Ατομική Διπλωματική Εργασία

ΜΕΛΕΤΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΔΙΑΧΕΙΡΙΣΗΣ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΣΥΝΕΡΓΕΙΟ ΑΥΤΟΚΙΝΗΤΩΝ

Μαρία Θεοδώρου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2017

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΛΕΤΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΔΙΑΧΕΙΡΙΣΗΣ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΣΥΝΕΡΓΕΙΟ ΑΥΤΟΚΙΝΗΤΩΝ

Μαρία Θεοδώρου

Επιβλέπων Καθηγητής
Παρασκευάς Ευριπίδου

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

Μάιος 2017

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω όσους συνέβαλαν έστω και στο ελάχιστο για την εκπόνηση της παρούσας διπλωματικής εργασίας. Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κ. Παρασκευά Ευριπίδου για την επίβλεψη αυτής της διπλωματικής εργασίας, όπως επίσης και για την ευκαιρία που μου έδωσε να ασχοληθώ με το αντικείμενο αυτό.

Επίσης θέλω να ευχαριστήσω άτομα του στενού φιλικού και οικογενειακού μου περιβάλλοντος για την συμπαράσταση που μου παρείχαν όλο αυτό το διάστημα.

Περίληψη

Το θέμα της διπλωματικής μου ήταν η σχεδίαση και η υλοποίηση ενός συστήματος διαχείρισης δεδομένων (CMS-CRM) για τον εργασιακό χώρο συνεργείο αυτοκινήτων. Η ιδέα μου στηρίχθηκε στο κενό που εντόπισα στο πεδίο του εργασιακού χώρου αυτού, που υπολειπόταν μιας τεχνολογικής υποδομής και εφαρμογής, μιας και όλα γίνονταν με αναχρονιστικές μεθόδους καταχώρησης και διαχείρισης των δεδομένων εργασίας τους. Γνωρίζοντας κάποιους επαγγελματίες του χώρου και αλληλεπιδρώντας μαζί τους σε μία βάση ανάλυσης απαιτήσεων, κατέληξα συμπερασματικά στη δημιουργία αυτού του συστήματος ούτως ώστε, να μπορέσω μέσα από μία προγραμματιστική προσέγγιση και λύση να άρω τα υφιστάμενα προβλήματα στον εργασιακό τους χώρο.

Μετά από μια λεπτομερή και διαλογιστική συζήτηση ανέλυσα και έπειτα κατέληξα στα προβληματικά κενά που υπέφεραν οι επαγγελματίες αυτοί. Μέσα από τη συζήτηση και αφουγκραζόμενη την εμπειρική λογική αυτών των ανθρώπων, ανήγαγα τις ανάγκες και απαιτήσεις αυτών, σε μία τεχνολογική και προγραμματιστική θεώρηση, ούτως ώστε να επέλθω με μια πραγματική λύση η οποία στο εγγύς μέλλον θα επίλυε θεωρητικά και ουσιαστικά τα προβλήματά τους στον εργασιακό χώρο, θα αύξανε την εργονομία και τέλος την ποιότητα ζωής τόσο των ιδιοκτητών όσο και των υπαλλήλων.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Γενική Εισαγωγή

Διαχρονικά σε όλους τους τομείς των υπηρεσιών και γενικά του επιχειρηματικού κόσμου υπήρχε πάντοτε η έννοια της αφαιρετικότητας και της υλοποίησης τεχνικών και εφαρμογών διαμέσου των οποίων να επιτυγχάνεται η καλυτέρευση της ποιότητας των εργαζομένων και γενικότερα της κοινωνίας. Ειδικότερα, η δημιουργία αυτών είχε πάντοτε την επίτευξη της εργονομίας. Έτσι για κάθε πεδίο επιστήμης ή του επιχειρηματικού φάσματος, ο άνθρωπος προσπαθούσε πάντα να ερευνήσει και να δημιουργήσει, έτσι ώστε, να κάνει την ζωή του πιο εύκολη.
Πάμπολλα παραδείγματα ανά τους αιώνες καταμαρτυρούν αυτή τη ανθρώπινη τάση. Αυτό το βλέπουμε από την απαρχή της δημιουργίας ή έστω της ένδειξης μιας οργανωμένης κοινωνίας. Από την εφεύρεση του τροχού, από τις διάφορες αρχιτεκτονικές πρακτικές, όπως αυτό της υγείας, της εκπαίδευσης κτλ. μέχρι τη σημερινή σύγχρονη εποχή, αποτελούν μια συνεχή και εξελισσόμενη κατά τη φύση της πορεία, η οποία κατά το πέρασμα των αιώνων είναι ασταμάτητη και συνεχώς μεταβαλλόμενη με γνώμονα πάντα τις δυναμικές ανάγκες της εποχής.
Μέσα σε αυτή τη εξελικτική πορεία, ιστορική και κεντρική θέση κατέχει η γένεση του Παγκόσμιου Ιστού (World Wide Web). Και αυτό μπορούμε αξιοκρατικά να ισχυριστούμε ότι αποτελεί ένα πρωτοφανές και άνευ προηγουμένου ορόσημο στην ιστορία της ανθρωπότητας, γιατί ήρθε επαναστατικά να αλλάξει ριζικά τον τρόπο ζωής και λειτουργίας όλου του κόσμου. Η ελεύθερη πρόσβαση σε μία τεράστια ποσότητα δεδομένων, η δυνατότητα επικοινωνίας και συνδεσιμότητας όλων των ανθρώπων, αποτέλεσε επαναστατικό σημείο. Κοντολογίς η τεχνολογία αποτέλεσε το εργαλείο και τον τρόπο μέσα από το οποίο, η σύγχρονη εποχή, έχει αυτή τη μορφή σήμερα.

Εμείς σαν φοιτητές του κλάδου Πληροφορικής έχουμε σαν αποστολή να μετέχουμε αυτής της εξέλιξης και να δίνουμε καθημερινά το στίγμα και τις πινελιές στον παγκόσμιο τεχνολογικό καμβά ούτως ώστε και εμείς με τη σειρά μας να καταφέρουμε με τον οποιονδήποτε τρόπο να συμβάλουμε και να παράξουμε γνώση τόσο σε θεωρητικό όσο και σε πρακτικό επίπεδο για το καλό του πλανήτη μας και κάθε συνανθρώπου μας.

1.2 Σκοπός

Σκοπός της παρούσας διπλωματικής εργασίας είναι να εντοπίσει με τον πλέον αποτελεσματικό τρόπο μέσα από την τεχνική της ανάλυσης απαιτήσεων τα προβλήματα που υπάρχουν στον εργασιακό χώρο των επιχειρήσεων επιδιόρθωσης αυτοκινήτων. Η εύρεση και εξιχνίαση των παθογόνων διαδικασιών και η αντικατάσταση τους με σύγχρονες τεχνολογικές λειτουργίες είναι ο μόνος στόχος μας. Το κύριο μέλημα μας είναι η επίτευξη όσο το δυνατό περισσότερο της εργονομίας και της σωστής λειτουργίας της επιχείρησης. Θέλουμε να καταστήσουμε την κάθε επιχείρηση σε αυτό τον τομέα, σύγχρονη και εναρμονισμένη με τα πολλά δεδομένα των ημερών μας έτσι ώστε να μπορούν να ανταποκριθούν στις πολύ απαιτητικές συνθήκες που υπάρχουν στις ημέρες μας.

Κεφάλαιο 2

Ανάλυση Απαιτήσεων

Με την διαδικασία της ανάλυσης απαιτήσεων και διαμέσου μιας συζήτησης με κάποιους επαγγελματίες μηχανικούς και παρατήρησής τους στο περιβάλλον εργασίας τους εξήγαγα και ανέλυσα τις παρακάτω απαιτήσεις.

Οι περισσότεροι εξ αυτών, λόγω άγνοιας και λόγω μη ικανοποιητικών τεχνικών και προγραμμάτων στην καθημερινότητά τους και στην εργασιακή στους ενασχόληση ταλαιπωρούνταν. Τόσο προσωπικά οι ίδιοι από τις απαρχαιωμένες τεχνικές, όσο και οι τυχόν υπάλληλοί τους. Στις περισσότερες περιπτώσεις των επιχειρήσεων, διατηρούσαν και χρησιμοποιούσαν παραδοσιακές και χειρόγραφες τεχνικές για διαχείριση των δεδομένων τους. Αυτές οι απαρχαιωμένες τεχνικές δημιουργούσαν πολλά προβλήματα στις καθημερινές διαδικασίες, καθιστώντας δυσκίνητες και δυσλειτουργικές τις εταιρείες. Ειδικότερα:

2.1 Ασφάλεια Δεδομένων

Όπως όλες οι επιχειρήσεις για την ομαλή λειτουργία τους χρειάζονται μία συστηματική καταγραφή των ευαίσθητων δεδομένων τους σε όλο το φάσμα δράσης τους. Με όσους επαγγελματίες του χώρου ήρθα σε επαφή, εξακρίβωσα ότι η εισαγωγή - καταγραφή των οποιονδήποτε δεδομένων αναφορικά με τα λειτουργικά στοιχεία της εταιρείας καθώς επίσης και των ευαίσθητων προσωπικών δεδομένων των πελατών γίνονταν χειρόγραφα. Έπειτα η αρχειοθέτησή τους γινόταν σε φακέλους με ένα παραδοσιακό σύστημα τοποθέτησης που θα χρησίμευε έπειτα σε μία κάπως πιο εύκολη ανάκτησή τους. Αναμφίβολα ο όγκος των δεδομένων της κάθε επιχείρησης που αναφέρεται στα λεπτά - ευαίσθητα στοιχεία των πελατών της, καθώς και το ιστορικό στη μεταξύ τους δοσοληψία είναι πολύ σημαντικά. Σημαντικό ως προς το περιεχόμενο, καθώς και ως προς τη συνέχεια που πρέπει να υπάρχει προς τη δομή τους. Κανείς έτσι δεν μπορεί να ισχυριστεί με τον παραδοσιακό χειρόγραφο τρόπο καταγραφής τα δεδομένα δεν εκτίθενται σε πολλαπλούς κινδύνους. Τι γίνεται στην περίπτωση μιας καταστροφής; Μιας πυρκαγιάς στην επιχείρηση ή της οποιασδήποτε άλλης φθοράς; Το αποτέλεσμα θα ήταν καταστροφικό, γιατί τα στοιχεία αυτά για να συλλεχθούν και να καταγραφούν ήταν μία διαδικασία επίπονη και μακροχρόνια. Έτσι οι υφιστάμενες μέθοδοι εκθέτουν τα δεδομένα σε πολλούς τέτοιους κινδύνους. Επίσης ένα προκύπτον θέμα ασφάλειας είναι αυτό της υπόκλεψης δεδομένων που πάντα αποτελεί ένα εφιαλτικό σενάριο για τον κάθε επιχειρηματικό οργανισμό. Αυτό είναι ένα από τα σημαντικότερα που αναδύθηκε στην επιφάνεια με την διαδικασία που ακολούθησα, της ανάλυσης απαιτήσεων.

