counter][0]=0.0;
}

if (rs.getString("Nutr_no").equals("205"))//carbs
{
```

```
            D3Nutr[counter][1]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              D3Nutr[counter][1]=0.0;
        }
        if (rs.getString("Nutr_no").equals("204"))//fat
        {

            D3Nutr[counter][2]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              D3Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
           D3Nutr[counter][3]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              D3Nutr[counter][3]=0.0;
        }

         if(prefered!=0)
        {
           if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
```

```
            {
                D3Nutr[counter][4]=rs.getDouble("Nutr_Val");
            }
             else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
            {
               D3Nutr[counter][4]=0.0;
            }
          }
          count++;
          }
          D3Count=counter;
          System.out.println("Dinner 3 Products found: "+counter);

      } catch (SQLException e) {
         System.err.println(e);
      }



              //Get Weight_file Info for Fish

         D3Amount=new Double [D3Count+1][15];
         D3Weight=new Double [D3Count+1][15];
         D3Measures=new String [D3Count+1][15];


            try {
         Connection                                    con                                    =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

         Statement stmt = (Statement) con.createStatement();

         ResultSet rs = stmt.executeQuery(D3QueryWeight);
```

```
int count=0;
int counter1=0, counter2=0;

String temp=new String();



while (rs.next()) {
   //pairnw ta dedomena me vasi to onoma tis stilis
   String s = rs.getString("ndb_no");
      if(count==0)
      {
         temp=s;



      }


   if (!temp.equals(s)){

     temp=s;

     counter2=0;
   }

   for(int i=0; D3IDs[i]!=null; i++)
   {
    if(D3IDs[i].equals(temp)){
      counter1=i;
      break;
    }
   }
```

```
        if(counter1<D3Count){
            D3Amount[counter1][counter2]=rs.getDouble("Amount");
            D3Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
            D3Measures[counter1][counter2]=rs.getString("Msre_Desc");
        }
        counter2++;
        count++;


    }


} catch (SQLException e) {
    System.err.println(e);
}



    //ipologismos nutrients ana grammario

    double[][] D3NutrValPerGram=new double [D3Count+1][5];
    for(int i=0; i<D3Count+1; i++)
        for(int j=0; j<5; j++)
        {
            if(D3Nutr[i][j]!=0)
                D3NutrValPerGram[i][j]=D3Nutr[i][j]/100;
            else
                D3NutrValPerGram[i][j]=0.0;
        }

    //metatropi timwn nutrients se megala integers gia skopous algorithmou
    //ikanopoiisis periorismwn

    D3IntNutr=new int[D3Count+1][5];
```

```
for(int i=0; i<D3Count+1; i++){
   for(int j=0; j<5; j++){
      D3IntNutr[i][j]=(int)(D3NutrValPerGram[i][j]*1000);


   }

}



programString+="\n\nint: D3Count;"
      +"\nint: minD3;"
      +"\nint: maxD3;"
      +"\nset of int: D3Items= 1..D3Count;"
      +"\narray[D3Items] of int: D3Calories;"
      +"\narray[D3Items] of int: D3Carbohydrate;"
      +"\narray[D3Items] of int: D3Fat;"
      +"\narray[D3Items] of int: D3Protein;"
      +"\narray[D3Items] of int: D3Preference;"

      + "\n\nvar minD3..maxD3: AmountD3;"
      + "\n\nvar 1..D3Count: d3Amount;"
      + "\nconstraint (D3Calories[d3Amount]!=0);";



   tempConstraint1+="+(D3Calories[d3Amount]*AmountD3*30)";
   tempConstraint2+="+(D3Carbohydrate[d3Amount]*AmountD3*30)";
   tempConstraint3+="+(D3Fat[d3Amount]*AmountD3*30)";
   tempConstraint4+="+(D3Protein[d3Amount]*AmountD3*30)";
   if(prefered!=0)
```

```
            tempConstraint5+="+(D3Preference[d3Amount]*AmountD3*30)";

            outputString+="\n++\"D3 Amount: \", show(AmountD3), \" in pos: \",show(d3Amount), \"\\n\"";
        }
        if(d4_choice==1){
         D4IDs = new String[2000];
          double [][]D4Nutr=new double[2000][5];

           for(int i=0;i< D4Count+1; i++){
              for(int j=0; j<5; j++)
                 D4Nutr[i][j]=0.0;
           }

        //Initialize Legumes & Legume Products
        try {

          Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

          Statement stmt = (Statement) con.createStatement();
          ResultSet rs = stmt.executeQuery(D4QueryNutr);

              String[] strings = new String[5000];
              String temp=new String();
              int count = 0, count1=0;
              int counter=0;

          while (rs.next()) {
          //pairnw ta dedomena me vasi to onoma tis stilis
          String s = rs.getString("ndb_no");

          strings[count]=s;
            if(count==0)
```

```
            {
               temp=strings[count];
               D4IDs[count1]=temp;
               count1++;
            }
          if (!temp.equals(strings[count])){
            temp=strings[count];
            D4IDs[count1]=temp;
            count1++;
            counter++;
          }
             if(rs.getString("Nutr_no").equals("208"))//calories
          {
             D4Nutr[counter][0]=rs.getDouble("Nutr_Val");
//              System.out.println("calories "+B2Nutr[counter][0]);
          }
          else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
          {
             if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
               D4Nutr[counter][0]=0.0;
          }

          if (rs.getString("Nutr_no").equals("205"))//carbs
          {

               D4Nutr[counter][1]=rs.getDouble("Nutr_Val");
          }
          else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
          {
              if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
               D4Nutr[counter][1]=0.0;
          }
```