2.2 Χρόνος Απόκρισης

Ο χρόνος απόκρισης και εύρεσης των όποιων δεδομένων της εταιρείας είναι πάρα πολύ σημαντικό θέμα. Για την ομαλή λειτουργία της επιχείρησης πρέπει όλες οι λειτουργίες να πειθαρχούν σε μια αυστηρώς καθορισμένη χρονική διάρκεια. Αυτό απαιτεί η ομαλή λειτουργία του κάθε οργανισμού. Η κάθε επαναλαμβανόμενη διαδικασία πρέπει να εκτελείται γρήγορα και ασφαλές για να μπορεί να υπάρχει συνοχή και ομαλή λειτουργία του συνόλου της επιχείρησης. Έτσι και εδώ. Η ανάκτηση και πρόσβαση σε πληροφορίες πρέπει να γίνεται αυτοματοποιημένα χωρίς χρονοτριβές και όλα τα κακά συνεπακόλουθα που μπορούν να επιφέρουν με μορφή ντόμινο σε άλλους τομείς και εργασίες. Ο παραδοσιακός τρόπος ανάκτησης των δεδομένων είναι πολύ χρονοβόρος και έτσι επιφέρει συνεχώς όλα τα κόστη τόσο σε εργασιακό χρόνο αλλά και σε ανθρώπινο δυναμικό, που στο τέλος απόβαιναν επιζήμια ως προς την επιχείρηση.

2.3 Ασφάλεια Πρόσβασης

Τα δεδομένα με τη χειρόγραφη κατακράτησή τους, φυσικά και θα φυλάγονται εντός της επιχείρησης σε φυσικό χώρο. Εδώ εγκυμονούν πολλοί κίνδυνοι. Ένας κίνδυνος είναι η μη επιθυμητή πρόσβαση σε ευαίσθητα δεδομένα της επιχείρησης που δεν πρέπει να έχει πρόσβαση κανείς εκτός αυτούς που πρέπει να έχουν το δικαίωμα. Με τη φυσική ύπαρξη των δεδομένων σε χώρο της επιχείρησης αυξάνεται και ο κίνδυνος τα δεδομένα να επιπέσουν σε λάθος άτομα, όπου αυτό μπορεί να αποβεί μοιραίο για την επιχείρηση σαν οργανισμό. Επιπλέον εδώ πολύ σημαντικό είναι ότι υπάρχει ανοικτό το ενδεχόμενο της κλοπής και δολιοφθοράς κάποιων δεδομένων. Ναι, αυτό είναι πολύ πιθανό σενάριο που πρέπει επίσης να μας προβληματίσει, για να σχεδιάσουμε το σύστημα ορθά ώστε να αποτρέπεται μία τέτοια περίπτωση.

2.4 Εξοικονόμηση Πόρων

Ένα ακόμη σοβαρό πρόβλημα είναι αυτό της εξοικονόμησης πόρων όπως χαρτιού, μελανιού και εκτυπωτών. Γενικά η χειρόγραφη καταγραφή εγγράφων απαιτεί μια πολύ μεγάλη ποσότητα χαρτικής ύλης. Αυτό αυτομάτως σημαίνει επιπλέον κόστη προς τη λειτουργία της εταιρείας που αυτό θέλουμε να αποφύγουμε όσο δύναται. Επίσης ένα ακόμη προκύπτον θέμα είναι ο χώρος και ο όγκος που απαιτούν αυτά τα δεδομένα για τη διαφύλαξή τους, που στο χρόνο τείνουν να αυξάνονται δυναμικά.

2.5 Εξοικονόμηση Ανθρώπινου Δυναμικού

Κάθε φορά που κάποιο μέλος της επιχείρησης προσπαθεί να έχει πρόσβαση σε κάποια από τα δεδομένα της εταιρείας έρχεται σε επαφή με το εξής πρόβλημα. Πρέπει να μεταβεί στο χώρο τον οποίο βρίσκονται τα δεδομένα και διαμέσου της εκάστοτε τεχνικής και μεθόδου που κάθε οργανισμός έχει αναπτύξει για να καταφέρει να ανακτήσει την συγκεκριμένη πληροφορία. Αυτό το ίδιο αποτελεί μία πολύ χρονοβόρα διαδικασία που στην κάθε εργάσιμη μέρα μπορεί να αποτελεί εφιάλτη για τον εργοδότη και τους υπαλλήλους. Ας υποθέσουμε ότι κάθε φορά που θέλει να ένας υπάλληλος να πάρει μία καρτέλα ενός συγκεκριμένου πελάτη και να τη τροποποιήσει, θα του έπαιρνε τουλάχιστον 10 λεπτά για να το κάνει αυτό. Όπως καταλαβαίνουμε μηνιαίως ή χρονιαίως αυτός ο αριθμός όταν υπολογιστεί θα έχει μία σίγουρη εκθετική αύξηση. Αυτός όλος ο χρόνος που εγώ θα τον χαρακτήριζα χαμένο χρόνο

δεν αποτελεί τίποτε άλλο στο τέλος την ημέρας ένα χρηματικό κόστος για τον εργοδότη και την εταιρεία. Ο λόγος είναι ότι όλες αυτές οι εργατοώρες μπορούσαν να διοχετευτούν στις οποιεσδήποτε άλλες δουλειές ή ανάγκες που υπάρχουν εντός της εταιρείας για την καλύτερη λειτουργικότητα και αποτελεσματικότητά της.

2.6 Περιοριστικά Όρια Προσβασιμότητας

Στη σύγχρονη κοινωνία στην οποία ζούμε, καθώς και οι ανάγκες στον σύγχρονο επαγγελματικό χώρο είναι πάρα πολλές. Οι ανάγκες πολλές φορές οδηγούν τα μέλη μιας επιχείρησης να ταξιδεύουν αρκετά τόσο εντός μιας χώρας αλλά παράλληλα και εκτός. Μπροστά στην κατάσταση, καταστροφική παρουσιάζεται μέχρι πρότινος χρησιμοποιούμενη παραδοσιακή διαδικασία ανάκτησης και διαφύλαξης. Είναι αδύνατο χωρίς τη φυσική παρουσία του οποιουδήποτε στο χώρο όπου φυλάσσονται τα δεδομένα, να μπορεί να έχει πρόσβαση. Τι γίνεται στην οποιανδήποτε δυναμική στιγμή όπου ο εργοδότης ή ο εργοδοτούμενος χρειαστεί την άμεση πρόσβαση στα δεδομένα αυτά για την οποιανδήποτε περίπτωση; Αυτό καθίσταται αδύνατο, οδηγώντας την ίδια την εταιρεία και τα στελέχη της σε μία δυσλειτουργική και δυσκινητική κατάσταση.

2.7 Δυσκολία στην Ανάκτηση, Μετατροπή και Εύρεση δεδομένων

Όπως προαναφέρθηκε ο αναχρονιστικός και πολύ χρονοβόρος τρόπος ανάκτησης των στοιχείων της εταιρείας είναι δεδομένο. Η ανάκτηση είναι πάρα πολύ δύσκολη όπως επεξηγήθηκε πιο πάνω, διότι οι μέθοδοι εύρεσης τους είναι πάρα πολύ απαιτητικοί. Ένα ακόμη σοβαρό θέμα που εμπίπτει εδώ είναι το θέμα της μετατροπής των εγγράφων αυτών. Σχεδόν πάντα όταν χρειαστεί κάποια διόρθωση ή αλλαγή σε αυτά να μην μπορεί να πραγματοποιηθεί λόγω της χειρόγραφης τους φύσης. Έτσι στις πλείστες φορές για την οποιανδήποτε αλλαγή απαιτείται εκ νέου εισαγωγή της πληροφορίας αυτής. Αυτό επίσης προσδίδει επιπλέον κόστος σε χρόνο, ανθρώπινο δυναμικό και πόρους.

Κεφάλαιο 3

Ανάλυση Προδιαγραφών

Στην ανάλυση προδιαγραφών των απαιτήσεων έχουμε ως στόχο να ανακαλύψουμε τυχόν ασάφειες του συστήματος που παραλείψαμε μέσω της ανάλυσης των λειτουργιών του συστήματος. Σε αυτό το στάδιο καθορίζονται οι προδιαγραφές του υπό μελέτη συστήματος και καταγράφονται στο πιο σημαντικό παραδοτέο υλικό της ανάλυσης το Κείμενο Προδιαγραφών των Απαιτήσεων. Επίσης αυτό το έγγραφο είναι καλό να συνοδεύεται από διαγράμματα ή και πίνακες, όπως επίσης και σενάρια χρήσης.

3.1 Σενάρια Χρήσης

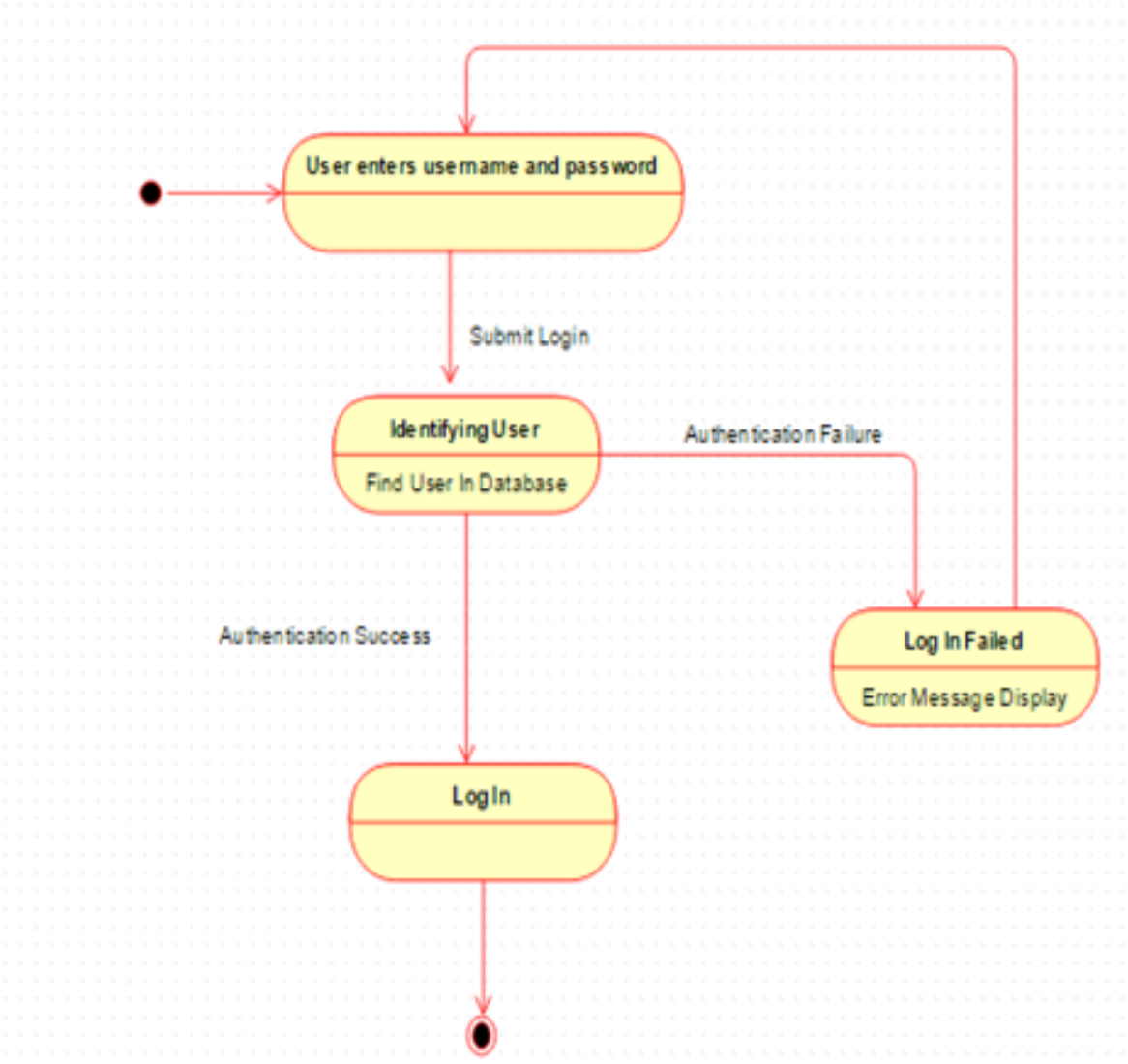
3.1.1.Λετουργία Log in

Περιγραφή Σεναρίου

Στο παρακάτω σενάριο φαίνεται διαγραμματικά πως ο χρήστης μπορεί να συνδεθεί για να εισέλθει στο σύστημα. Συγκεκριμένα ο χρήστης του συστήματος χρειάζεται να δώσει το

όνομα χρήστη (username) και το συνθηματικό (password) τα οποία είναι καταχωρημένα στην βάση δεδομένων. Στη συνέχεια γίνεται επαλήθευση των δύο αυτών γνωρισμάτων από την βάση και ο χρήστης συνδέεται στο σύστημα. Σε περίπτωση εισαγωγής λάθους στα γνωρίσματα, εμφανίζεται το ανάλογο μήνυμα και το σύστημα ζητά από τον χρήστη να επαναλάβει την διαδικασία δίνοντας ξανά το όνομα χρήστη και το συνθηματικό.

State διάγραμμα



Σχήμα 1.1

3.1.2 Λειτουργία Αναζήτησης

Περιγραφή Σεναρίου

Η διεπαφή αναζήτησης επιτρέπει στον διαχειριστή να αναζητήσει οποιονδήποτε πελάτη με κάποιες πληροφορίες.

State διάγραμμα

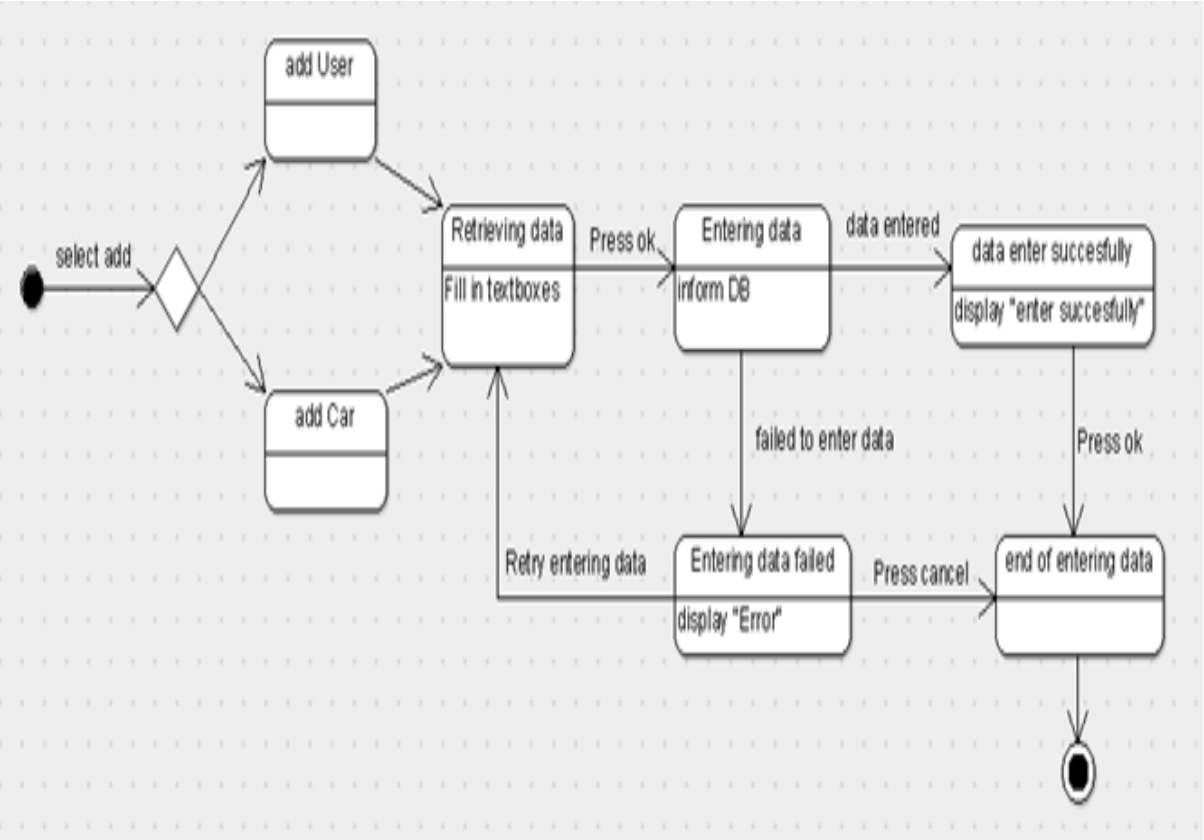
Σχήμα 1.2

3.1.3 Λειτουργία εισαγωγής πελάτη - αυτοκινήτου

Περιγραφή Σεναρίου

Στο πιο κάτω σενάριο παρουσιάζεται διαγραμματικά πως ο χρήστης μπορεί να εισάγει ένα νέο πελάτη και τα στοιχεία του αυτοκινήτου του στο σύστημα. Για να επιτευχθεί αυτό, ο χρήστης αφού κάνει log in στο σύστημα, καλείται να επιλέξει την λειτουργία Add User ή Add Car και να συμπληρώσει τα ανάλογα πεδία που του ζητούνται. Έπειτα διακρίνονται δύο περιπτώσεις. Στην πρώτη περίπτωση ο χρήστης εισάγει με επιτυχία τα στοιχεία του πελάτη και αυτά αποθηκεύονται στη ΒΔ. Στη δεύτερη περίπτωση τα στοιχεία του πελάτη δεν εισάγονται επιτυχώς και δίνεται η δυνατότητα στο χρήστη να δοκιμάσει να εισάγει ξανά τα στοιχεία.