```
        if (rs.getString("Nutr_no").equals("204"))//fat
        {

            D4Nutr[counter][2]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              D4Nutr[counter][2]=0.0;
        }
        if (rs.getString("Nutr_no").equals("203"))//protein
        {
           D4Nutr[counter][3]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              D4Nutr[counter][3]=0.0;
        }

         if(prefered!=0)
        {
           if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
           {
               D4Nutr[counter][4]=rs.getDouble("Nutr_Val");
           }
            else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
           {
              D4Nutr[counter][4]=0.0;
           }
        }
```

```
            count++;
            }
            D4Count=counter;
            System.out.println("Dinner 4 Products found: "+counter);

        } catch (SQLException e) {
           System.err.println(e);
        }




                //Get Weight_file Info for Legumes

           D4Amount=new Double [D4Count+1][15];
           D4Weight=new Double [D4Count+1][15];
           D4Measures=new String [D4Count+1][15];


              try {
           Connection                                    con                                    =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

           Statement stmt = (Statement) con.createStatement();
           //
           ResultSet rs = stmt.executeQuery(D4QueryWeight);

           int count=0;
           int counter1=0, counter2=0;

           String temp=new String();

           //
           while (rs.next()) {
```

```
//pairnw ta dedomena me vasi to onoma tis stilis
String s = rs.getString("ndb_no");
   if(count==0)
   {
      temp=s;



   }


if (!temp.equals(s)){
  //tempDairy1[counter]=Integer.parseInt(strings[i]);
  temp=s;

  counter2=0;
}

for(int i=0; D4IDs[i]!=null; i++)
{
 if(D4IDs[i].equals(temp)){
   counter1=i;
   break;
 }
}



if(counter1<D4Count){
   D4Amount[counter1][counter2]=rs.getDouble("Amount");
   D4Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
   D4Measures[counter1][counter2]=rs.getString("Msre_Desc");
}
counter2++;
count++;
```

```
    }

} catch (SQLException e) {
   System.err.println(e);
}


   //ipologismos nutrients ana grammario

   double[][] D4NutrValPerGram=new double [D4Count+1][5];
   for(int i=0; i<D4Count+1; i++)
      for(int j=0; j<5; j++)
      {
         if(D4Nutr[i][j]!=0)
            D4NutrValPerGram[i][j]=D4Nutr[i][j]/100;
         else
            D4NutrValPerGram[i][j]=0.0;
      }

   //metatropi timwn nutrients se megala integers gia skopous algorithmou
   //ikanopoiisis periorismwn

   D4IntNutr=new int[D4Count+1][5];

   for(int i=0; i<D4Count+1; i++){
      for(int j=0; j<5; j++){
         D4IntNutr[i][j]=(int)(D4NutrValPerGram[i][j]*1000);


      }
```

```
        }


        programString+="\n\nint: D4Count;"
              +"\nint: minD4;"
              +"\nint: maxD4;"
              +"\nset of int: D4Items= 1..D4Count;"
              +"\narray[D4Items] of int: D4Calories;"
              +"\narray[D4Items] of int: D4Carbohydrate;"
              +"\narray[D4Items] of int: D4Fat;"
              +"\narray[D4Items] of int: D4Protein;"
              +"\narray[D4Items] of int: D4Preference;"

              + "\n\nvar minD4..maxD4: AmountD4;"
              + "\n\nvar 1..D4Count: d4Amount;"
              + "\nconstraint (D4Calories[d4Amount]!=0);";


               tempConstraint1+="+(D4Calories[d4Amount]*AmountD4*30)";
            tempConstraint2+="+(D4Carbohydrate[d4Amount]*AmountD4*30)";
            tempConstraint3+="+(D4Fat[d4Amount]*AmountD4*30)";
            tempConstraint4+="+(D4Protein[d4Amount]*AmountD4*30)";
            if(prefered!=0)
            tempConstraint5+="+(D4Preference[d4Amount]*AmountD4*30)";

            outputString+="\n++\"D4 Amount: \", show(AmountD4), \" in pos: \",show(d4Amount),
\"\\n\"";
       }
        if(d5_choice==1){
        D5IDs = new String[2000];
        double [][]D5Nutr=new double[2000][5];

         for(int i=0;i< D5Count+1; i++){
            for(int j=0; j<5; j++)
```

```
            D5Nutr[i][j]=0.0;
        }

    //Initialize Lamp, Veal and Game Products
    try {

      Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");

      Statement stmt = (Statement) con.createStatement();
      ResultSet rs = stmt.executeQuery(D5QueryNutr);

          String[] strings = new String[7000];
          String temp=new String();
          int count = 0, count1=0;
          int counter=0;

        while (rs.next()) {
        //pairnw ta dedomena me vasi to onoma tis stilis
        String s = rs.getString("ndb_no");

        strings[count]=s;
         if(count==0)
          {
            temp=strings[count];
            D5IDs[count1]=temp;
            count1++;
          }
        if (!temp.equals(strings[count])){
          temp=strings[count];
          D5IDs[count1]=temp;
          count1++;
          counter++;
        }
```

```
            if(rs.getString("Nutr_no").equals("208"))//calories
        {
           D5Nutr[counter][0]=rs.getDouble("Nutr_Val");
//              System.out.println("calories "+B2Nutr[counter][0]);
        }
        else if(!rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203"))
        {
           if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              D5Nutr[counter][0]=0.0;
        }

        if (rs.getString("Nutr_no").equals("205"))//carbs
        {

              D5Nutr[counter][1]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              D5Nutr[counter][1]=0.0;
        }
        if (rs.getString("Nutr_no").equals("204"))//fat
        {