State διάγραμμα



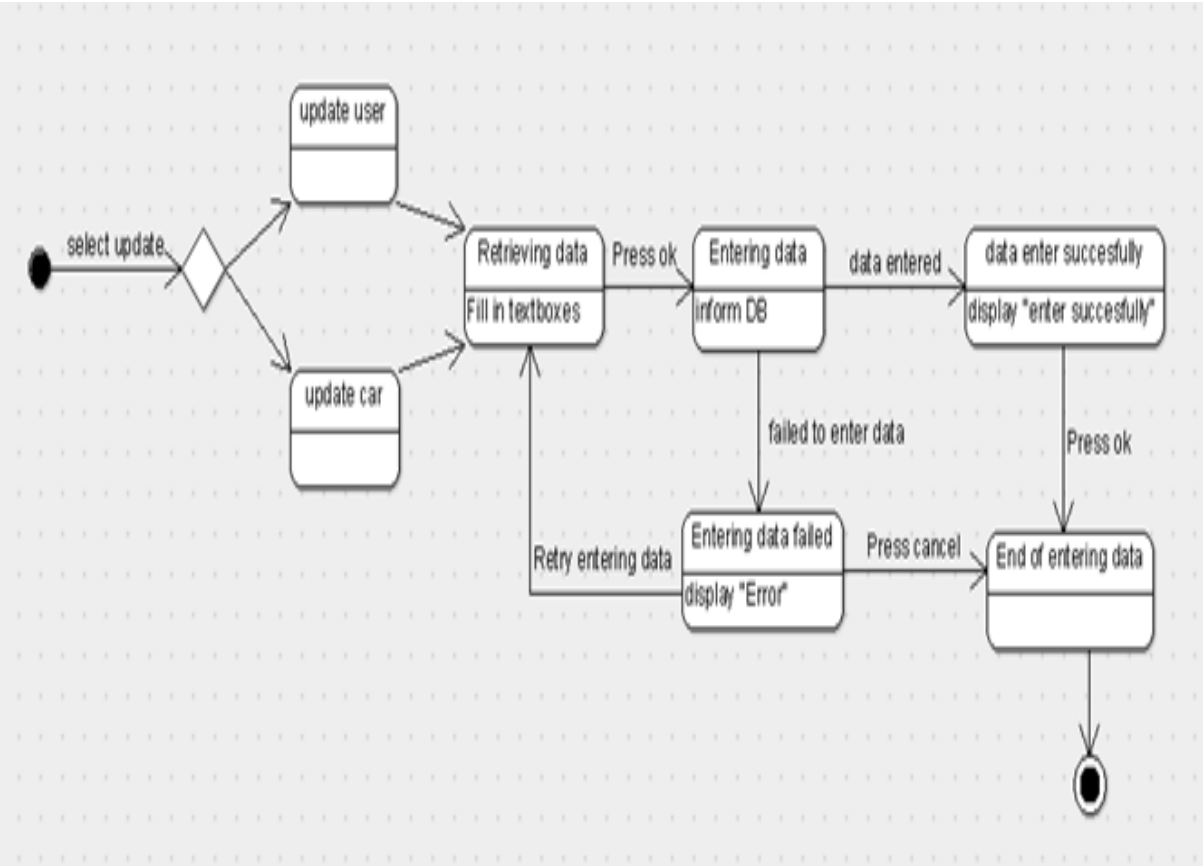
Σχήμα 3.1

3.1.4 Λειτουργία τροποποίησης πελάτη - αυτοκινήτου

Περιγραφή Σεναρίου

Στο πιο κάτω σενάριο παρουσιάζεται διαγραμματικά πως ο χρήστης μπορεί να τροποποιήσει τα στοιχεία κάποιου πελάτη ή τα στοιχεία του αυτοκινήτου του. Για να επιτευχθεί αυτό, ο χρήστης καλείται να επιλέξει τη λειτουργία Update User ή Update Car και έτσι μπορεί να τροποποιήσει κάποια στοιχεία που θέλει. Έπειτα διακρίνονται δύο περιπτώσεις. Στην πρώτη περίπτωση ο χρήστης καταφέρνει επιτυχώς την τροποποίηση των στοιχείων του πελάτη και αυτά αποθηκεύονται στη ΒΔ. Στη δεύτερη περίπτωση δεν γίνεται επιτυχώς η τροποποίηση των στοιχείων και έτσι δίνεται η δυνατότητα στο χρήστη να δοκιμάσει ξανά.

State διάγραμμα



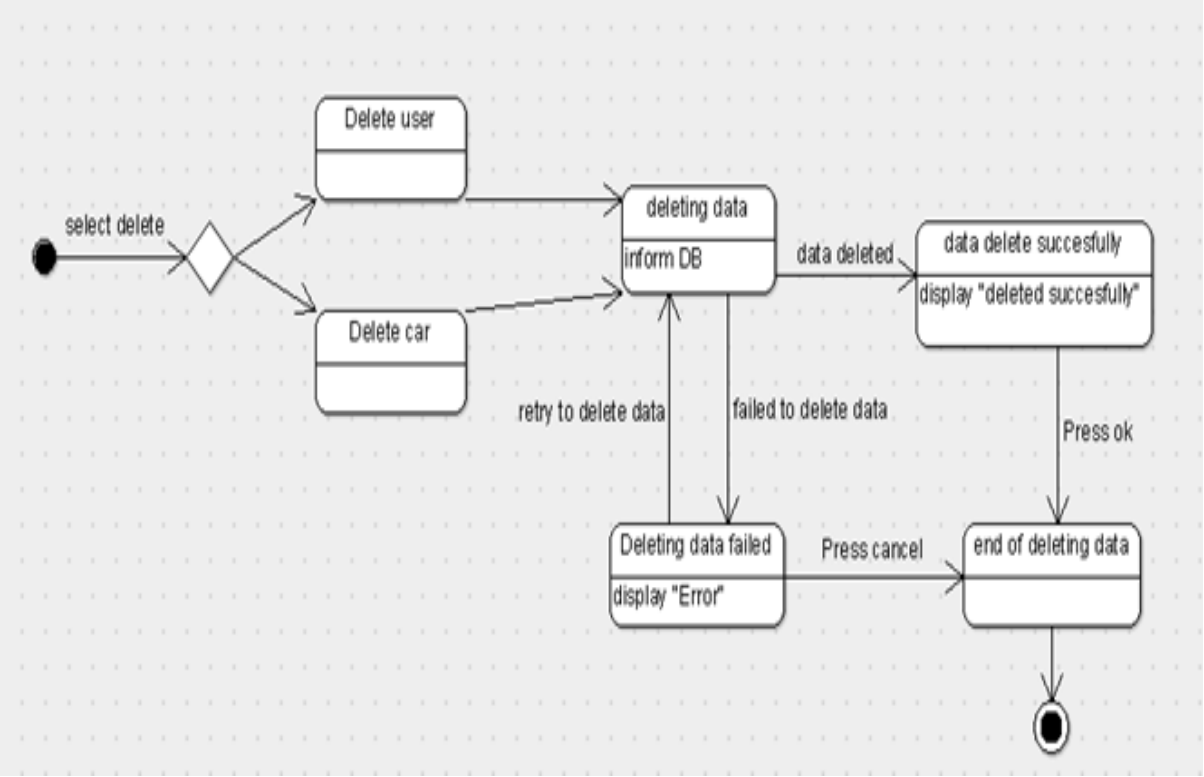
Σχήμα 3.2

3.1.5 Λειτουργία διαγραφής πελάτη-αυτοκινήτου

Περιγραφή Σεναρίου

Στο πιο κάτω σενάριο παρουσιάζεται διαγραμματικά πως ο χρήστης μπορεί να διαγράψει ένα πελάτη και τα στοιχεία του. Για να επιτευχθεί αυτό, ο χρήστης καλείται να επιλέξει τη λειτουργία Delete User ή Delete Car και έτσι μπορεί να διαγράψει κάποιο πελάτη ή αυτοκίνητο και τα στοιχεία του. Έπειτα διακρίνονται δύο περιπτώσεις. Στην πρώτη περίπτωση ο χρήστης καταφέρνει επιτυχώς την διαγραφή του πελάτη και έπειτα ενημερώνεται η ΒΔ. Στη δεύτερη περίπτωση η διαγραφή δεν γίνεται με επιτυχία και έτσι δίνεται η δυνατότητα στο χρήστη να δοκιμάσει ξανά.

State διάγραμμα



Σχήμα 3.3

3.2 Διαγράμματα Περιπτώσεων Χρήσης για Διαχείριση Δεδομένων

Παρατίθενται όλες οι επί μέρους περιπτώσεις χρήσης παρουσιάζοντας τις μεταξύ τους διασυνδέσεις, εξαρτήσεις και επικοινωνίες.

Όπως βλέπουμε στα πιο κάτω διαγράμματα, στο σύστημα υπάρχουν δύο τύποι "actor". Ο πρώτος, είναι ο κάθε χρήστης ο οποίος αλληλεπιδρά με το σύστημα (μέλος), ενώ δεύτερος τύπος είναι η ΒΔ (Βάση Δεδομένων).

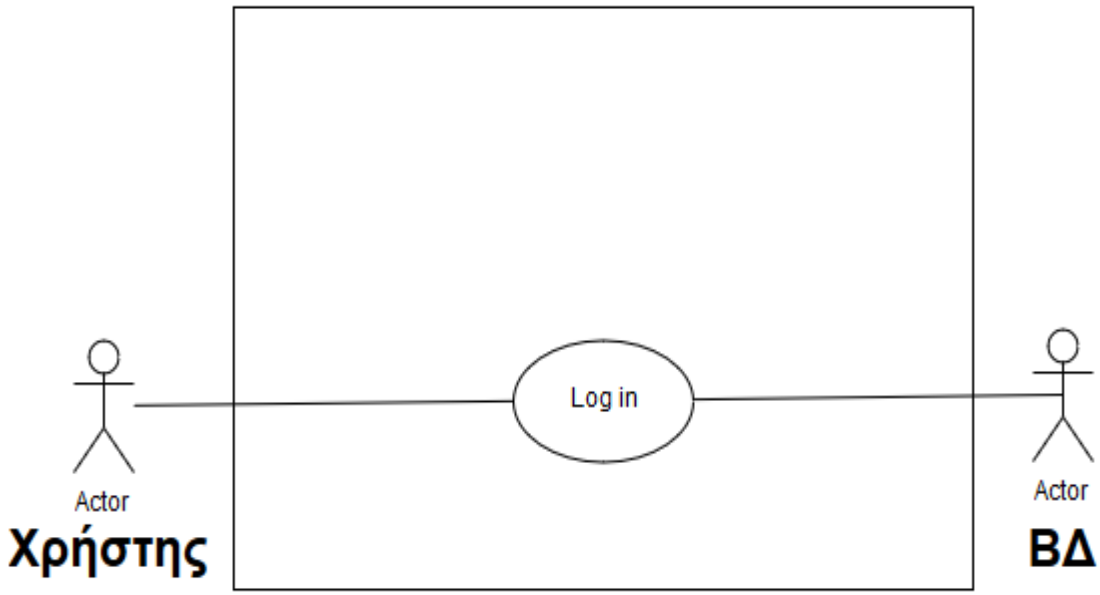
Περιγραφή του actor «Χρήστης»:

Η αναφορά στον actor «Χρήστης» αντικατοπτρίζει τον εκάστοτε άνθρωπο που χρησιμοποιεί τη εφαρμογή. Η οποιαδήποτε πράξη του χρήστη, ενεργοποιεί το ανάλογο use case, το οποίο παρουσιάζει μια συγκεκριμένη λειτουργία του συστήματος.