              D5Nutr[counter][2]=rs.getDouble("Nutr_Val");
        }
        else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("205") &&
!rs.getString("Nutr_no").equals("203") )
        {
            if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
              D5Nutr[counter][2]=0.0;
        }
```

```
            if (rs.getString("Nutr_no").equals("203"))//protein
            {
               D5Nutr[counter][3]=rs.getDouble("Nutr_Val");
            }
            else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204") &&
!rs.getString("Nutr_no").equals("205") )
            {
                if(prefered!=0 && !rs.getString("Nutr_no").equals(Integer.toString(prefered) ))
                  D5Nutr[counter][3]=0.0;
            }

             if(prefered!=0)
            {
               if (rs.getString("Nutr_no").equals(Integer.toString(prefered)))//prefered nutrient
               {
                   D5Nutr[counter][4]=rs.getDouble("Nutr_Val");
               }
                else if(!rs.getString("Nutr_no").equals("208") && !rs.getString("Nutr_no").equals("204")
&& !rs.getString("Nutr_no").equals("205") && !rs.getString("Nutr_no").equals("203") )
               {
                  D5Nutr[counter][4]=0.0;
               }
            }
            count++;
            }
            D5Count=counter;
            System.out.println("Dinner 5 Products found: "+counter);

      } catch (SQLException e) {
         System.err.println(e);
      }
```

```
//Get Weight_file Info for Lamp

 D5Amount=new Double [D5Count+1][15];
 D5Weight=new Double [D5Count+1][15];
 D5Measures=new String [D5Count+1][15];



    try {
 Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nutrient_data", "root", "MyNewPass");
 //getConnection();
 Statement stmt = (Statement) con.createStatement();
 //
 ResultSet rs = stmt.executeQuery(D5QueryWeight);

 int count=0;
 int counter1=0, counter2=0;

 String temp=new String();

 //
 while (rs.next()) {
   //pairnw ta dedomena me vasi to onoma tis stilis
   String s = rs.getString("ndb_no");
      if(count==0)
      {
         temp=s;


      }


   if (!temp.equals(s)){
```

```
            //tempDairy1[counter]=Integer.parseInt(strings[i]);
            temp=s;

            counter2=0;
          }

          for(int i=0; D5IDs[i]!=null; i++)
          {
           if(D5IDs[i].equals(temp)){
             counter1=i;
             break;
           }
          }


          if(counter1<D5Count){
             D5Amount[counter1][counter2]=rs.getDouble("Amount");
             D5Weight[counter1][counter2]=rs.getDouble("Gm_Wgt");
             D5Measures[counter1][counter2]=rs.getString("Msre_Desc");
          }
          counter2++;
          count++;


        }


      } catch (SQLException e) {
        System.err.println(e);
      }
//


        //ipologismos nutrients ana grammario
```

```
double[][] D5NutrValPerGram=new double [D5Count+1][5];
for(int i=0; i<D5Count+1; i++)
   for(int j=0; j<5; j++)
   {
      if(D5Nutr[i][j]!=0)
         D5NutrValPerGram[i][j]=D5Nutr[i][j]/100;
      else
         D5NutrValPerGram[i][j]=0.0;
   }

//metatropi timwn nutrients se megala integers gia skopous algorithmou
//ikanopoiisis periorismwn

D5IntNutr=new int[D5Count+1][5];



for(int i=0; i<D5Count+1; i++){
   for(int j=0; j<5; j++){
      D5IntNutr[i][j]=(int)(D5NutrValPerGram[i][j]*1000);

   }

}



programString+="\n\nint: D5Count;"
      +"\nint: minD5;"
      +"\nint: maxD5;"
      +"\nset of int: D5Items= 1..D5Count;"
      +"\narray[D5Items] of int: D5Calories;"
      +"\narray[D5Items] of int: D5Carbohydrate;"
      +"\narray[D5Items] of int: D5Fat;"
      +"\narray[D5Items] of int: D5Protein;"
```

```
            +"\narray[D5Items] of int: D5Preference;"

            + "\n\nvar minD5..maxD5: AmountD5;"
            + "\n\nvar 1..D5Count: d5Amount;"
            + "\nconstraint (D5Calories[d5Amount]!=0);";



            tempConstraint1+="+(D5Calories[d5Amount]*AmountD5*30)";
            tempConstraint2+="+(D5Carbohydrate[d5Amount]*AmountD5*30)";
            tempConstraint3+="+(D5Fat[d5Amount]*AmountD5*30)";
            tempConstraint4+="+(D5Protein[d5Amount]*AmountD5*30)";
            if(prefered!=0)
            tempConstraint5+="+(D5Preference[d5Amount]*AmountD5*30)";


            outputString+="\n++\"D5 Amount: \", show(AmountD5), \" in pos: \",show(d5Amount),
\"\\n\"";
        }



       }


     String maxsolv=tempConstraint1;
       constraintString="\nconstraint "
          +tempConstraint1+";\nconstraint temp[1] < Nutrients[1];\nconstraint "
          + tempConstraint2+";\nconstraint temp[2] < Nutrients[2]::domain;\nconstraint "
          +tempConstraint3+";\nconstraint temp[3] < Nutrients[3]::domain;\nconstraint "
          + tempConstraint4+";\nconstraint temp[4] < Nutrients[4]::domain;\n";

       if(prefered!=0){
          constraintString+="constraint "+tempConstraint5+";\nconstraint temp[5]";
```

```
        if(minmax==0){
        constraintString+=" > ";
        }
        else if(minmax==1){
         constraintString+=" < ";
        }

    constraintString+= "Nutrients[5]::domain;\n";
    }

  programString+="%%%%"
    +"\nset of int: nutrItems=1..nutrCount;"
    +"\narray[nutrItems] of int: Nutrients;"
       + "\narray[nutrItems] of var int: temp;\n"+constraintString
       + "\nsolve maximize temp[1];"
       + "\noutput ["+outputString+"];";

  calories=(int)(Calories*1000);
  carbs=(int)(Carbohydrate*1000);
  fat=(int)(Fat*1000);
  protein=(int)(Protein*1000);
  sodium=(int)(Sodium*1000);


   try{
// Create program file
    FileWriter fstream = new FileWriter("quick-nutrition.mzn");

    BufferedWriter out = new BufferedWriter(fstream);
    out.write(programString);

    //Close the output stream

    out.close();
```

```
    }
    catch (Exception e){//Catch exception if any
    System.err.println("Error: " + e.getMessage());
 }


  }

 private  void CreateDataString(int b1, int b2, int b3, int b4,int b5, int l1, int l2, int l3, int l4,int l5,
        int d1, int d2, int d3, int d4, int d5, int s1, int s2, int s3, int calories,
        int carbs, int fat, int protein, int Preference)
  {
    String fileString=null;
    int [] amount=new int[2];
    if(b1!=0)
    {
       B1randomIndex=getRandomIndexes(B1Count+1);
//          B1randomIndex=getMedIndex(B1Count+1);
       int[][]B1randomFoods=new int[5][5];
        System.out.print("B1 Ids: ");
       for(int i=0; i<B1randomIndex.length; i++){

          System.out.print( B1IDs[B1randomIndex[i]]+" ");
          for(int j=0; j<5; j++)
          {
             B1randomFoods[i][j]=B1IntNutr[B1randomIndex[i]][j];

          }

          }System.out.println();
          amount=returnAmount(b1, "B1");
          fileString=CreateDataString1(B1randomFoods.length,  B1randomFoods,  "B1",  amount[0],
amount[1],  Prefered);
    }
```

```
        if(b2!=0)
        {
           B2randomIndex=getRandomIndexes(B2Count+1);
//           B2randomIndex=getMedIndex(B2Count+1);
           int[][]B2randomFoods=new int[5][5];
             System.out.println("B2 Ids: ");
           for(int i=0; i<B2randomIndex.length; i++){

              System.out.print( B2IDs[B2randomIndex[i]]+" ");
              for(int j=0; j<5; j++)
              {
                 B2randomFoods[i][j]=B2IntNutr[B2randomIndex[i]][j];

              }



           }
         System.out.println();
           amount=returnAmount(b2, "B2");
           fileString+="\n"+CreateDataString1(B2randomFoods.length,          B2randomFoods,          "B2",
amount[0], amount[1], Prefered);
        }
        if(b3!=0)
        {
           B3randomIndex=getRandomIndexes(B3Count+1);
//            B3randomIndex=getMedIndex(B3Count+1);
           int[][]B3randomFoods=new int[5][5];
             System.out.println("B3 Ids: ");
              for(int i=0; i<B3randomIndex.length; i++){
                   System.out.print( B3IDs[B3randomIndex[i]]+" ");
                 for(int j=0; j<5; j++)
                 {
                    B3randomFoods[i][j]=B3IntNutr[B3randomIndex[i]][j];
```

```
            }

          }
              System.out.println();
          amount=returnAmount(b3, "B3");
          fileString+="\n"+CreateDataString1(B3randomFoods.length, B3randomFoods, "B3",
amount[0], amount[1], Prefered);
    }
    if(b4!=0)
    {
       B4randomIndex=getRandomIndexes(B4Count+1);
//          B4randomIndex=getMedIndex(B4Count+1);
       int[][]B4randomFoods=new int[5][5];
          System.out.print("B4 Ids: ");
          for(int i=0; i<B4randomIndex.length; i++){

              System.out.print( B4IDs[B4randomIndex[i]]+" ");
            for(int j=0; j<5; j++)
            {
              B4randomFoods[i][j]=B4IntNutr[B4randomIndex[i]][j];

            }

          }
           System.out.println();
          amount=returnAmount(b4, "B4");
          fileString+="\n"+CreateDataString1(B4randomFoods.length, B4randomFoods, "B4",
amount[0], amount[1], Prefered);
    }
       if(b5!=0)
    {
       B5randomIndex=getRandomIndexes(B5Count+1);
//          B5randomIndex=getMedIndex(B5Count+1);
```

```
        int[][]B5randomFoods=new int[5][5];
          System.out.print("B5 Ids: ");
           for(int i=0; i<B5randomIndex.length; i++){

               System.out.print( B5IDs[B5randomIndex[i]]+" ");
              for(int j=0; j<5; j++)
              {
                 B5randomFoods[i][j]=B5IntNutr[B5randomIndex[i]][j];

              }

           }
            System.out.println();
           amount=returnAmount(b5, "B5");
           fileString+="\n"+CreateDataString1(B5randomFoods.length, B5randomFoods, "B5",
amount[0], amount[1], Prefered);
      }
      if(l1!=0)
      {
         L1randomIndex=getRandomIndexes(L1Count+1);
//          L1randomIndex=getMedIndex(L1Count+1);
        int[][]L1randomFoods=new int[5][5];
          System.out.print("L1 Ids: ");
        for(int i=0; i<L1randomIndex.length; i++){