Περιγραφή του actor ΒΔ (Βάση Δεδομένων) :

Η ΒΔ αποτελεί τον πιο σημαντικό παράγοντα για τη λειτουργικότητα της εφαρμογής. Θα είναι υπεύθυνη για την διαφύλαξη των προσωπικών στοιχείων και των στοιχείων επαλήθευσης του κάθε μέλους, ως επίσης τα διάφορα γεγονότα βάσει είδους κτλ. Κατά τη χρήση του συστήματος, η ΒΔ θα επιστρέφει όλα τα στοιχεία και πληροφορίες πίσω στην οθόνη.

3.2.1 Λειτουργία Log In



Σχήμα 3.4

Περιγραφή:

Με αυτή τη λειτουργία ο χρήστης είναι σε θέση να εισέλθει στο σύστημα δίνοντας το username και το password του. Αναλόγως της ορθότητας των στοιχείων που δίνει, η εφαρμογή επιτρέπει ή αποτρέπει την πρόσβαση.

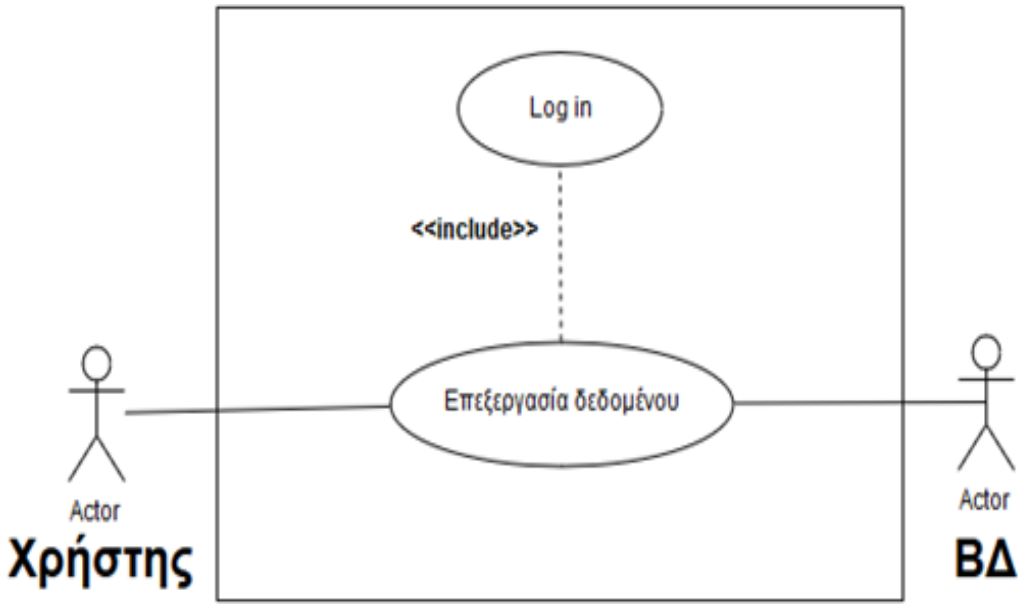
Actors:

Σε αυτή τη περίπτωση συμμετέχουν ο χρήστης και η ΒΔ προκειμένου να κάνει Log In ο χρήστης.

Προϋπόθεση:

Προϋποθέτεται ότι ο χρήστης είναι ήδη εγγεγραμμένος στο σύστημα.

3.2.2 Λειτουργία επεξεργασίας δεδομένων



Σχήμα 3.5

Περιγραφή:

Αυτή η περίπτωση χρήσης δίνει στο χρήστη τη δυνατότητα να επεξεργαστεί ένα δεδομένο από τη βάση.

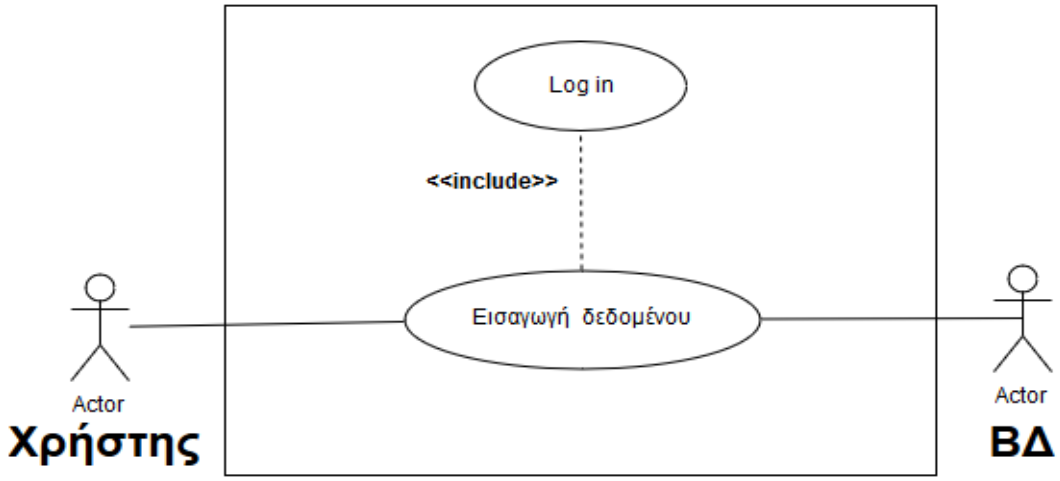
Actors:

Στη συγκεκριμένη περίπτωση χρήσης λαμβάνουν μέρος ο χρήστης και η ΒΔ για να μπορεί η τελευταία να ενημερώνεται για την κάθε επεξεργασία

Προϋπόθεση:

Προϋποθέτεται ότι ο χρήστης είναι ήδη εγγεγραμμένος στο σύστημα.

3.2.3 Λειτουργία εισαγωγής δεδομένων



Σχήμα 3.6

Περιγραφή:

Αυτή η περίπτωση χρήσης δίνει στο χρήστη τη δυνατότητα να δημιουργεί ένα δεδομένο και να το εισάγει στη βάση. Με όλα τα απαραίτητα στοιχεία, αριθμός ιδιόκτητη, τηλέφωνο, ημερομηνία, services, κτλ.

Actors:

Στη συγκεκριμένη περίπτωση χρήσης λαμβάνουν μέρος ο χρήστης και η ΒΔ για να μπορεί η τελευταία να ενημερώνεται για την κάθε εισαγωγή.

Προϋπόθεση:

Προϋποθέτεται ότι ο χρήστης είναι ήδη εγγεγραμμένος στο σύστημα.

Κεφάλαιο 4

Σχεδιασμός Συστήματος

4.2 Εννοιολογική Σχεδίαση Βάσης (Conceptual Database Design)

Στην εννοιολογική σχεδίαση της βάσης ο Database Designer ετοιμάζει ένα ER (Entity Relationship) διάγραμμα το οποίο μπορεί να γίνει κατανοητό στο πελάτη για επικύρωση. Το διάγραμμα αυτό πρέπει να ορθό, πλήρες και αποδοτικό για να είναι εύκολη η μετατροπή στο επόμενο στάδιο.

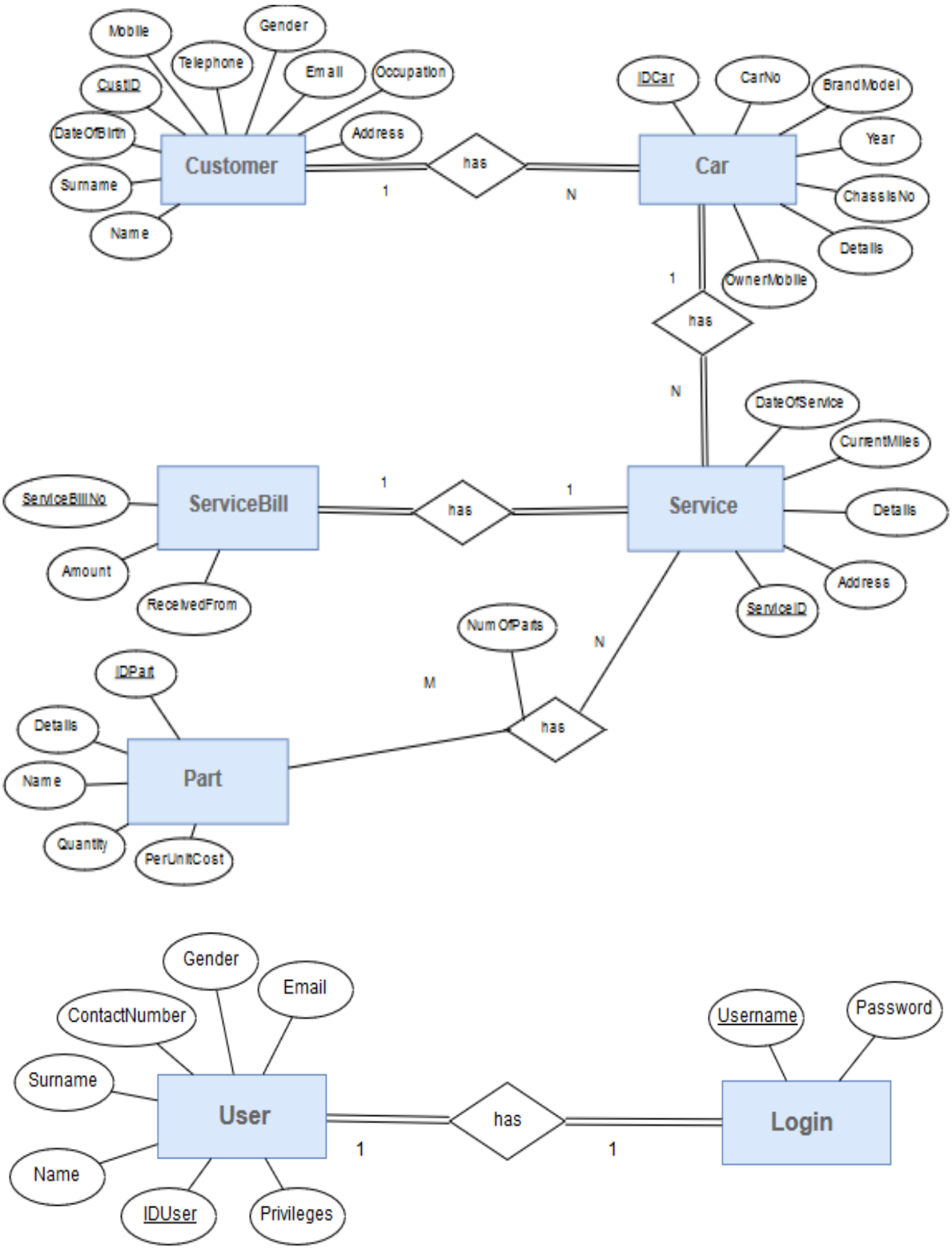
4.2.1 E-R Model (Entity-Relationship Model)

Το μοντέλο **Οντοτήτων Συσχετίσεων** είναι ένα εννοιολογικό μοντέλο που χρησιμοποιείται για να καταγράψει τις απαιτήσεις των χρηστών ενός νέου πληροφοριακού συστήματος με γραφικό τρόπο.