                            System.out.print( L1IDs[L1randomIndex[i]]+" ");
           for(int j=0; j<5; j++)
           {
              L1randomFoods[i][j]=L1IntNutr[L1randomIndex[i]][j];

           }

        }
          System.out.println();
```

```
        amount=returnAmount(l1, "L1");
        fileString+="\n"+CreateDataString1(L1randomFoods.length, L1randomFoods, "L1", amount[0],
amount[1], Prefered);
      }
      if(l2!=0)
      {
        L2randomIndex=getRandomIndexes(L2Count+1);
//          L2randomIndex=getMedIndex(L2Count+1);
        int[][]L2randomFoods=new int[5][5];
          System.out.print("L2 Ids: ");
           for(int i=0; i<L2randomIndex.length; i++){

                              System.out.print( L2IDs[L2randomIndex[i]]+" ");
             for(int j=0; j<5; j++)
             {
                L2randomFoods[i][j]=L2IntNutr[L2randomIndex[i]][j];

             }

           }
              System.out.println();
           amount=returnAmount(l2, "L2");
           fileString+="\n"+CreateDataString1(L2randomFoods.length,        L2randomFoods,        "L2",
amount[0], amount[1], Prefered);

      }
      if(l3!=0)
      {
        L3randomIndex=getRandomIndexes(L3Count+1);
//          L3randomIndex=getMedIndex(L3Count+1);
        int[][]L3randomFoods=new int[5][5];
          System.out.print("L3 Ids: ");
           for(int i=0; i<L3randomIndex.length; i++){
```

```
                System.out.print( L3IDs[L3randomIndex[i]]+" ");
            for(int j=0; j<5; j++)
            {
               L3randomFoods[i][j]=L3IntNutr[L3randomIndex[i]][j];

            }

          }
           System.out.println();
          amount=returnAmount(l3, "L3");
          fileString+="\n"+CreateDataString1(L3randomFoods.length, L3randomFoods, "L3",
amount[0], amount[1], Prefered);
      }
      if(l4!=0)
      {
         L4randomIndex=getRandomIndexes(L4Count+1);
//          L4randomIndex=getMedIndex(L4Count+1);
         int[][]L4randomFoods=new int[5][5];
           System.out.print("L4 Ids: ");
          for(int i=0; i<L4randomIndex.length; i++){

                    System.out.print( L4IDs[L4randomIndex[i]]+" ");
            for(int j=0; j<5; j++)
            {
               L4randomFoods[i][j]=L4IntNutr[L4randomIndex[i]][j];

            }

          }
            System.out.println();
          amount=returnAmount(l4, "L4");
          fileString+="\n"+CreateDataString1(L4randomFoods.length, L4randomFoods, "L4",
amount[0], amount[1], Prefered);
      }
```

```
        if(l5!=0)
       {
          L5randomIndex=getRandomIndexes(L5Count+1);
//          L5randomIndex=getMedIndex(L5Count+1);
          int[][]L5randomFoods=new int[5][5];
            System.out.print("L5 Ids: ");
             for(int i=0; i<L5randomIndex.length; i++){

                          System.out.print( L5IDs[L5randomIndex[i]]+" ");
               for(int j=0; j<5; j++)
               {
                  L5randomFoods[i][j]=L5IntNutr[L5randomIndex[i]][j];

               }

             }
              System.out.println();
             amount=returnAmount(l5, "L5");
             fileString+="\n"+CreateDataString1(L5randomFoods.length,       L5randomFoods,       "L5",
amount[0], amount[1], Prefered);
       }
       if(d1!=0)
       {
          D1randomIndex=getRandomIndexes(D1Count+1);
//          D1randomIndex=getMedIndex(D1Count+1);
          int[][]D1randomFoods=new int[5][5];
            System.out.print("D1 Ids: ");
             for(int i=0; i<D1randomIndex.length; i++){
                  System.out.print( D1IDs[D1randomIndex[i]]+" ");
               for(int j=0; j<5; j++)
               {
                  D1randomFoods[i][j]=D1IntNutr[D1randomIndex[i]][j];

               }
```

```
            }
                System.out.println();
          amount=returnAmount(d1, "D1");
          fileString+="\n"+CreateDataString1(D1randomFoods.length, D1randomFoods, "D1",
amount[0], amount[1], Prefered);
      }
      if(d2!=0)
      {
          D2randomIndex=getRandomIndexes(D2Count+1);
//          D2randomIndex=getMedIndex(D2Count+1);
          int[][]D2randomFoods=new int[5][5];
           System.out.print("D2 Ids: ");
             for(int i=0; i<D2randomIndex.length; i++){

                 System.out.print( D2IDs[D2randomIndex[i]]+" ");
               for(int j=0; j<5; j++)
               {
                  D2randomFoods[i][j]=D2IntNutr[D2randomIndex[i]][j];

               }

             }
             System.out.println();
             amount=returnAmount(d2, "D2");
             fileString+="\n"+CreateDataString1(D2randomFoods.length, D2randomFoods,
"D2",amount[0], amount[1], Prefered);
      }
      if(d3!=0)
      {
          D3randomIndex=getRandomIndexes(D3Count+1);
//          D3randomIndex=getMedIndex(D3Count+1);
          int[][]D3randomFoods=new int[5][5];
           System.out.print("D3 Ids: ");
```