Βασικές Έννοιες:

- Οντότητες
- Συσχετίσεις
- Γνωρίσματα
- Περιορισμοί (κλειδιά, συμμετοχές, πληθικότητα, κλπ)

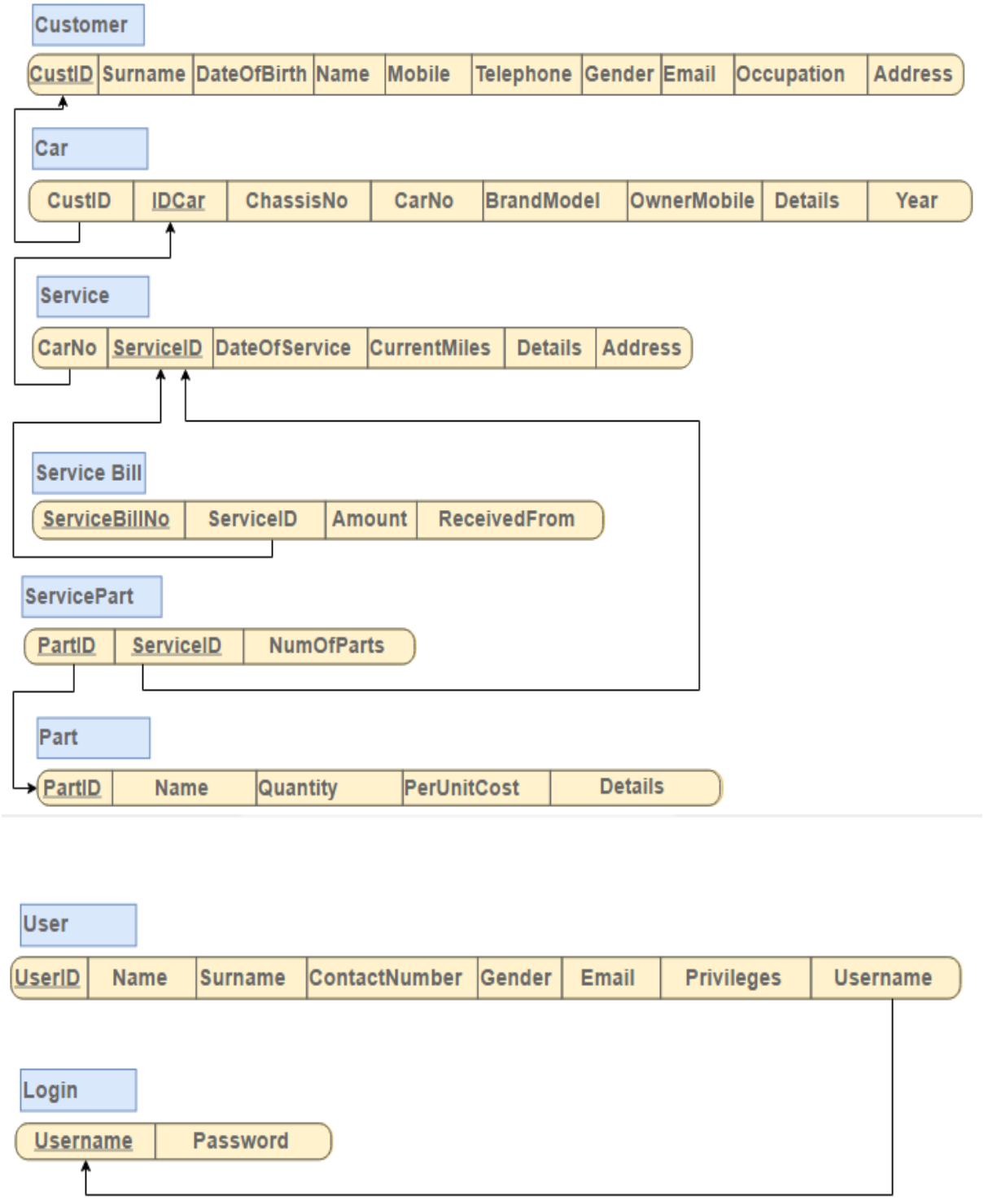
E-R Model:



Σχήμα 4.1

4.3 . Λογική Σχεδίαση Βάσης (Logical Database Design)

Μετατροπή του ER διαγράμματος σε ένα Σχεσιακό Σχήμα Βάσης:



Σχήμα 4.2

Κεφάλαιο 5

Υλοποίηση

5.1 Τεχνολογίες που χρησιμοποιήθηκαν

Για τις ανάγκες δημιουργίας του προγράμματος και της λογικής που έχω περιγράψει , αποφάσισα να χρησιμοποιήσω τις εξής τεχνολογίες:

5.1.1 Γλώσσα Προγραμματισμού Java

Για την υλοποίηση των προγραμματιστικών λειτουργιών του συστήματός μας χρησιμοποιήθηκε η γλώσσα προγραμματισμού Java. Κατέληξα σε αυτή την επιλογή γιατί αποτελεί μία ευρέως διαδεδομένη γλώσσα . Επίσης είναι μία open source γλώσσα που δίνει την δυνατότητα στον καθένα να έχει μια μεγάλη πρόσβαση σε μια πληθώρα πληροφοριών και δεδομένων σχετικά με αυτή δωρεάν. Επίσης ένα πολύ σημαντικό κριτήριο που μας οδήγησε στην επιλογή της είναι η multi-platform independet ιδιότητά της. Δηλαδή, το κάθε

λογισμικό που θα αναπτυχθεί με τη γλώσσα αυτή μπορεί να εκτελεστεί και να τρέξει σε κάθε μηχανή ανεξαρτήτως λειτουργικού συστήματος που υπάρχει στην εκάστοτε μηχανή.

5.1.2 MySQL γλώσσα

Ασφαλώς για την ανάπτυξη ενός συστήματος όπως αυτού που ανήκει στην γενική οικογένεια των CMS (Content Management Systems) είναι απαραίτητη η ύπαρξη μιας βάσης δεδομένων στην οποία θα αποθηκεύονται τα δεδομένα σε μια δομημένη μορφή που αναδύθηκε κατά το σχεδιασμό της βάσης δεδομένων. Η MySQL αποτελεί επίσης μια ανοικτού κώδικα γλώσσα για βάσεις δεδομένων και είναι μια από τις πλέον πιο γνωστές ανά το παγκόσμιο. Λόγω της ασφάλειας που προσφέρει, της σταθερότητας καθώς και της πλήρης υποστηριζόμενης συνεργασίας και συνδυασμού με την γλώσσα προγραμματισμού Java, κατέληξα ότι αποτελεί μια από τις καλύτερες επιλογές που έχω.

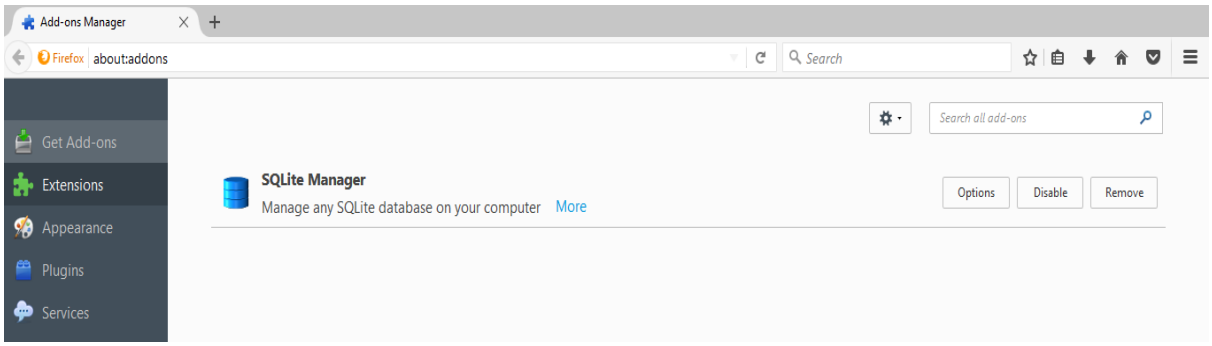
Κάθε φορά που κάποια λειτουργία εντός του συστήματος που απαιτούσε αλληλεπίδραση με την βάση δεδομένων επισύναπτα στον κώδικα του προγράμματός μου το ανάλογο query που στην συνέχεια θα εκτελείτω από το πρόγραμμά μου για το ανάλογο αποτέλεσμα. Η αλληλεπίδραση του προγράμματος και της βάσης δεδομένων πραγματοποιείται κάθε φορά με τη χρήση του JDBC (Java Database Connectivity) που είναι μια έτοιμη κλάση για επικοινωνία μεταξύ του προγράμματος και της βάσης.