```
            for(int i=0; i<D3randomIndex.length; i++){

                System.out.print( D3IDs[D3randomIndex[i]]+" ");
               for(int j=0; j<5; j++)
               {
                  D3randomFoods[i][j]=D3IntNutr[D3randomIndex[i]][j];

               }

            }
             System.out.println();
            amount=returnAmount(d3, "D3");
            fileString+="\n"+CreateDataString1(D3randomFoods.length, D3randomFoods, "D3",
amount[0], amount[1], Prefered);
      }
      if(d4!=0)
      {
         D4randomIndex=getRandomIndexes(D4Count+1);
//          D4randomIndex=getMedIndex(D4Count+1);
         int[][]D4randomFoods=new int[5][5];
         System.out.print("D4 Ids: ");
            for(int i=0; i<D4randomIndex.length; i++){

                System.out.print( D4IDs[D4randomIndex[i]]+" ");
               for(int j=0; j<5; j++)
               {
                  D4randomFoods[i][j]=D4IntNutr[D4randomIndex[i]][j];

               }

            }
            System.out.println();
            amount=returnAmount(d4, "D4");
```

```
            fileString+="\n"+CreateDataString1(D4randomFoods.length,      D4randomFoods,      "D4",
amount[0], amount[1], Prefered);
      }
      if(d5!=0)
      {
         D5randomIndex=getRandomIndexes(D5Count+1);
//          D5randomIndex=getMedIndex(D5Count+1);
         int[][]D5randomFoods=new int[5][5];
          System.out.print("D5 Ids: ");
            for(int i=0; i<D5randomIndex.length; i++){

                System.out.print( D5IDs[D5randomIndex[i]]+" ");
              for(int j=0; j<5; j++)
              {
                 D5randomFoods[i][j]=D5IntNutr[D5randomIndex[i]][j];

              }

            }
            System.out.println();
            amount=returnAmount(d5, "D5");
            fileString+="\n"+CreateDataString1(D5randomFoods.length,      D5randomFoods,      "D5",
amount[0], amount[1], Prefered);
      }
      if(s1!=0)
      {
         S1randomIndex=getRandomIndexes(S1Count+1);
//          S1randomIndex=getMedIndex(S1Count+1);
         int[][]S1randomFoods=new int[5][5];
          System.out.print("S1 Ids: ");
            for(int i=0; i<S1randomIndex.length; i++){

                System.out.print( S1IDs[S1randomIndex[i]]+" ");
              for(int j=0; j<5; j++)
```

```
                {
                    S1randomFoods[i][j]=S1IntNutr[S1randomIndex[i]][j];

                }

            }
             System.out.println();
            amount=returnAmount(s1, "S1");
            fileString+="\n"+CreateDataString1(S1randomFoods.length, S1randomFoods, "S1",
amount[0], amount[1], Prefered);
        }
        if(s2!=0)
        {
           S2randomIndex=getRandomIndexes(S2Count+1);
//           S2randomIndex=getMedIndex(S2Count+1);
           int[][]S2randomFoods=new int[5][5];
            System.out.print("S2 Ids: ");
              for(int i=0; i<S2randomIndex.length; i++){
                   System.out.print(S2IDs[S2randomIndex[i]]+" ");
                for(int j=0; j<5; j++)
                {
                    S2randomFoods[i][j]=S2IntNutr[S2randomIndex[i]][j];

                }

            }
            System.out.println();
            amount=returnAmount(s2, "S2");
            fileString+="\n"+CreateDataString1(S2randomFoods.length, S2randomFoods, "S2",
amount[0], amount[1], Prefered);
        }
        if(s3!=0)
        {
           S3randomIndex=getRandomIndexes(S3Count+1);
```

```
//          S3randomIndex=getMedIndex(S3Count+1);
          int[][]S3randomFoods=new int[5][5];
           System.out.print("S3 Ids: ");
             for(int i=0; i<S3randomIndex.length; i++){

                 System.out.print( S3IDs[S3randomIndex[i]]+" ");
               for(int j=0; j<5; j++)
               {
                  S3randomFoods[i][j]=S3IntNutr[S3randomIndex[i]][j];

               }

             }
                 System.out.println();
             amount=returnAmount(s3, "S3");
             fileString+="\n"+CreateDataString1(S3randomFoods.length,       S3randomFoods,       "S3",
amount[0], amount[1], Prefered);
      }
       fileString+="Nutrients = ["+calories+","+carbs+","+fat+","+protein+","+Preference+"];\n";

         try{
       // Create file
         FileWriter fstream = new FileWriter("quick-nutrition.dzn");

         BufferedWriter out = new BufferedWriter(fstream);
         out.write(fileString);

         //Close the output stream

         out.close();
         }
         catch (Exception e){//Catch exception if any
         System.err.println("Error: " + e.getMessage());
         }
```

```
waiting(2);



}
 public static void waiting (int n){

   long t0, t1;

   t0 =  System.currentTimeMillis();

   do{
      t1 = System.currentTimeMillis();
   }
   while ((t1 - t0) < (n * 1000));
}



String ExecuteConstraintProgram(int b1, int b2, int b3, int b4,int b5, int l1, int l2, int l3, int l4,int l5,
      int d1, int d2, int d3, int d4, int d5, int s1, int s2, int s3, int pref)
{
   String output=null;
   String newFile=null;
   CreateDataString(b1, b2, b3, b4,b5, l1, l2, l3, l4,l5,  d1, d2, d3, d4, d5, s1, s2, s3, calories,
      carbs,fat, protein, pref);



   try {

      Process process = Runtime.getRuntime ().exec ("cmd /k command_file.bat");

   } catch (IOException ex) {
      Logger.getLogger(QuickNutrition.class.getName()).log(Level.SEVERE, null, ex);
   }
```

```
//Wait for a few seconds to create new output file

waiting(5);

//read output from file

try {
FileInputStream fstream = new FileInputStream("output.txt");
DataInputStream in = new DataInputStream(fstream);
BufferedReader br = new BufferedReader(new InputStreamReader(in));
String str;
int count=0;
 System.out.println("file read");
while ((str = br.readLine()) != null) {
    System.out.println(str);
   if(str.contains("UNSATISFIABLE"))

   {
      output="UNSATISFIABLE";
      break;
   }
   else
   {
      if(count==0)
         output=str+" ";
      else
         output+=str+" ";
   }
   count++;
}
in.close();
} catch (Exception e) {
System.err.println(e);
```

```
    }


    return output;


}


String filterResultString(String file){
String newFile=null;

String[]splitted=file.split("----------");
System.out.println("filtered output is: "+splitted[splitted.length-2]);

for(int i=0; i<splitted.length; i++)
{
    System.out.println(i+" "+splitted[i]);
}

if(splitted[splitted.length-1].equals("==========")){
    newFile=splitted[splitted.length-2]+" ---------- "+"========== ";
}
else
    newFile=splitted[splitted.length-2]+" ---------- "+"========== ";;




return newFile;
}

String[][] getFoodIDs(String outputText){
```

```
String temp=null;
String []temp1=null;
  String [][]SelectedIds=null;
  Integer []positions;
  String [] mealtags;

  String[]splitted=outputText.split("----------");
  if(splitted.length>2){
  if(splitted[splitted.length-1].contains("==========")){

  temp=splitted[splitted.length-2];

  }
  else
     temp=splitted[splitted.length-2];
  }

  System.out.println();
  temp1=temp.split("\n");
  temp="";
  for(int i=0; i<temp1.length; i++){
     System.out.print(i+" "+temp1[i]);
     if(!temp1[i].equals("\n"))
     temp=temp+temp1[i];
  }
  splitted=temp.split(" ");


for(int i=0; i<splitted.length; i++)
{
  System.out.println(i+" "+splitted[i]);
}
```

```
positions=new Integer[(int)(splitted.length/6)];
mealtags=new String[(int)(splitted.length/6)];
SelectedIds=new String[(int)(splitted.length/6)][2];
 System.out.println("length of pos: "+positions.length);


 for(int i=6, j=0; i<splitted.length; i+=6, j++){
   positions[j]=Integer.parseInt(splitted[i]);

 }

  for(int i=1, j=0; i<splitted.length; i+=6, j++){
   mealtags[j]=splitted[i];

 }


for(int i=0; i<mealtags.length; i++)
{
   if(mealtags[i].equals("B1"))
   {
      SelectedIds[i][0]="B1";
      System.out.println("position of id for breakfast food 1: "+B1randomIndex[positions[i]-1]+"
with food ID: "+B1IDs[B1randomIndex[positions[i]-1]]);
      SelectedIds[i][1]=B1IDs[B1randomIndex[positions[i]-1]];
   }
   else if(mealtags[i].equals("B2"))
   {
      SelectedIds[i][0]="B2";
      System.out.println("position of id for breakfast food 2: "+B2randomIndex[positions[i]-1]+"
with food ID: "+B2IDs[B2randomIndex[positions[i]-1]]);
      SelectedIds[i][1]=B2IDs[B2randomIndex[positions[i]-1]];
```

```
        }
        else if(mealtags[i].equals("B3"))
        {
           SelectedIds[i][0]="B3";
           System.out.println("position of id for breakfast food 3: "+B3randomIndex[positions[i]-1]+"
with food ID: "+B3IDs[B3randomIndex[positions[i]-1]]);
           SelectedIds[i][1]=B3IDs[B3randomIndex[positions[i]-1]];
        }
        else if(mealtags[i].equals("B4"))
        {
           SelectedIds[i][0]="B4";
           System.out.println("position of id for breakfast food 4: "+B4randomIndex[positions[i]-1]+"
with food ID: "+B4IDs[B4randomIndex[positions[i]-1]]);
           SelectedIds[i][1]=B4IDs[B4randomIndex[positions[i]-1]];
        }
         else if(mealtags[i].equals("B5"))
        {
           SelectedIds[i][0]="B5";
           System.out.println("position of id for breakfast food 5: "+B5randomIndex[positions[i]-1]+"
with food ID: "+B5IDs[B5randomIndex[positions[i]-1]]);
           SelectedIds[i][1]=B5IDs[B5randomIndex[positions[i]-1]];
        }
        else if(mealtags[i].equals("L1"))
        {
           SelectedIds[i][0]="L1";
           System.out.println("position of id for lunch food 1: "+L1randomIndex[positions[i]-1]+" with
food ID: "+L1IDs[L1randomIndex[positions[i]-1]]);
           SelectedIds[i][1]=L1IDs[L1randomIndex[positions[i]-1]];
        }
        else if(mealtags[i].equals("L2"))
        {
           SelectedIds[i][0]="L2";
           System.out.println("position of id for lunch food 2: "+L2randomIndex[positions[i]-1]+" with
food ID: "+L2IDs[L2randomIndex[positions[i]-1]]);
```