5.1.3 SQLite Manager (Add-on for Firefox)

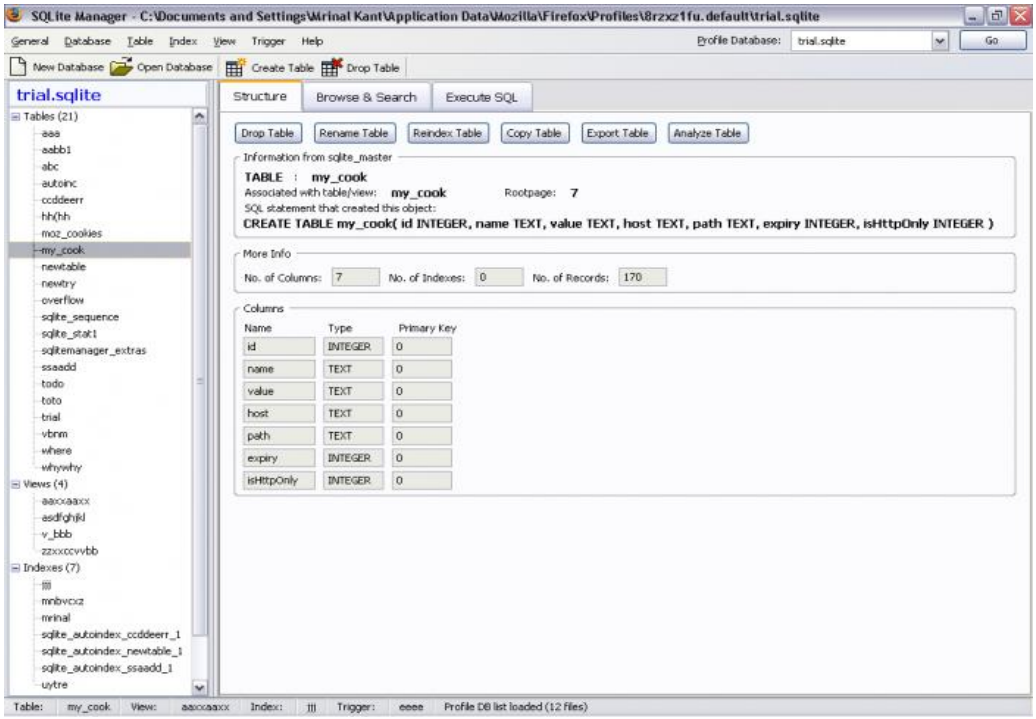
Το SQLite Manager είναι ένα Add-on το οποίο μπορεί να ενσωματώσει κάποιος μόνο στο Mozilla Firefox για να διαχειριστεί οποιαδήποτε βάση δεδομένων SQLite στον υπολογιστή του. Ειδικότερα:

- Είναι ένα heirarchical tree που δείχνει αντικείμενα βάσης δεδομένων.
- Παρέχει χρήσιμα παράθυρα διαλόγου για τη διαχείριση πινάκων, ευρετηρίων, προβολών και ενεργοποιήσεων.
- Μπορούμε να περιηγηθούμε και να κάνουμε αναζήτηση στους πίνακες, καθώς και να προσθέσουμε, να επεξεργαστούμε, να διαγράψουμε και να αντιγράψουμε τις εγγραφές.
- Εύκολη εκτέλεση οποιουδήποτε ερωτήματος SQL.

- Οι εγγραφές μπορούν επίσης να αναζητηθούν.
- Έχει ένα αναπτυσσόμενο μενού που βοηθά με τη σύνταξη sql, καθιστώντας έτσι ευκολότερη την εγγραφή sql.
- Παρέχει εύκολη πρόσβαση σε κοινές λειτουργίες μέσω μενού, γραμμών εργαλείων, κουμπιών και του μενού περιβάλλοντος.
- Εξαγωγή πινάκων / προβολών / βάσης δεδομένων σε μορφή csv / xml / sql. Εισαγωγή από csv / xml / sql (UTF-8 και UTF-16).
- Δυνατότητα εκτέλεσης πολλαπλών καταστάσεων sql στο Execute.
- Μπορούμε να αποθηκεύσετε τα ερωτήματα.
- Παρέχει υποστήριξη για το ADS στα Windows.



Σχήμα 5.1.1



Σχήμα 5.1.2

5.1.4 NetBeans IDE

Το NetBeans είναι ένα επιτυχημένο open source ερευνητικό έργο με μεγάλο αριθμό χρηστών, μια αναπτυσσόμενη κοινωνία, κοντά στους 100 και πλέον συνεργάτες παγκοσμίως. Η Sun Microsystems ίδρυσε τη NetBeans τον Ιούνιο του 2000 και συνεχίζει να είναι ο κύριος ανάδοχος.

Σήμερα υπάρχουν δύο ερευνητικά έργα, το NetBeans IDE και το NetBeans Platform.

Το NetBeans IDE είναι ένα περιβαλλοντικό ανάπτυγμα IDE - ένα εργαλείο που δίνει την δυνατότητα στους προγραμματιστές να γράψουν, να κάνουν compile, debug και να αναπτύξουν προγράμματα. Είναι γραμμένο σε Java - αλλά μπορεί να υποστηρίξει κι άλλες γλώσσες προγραμματισμού. Υπάρχει επίσης ένας μεγάλος αριθμός υπομονάδων (modules) που βοηθάνε στην επέκταση της λειτουργικότητας του NetBeans IDE. Το NetBeans IDE είναι ένα ελεύθερο προϊόν δίχως περιορισμούς στον τρόπο χρησιμοποίησής του.

Διαθέσιμο επίσης είναι το NetBeans Platform, που χρησιμοποιείται σαν βάση λογισμικού για τη δημιουργία μεγάλων επιτραπέζιων (desktop) εφαρμογών. Οι ISV συνεργάτες διαθέτουν προσθήκες, επιπρόσθετα προγράμματα (plug-ins) που εύκολα συνενώνονται στο Platform και μπορούν επίσης να χρησιμοποιηθούν για την ανάπτυξη άλλων εργαλίων και λύσεων.

Και τα δύο τα προϊόντα είναι open source και ελεύθερα για εμπορική χρήση ή μη.

5.1.5 Java Graphics

Η κλάση Graphics είναι η αφηρημένη βασική κλάση για όλα τα πλαίσια γραφικών που επιτρέπει σε μία εφαρμογή να σχεδιάσουμε πάνω σε στοιχεία που πραγματοποιούνται σε διάφορες συσκευές.

Για το σχεδιασμό της γραφικής διαπροσωπείας του προγράμματος χρησιμοποιήθηκε η τεχνολογία Java Swing Graphics της ομώνυμης γλώσσας προγραμματισμού. Η τελευταία συνδυασμένη με το εργαλείο NetBeans IDE δίνει την δυνατότητα του σχεδιασμού αυτού.

<u>Συστατικά εισαγωγής δεδομένων</u>

- Διάλογοι (*javax.swing.JDialog*) και παράθυρα (*javax.swing.JWindow* και *javax.swing.JFrame*).
- Ετικέτες (*javax.swing.JLabel* και Κουμπιά (*javax.swing.JButton*)

- Λίστες επιλογών (πολλαπλών ή μη) (*javax.swing.JList* και *javax.swing.JComboBox*)

Συστατικά εμφάνισης δεδομένων

- Μπάρες προόδου (*javax.swing.JProgressBar*)
- Πίνακες (*javax.swing.JTable*)
- Ιεραρχικές δομές δένδρων (*javax.swing.JTree*)
- Αόρατα συστατικά (διαχειριστές διάταξης)

Προγραμματιστική διεπαφή συστατικών

Παράθυρο (*javax.swing.JFrame*, Κλάση)

- Κατασκευαστής: *JFrame(String title)*
- Ορισμός μεγέθους παραθύρου: *setBounds(int x, int y, int width, int height)*
- Καθορισμός τίτλου (εκτός του κατασκευαστή): *setTitle(String title)*
- Καθορισμός δυναμικού μεγέθους (ή οχι): *setResizable(boolean resizable)*
- Καθορισμός διαχειριστή διάταξης: *setLayout(LayoutManager lm)*
- Προσθήκη συστατικού:
- add(Component c)
- Εμφάνιση παραθύρου: *setVisible(boolean state)*

Προγραμματιστική διεπαφή συμβάντων

Κουμπιά (*javax.swing.JButton*)

- Κατασκευαστής: *JButton(String title)*
- Εναλλακτικός Κατασκευαστής: *JButton(String title, Icon icon)*
- Προσθήκη Γεγονότος Επιλογής (click event): *addActionListernet(ActionListener l)*

Διαχειριστές διάταξης

Οι διαχειριστές διάταξης (layout managers) είναι αόρατα συστατικά που καθορίζουν δυναμικά την τοποθέτηση των ορατών συστατικών (πχ κουμπιά, πίνακες) στην εφαρμογή. Οι κύριοι διαχειριστές διάταξης είναι:

1. FlowLayout (*java.awt.FlowLayout*)
    - Κατασκευαστής: *FlowLayout(int align)* (FlowLayout.LEFT, RIGHT, CENTER, LEADING, TRAILING)
    - Προσθήκη συστατικού: *addLayoutComponent(String name, Component comp)*
2. BorderLayout (*java.awt.BorderLayout*)
    - Κατασκευαστής: *BorderLayout()*
    - Προσθήκη συστατικού: *addLayoutComponent(Component comp, Object constraints)*
3. GridLayout (*java.awt.GridLayout*)
    - Κατασκευαστής: *GridLayout(int rows, int cols)*
    - Προσθήκη συστατικού: *addLayoutComponent(String name, Component comp)*

Προγραμματισμός με συμβάντα

Η διαδραστικότητα στις εφαρμογές καθορίζεται από προκαθορισμένα συμβάντα. Η νοοτροπία τους είναι πολύ απλή:

1. Επιλέγεται ένα γεγονός στο οποίο θέλουμε να επέμβουμε (πάτημα ενός πλήκτρου κτλ.)
2. Υλοποιώντας την κατάλληλη διεπαφή, γράφουμε το κομμάτι κώδικα που θέλουμε να εκτελεστεί για το συγκεκριμένο γεγονός
3. Συσχετίζουμε το κομμάτι αυτό κώδικα με τη λίστα γεγονότων του συστατικού

Τα γεγονότα είναι προκαθορισμένα και αφορούν είσοδο από το πληκτρολόγιο ή το ποντίκι, μετακινήσεις ή διαφοροποίηση του μεγέθους παραθύρων κτλ. Συνήθως οι κλάσεις που τα υλοποιούν βρίσκονται στο πακέτο *java.awt.event.**.

Σχεδίαση σχημάτων

- Όλες οι συναρτήσεις που αφορούν τη σχεδίαση σχημάτων, είναι υλοποιημένες στην κλάση *java.awt.Graphics*.
- Υπάρχει δυνατότητα για τη σχεδίαση των βασικών σχημάτων, όπως γραμμές, κύκλους, τετράγωνα κτλ.
- Όλες αυτές οι συναρτήσεις δουλεύουν σε συνδυασμό με τις συναρτήσεις για τον καθορισμό των χρωμάτων, όπως ακριβώς και στη σχεδίαση γραμματοσειρών.