```
            SelectedIds[i][1]=L2IDs[L2randomIndex[positions[i]-1]];
        }
        else if(mealtags[i].equals("L3"))
        {
            SelectedIds[i][0]="L3";
            System.out.println("position of id for lunch food 3: "+L3randomIndex[positions[i]-1]+" with
food ID: "+L3IDs[L3randomIndex[positions[i]-1]]);
            SelectedIds[i][1]=L3IDs[L3randomIndex[positions[i]-1]];
        }
        else if(mealtags[i].equals("L4"))
        {
            SelectedIds[i][0]="L4";
            System.out.println("position of id for lunch food 4: "+L4randomIndex[positions[i]-1]+" with
food ID: "+L4IDs[L4randomIndex[positions[i]-1]]);
            SelectedIds[i][1]=L4IDs[L4randomIndex[positions[i]-1]];
        }
         else if(mealtags[i].equals("L5"))
        {
            SelectedIds[i][0]="L5";
            System.out.println("position of id for lunch food 5: "+L5randomIndex[positions[i]-1]+" with
food ID: "+L5IDs[L5randomIndex[positions[i]-1]]);
            SelectedIds[i][1]=L5IDs[L5randomIndex[positions[i]-1]];
        }
        else if(mealtags[i].equals("D1"))
        {
            SelectedIds[i][0]="D1";
            System.out.println("position of id for dinner food 1: "+D1randomIndex[positions[i]-1]+"
with food ID: "+D1IDs[D1randomIndex[positions[i]-1]]);
            SelectedIds[i][1]=D1IDs[D1randomIndex[positions[i]-1]];
       }
        else if(mealtags[i].equals("D2"))
        {
            SelectedIds[i][0]="D2";
```

```
            System.out.println("position of id for dinner food 2: "+D2randomIndex[positions[i]-1]+"
with food ID: "+D2IDs[D2randomIndex[positions[i]-1]]);
            SelectedIds[i][1]=D2IDs[D2randomIndex[positions[i]-1]];
        }
        else if(mealtags[i].equals("D3"))
        {
            SelectedIds[i][0]="D3";
            System.out.println("position of id for dinner food 3: "+D3randomIndex[positions[i]-1]+"
with food ID: "+D3IDs[D3randomIndex[positions[i]-1]]);
            SelectedIds[i][1]=D3IDs[D3randomIndex[positions[i]-1]];
        }
        else if(mealtags[i].equals("D4"))
        {
            SelectedIds[i][0]="D4";
            System.out.println("position of id for dinner food 4: "+D4randomIndex[positions[i]-1]+"
with food ID: "+D4IDs[D4randomIndex[positions[i]-1]]);
            SelectedIds[i][1]=D4IDs[D4randomIndex[positions[i]-1]];
        }
        else if(mealtags[i].equals("D5"))
        {
            SelectedIds[i][0]="D5";
            System.out.println("position of id for dinner food 5: "+D5randomIndex[positions[i]-1]+"
with food ID: "+D5IDs[D5randomIndex[positions[i]-1]]);
            SelectedIds[i][1]=D5IDs[D5randomIndex[positions[i]-1]];
        }
        else if(mealtags[i].equals("S1"))
        {
            SelectedIds[i][0]="S1";
            System.out.println("position of id for snack food 1: "+S1randomIndex[positions[i]-1]+" with
food ID: "+S1IDs[S1randomIndex[positions[i]-1]]);
            SelectedIds[i][1]=S1IDs[S1randomIndex[positions[i]-1]];
        }
        else if(mealtags[i].equals("S2"))
        {
```

```
            SelectedIds[i][0]="S2";
            System.out.println("position of id for snack food 2: "+S2randomIndex[positions[i]-1]+" with
food ID: "+S2IDs[S2randomIndex[positions[i]-1]]);
            SelectedIds[i][1]=S2IDs[S2randomIndex[positions[i]-1]];
         }
         else if(mealtags[i].equals("S3"))
         {
            SelectedIds[i][0]="S3";
            System.out.println("position of id for snack food 3: "+S3randomIndex[positions[i]-1]+" with
food ID: "+S3IDs[S3randomIndex[positions[i]-1]]);
            SelectedIds[i][1]=S3IDs[S3randomIndex[positions[i]-1]];
         }
      }
      return SelectedIds;
   }

   Integer[] getFoodAmount(String outputText)
   {
      String temp=null;
      String []temp1=null;
       String[]splitted=outputText.split("----------");
      if(splitted.length>2){
      if(splitted[splitted.length-1].contains("==========")){

      temp=splitted[splitted.length-2];
           }
      else
         temp=splitted[splitted.length-2];
      }

      System.out.println();
      temp1=temp.split("\n");
      temp="";
      for(int i=0; i<temp1.length; i++){
```

```
            System.out.print(i+" "+temp1[i]);
            if(!temp1[i].equals("\n"))
            temp=temp+temp1[i];
        }
        splitted=temp.split(" ");
        Integer [] amounts=new Integer[(int)(splitted.length/6)];
    for(int i=3, j=0; i<splitted.length; i+=6, j++)
        {
            amounts[j]=Integer.parseInt(splitted[i])*30;
        }
            return amounts;
    }
}
```

Δ.14 Κώδικας Log Out

```
    username = null;
        password = null;
        jInternalFrame8.setVisible(false);
        jInternalFrame9.setVisible(true);

            status = 0;
            jLabel126.setText(" ");
            jLabel124.setText("[Log In]");
```