Αρχεία εικόνων

Επίσης οι βιβλιοθήκες που παρέχονται μαζί με το περιβάλλον ανάπτυξης της Java, υποστηρίζουν την επεξεργασία εικόνων μέσω των πακέτων *java.awt.image.** και *javax.imageio.**.

5.1.6 MySQL Workbench

Το MySQL Workbench είναι ένα ενιαίο οπτικό εργαλείο για τους αρχιτέκτονες της βάσης δεδομένων, τους προγραμματιστές και τους διαχειριστές τη βάσης δεδομένων. Το εργαλείο MySQL Workbench παρέχει τη μοντελοποίηση των δεδομένων, την ανάπτυξη SQL, και ολοκληρωμένα εργαλεία διαχείρισης για τη διαμόρφωση του διακομιστή, διαχείριση χρηστών, δημιουργία αντιγράφων ασφαλείας, και πολλά άλλα. Το εργαλείο αυτό είναι διαθέσιμη για τα Windows, Linux και Mac OS X. Επίσης το MySQL Workbench είναι ένα αναπτυγμένο εργαλείο με user interface για την διευκόλυνση του χειρισμού, δημιουργίας, επεξεργασίας και συντήρησης των βάσεων δεδομένων που υπάρχουν σε ένα διακομιστή.
Το εργαλείο αυτό σου δίνει την δυνατότητα να δημιουργήσεις τους πίνακες που θα χρησιμοποιηθούν για τη βάση και ακολούθως, μετά την δημιουργία των πινάκων να ελέγξεις τα mysql queries πριν να τα ενσωματώσεις στον υπόλοιπο java κώδικά. Σχεδιάζοντας το ER μοντέλο της βάσης και έπειτα με forward engineer δημιουργούνται οι πίνακές μας.

5.1.7 JDBC Connectivity

Το JDBC είναι η συντομογραφία του όρου Java Database Connectivity (Συνδετικότητα Βάσης Δεδομένων JAVA). Το JDBC είναι μία διεπαφή προγραμματισμού εφαρμογών (API) για την γλώσσα προγραμματισμού Java. Είναι ένα Java API που μπορούν να έχουμε πρόσβαση σε κάθε είδους πίνακα δεδομένων, ιδίως των δεδομένων που αποθηκεύονται σε

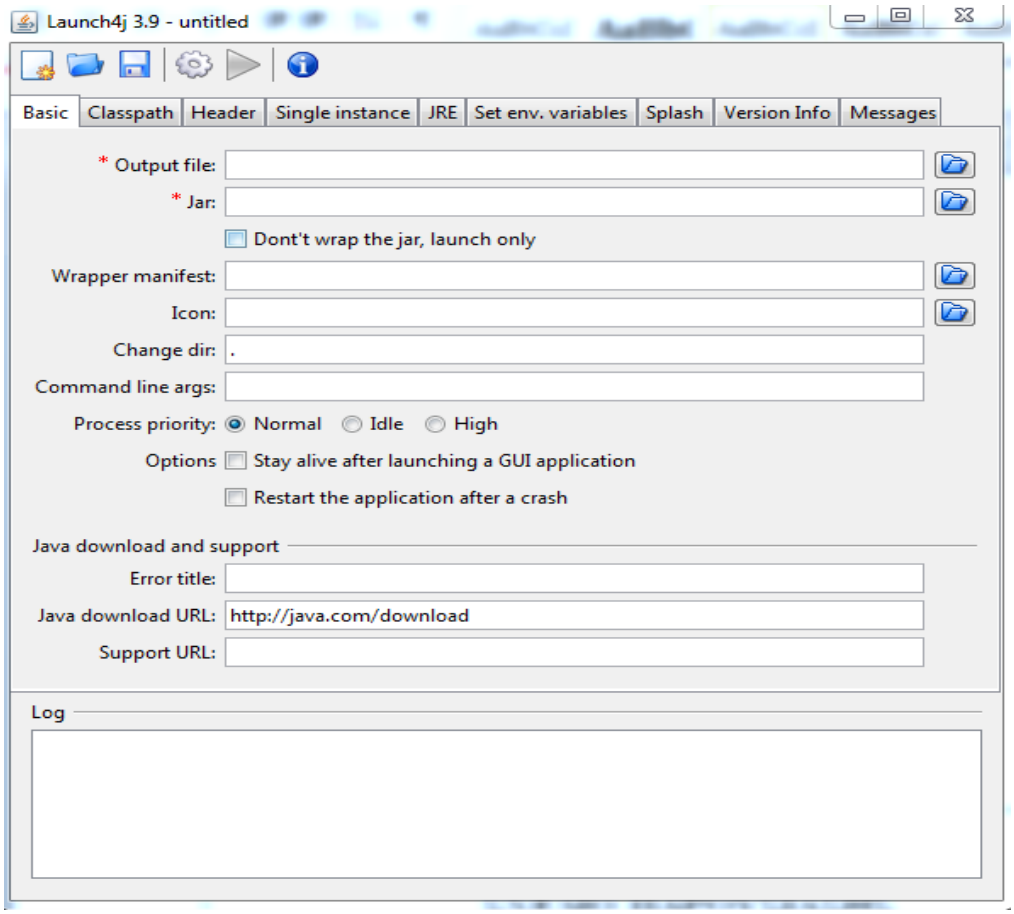
μια σχεσιακή βάση δεδομένων. Το JDBC λειτουργεί σε μια ποικιλία από πλατφόρμες, όπως τα Windows, Mac OS, και τις διάφορες εκδόσεις του UNIX.

Εχει σχεδιαστεί για Java προγραμματιστές που θα ήθελαν να κατανοήσουν το JDBC framework λεπτομερώς μαζί με την αρχιτεκτονική του και την πραγματική χρήση.

Αυτοί που το χρησιμοποιούν πρέπει να έχουν μια καλή κατανόηση της γλώσσας προγραμματισμού Java. Αν θα ασχοληθούν με RDBMS, θα πρέπει πριν να εκτεθούν σε SQL, να γνωρίζουν τις έννοιες των βάσεων δεδομένων.

5.1.8 Launch4j

Το Launch4j είναι ένα εργαλείο για τη δημιουργία εκτελέσιμου αρχείου .exe παίρνοντας σαν δεδομένο εισόδου το αρχείο .jar της εφαρμογής μας το οποίο είναι γραμμένο σε γλώσσα προγραμματισμού Java. Επίσης με την συγκεκριμένη πλατφόρμα μπορούμε να ρυθμίσουμε το εκτελέσιμο για μια συγκεκριμένη έκδοση JRE, όπως επίσης να βάλουμε εικονίδιο στην εφαρμογή μας.



Σχήμα 5.2

5.2 Στιγμιότυπα οθόνης του τελικού συστήματος

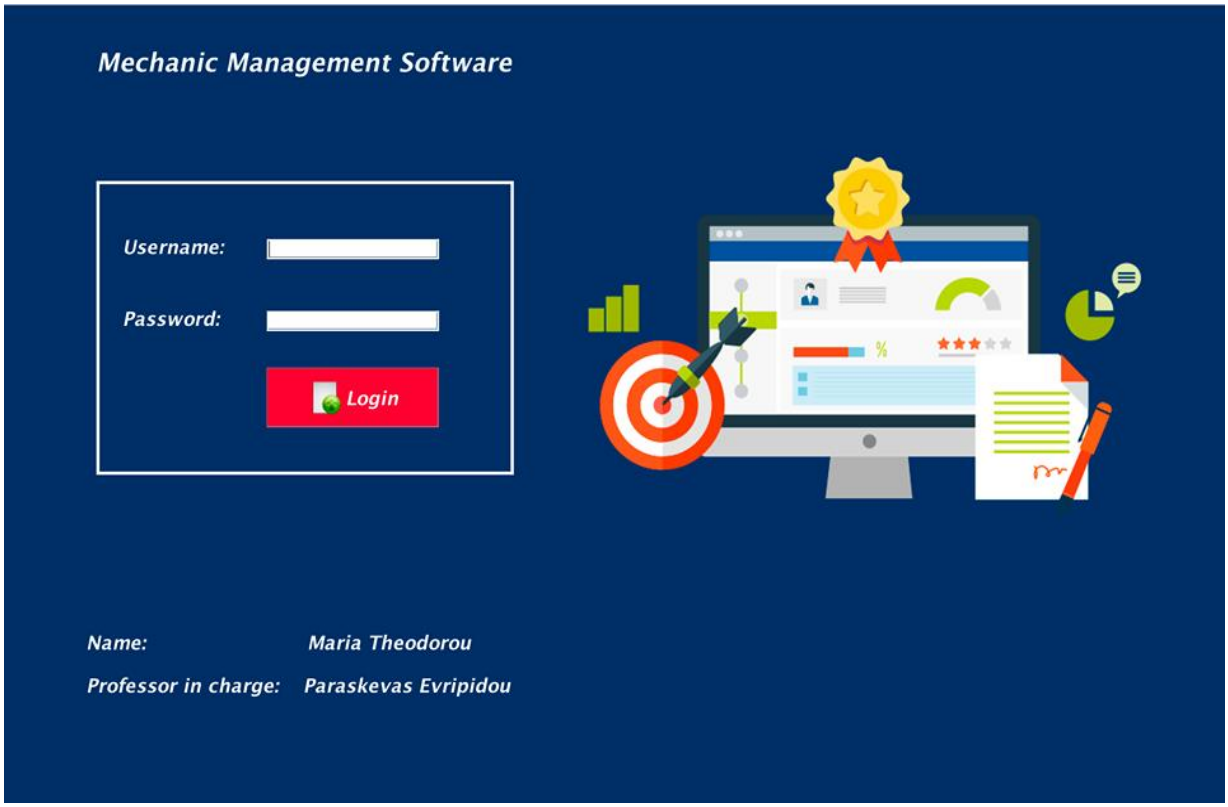
*Τα όποια δεδομένα που φαίνονται πιο κάτω στα στιγμιότυπα εισάχθηκαν τυχαία για σκοπούς παρουσίασης των λειτουργιών του συστήματος και ευκολότερης κατανόησής τους.

Εικονίδιο του εκτελέσιμου της εφαρμογής:

Σχήμα 5.3

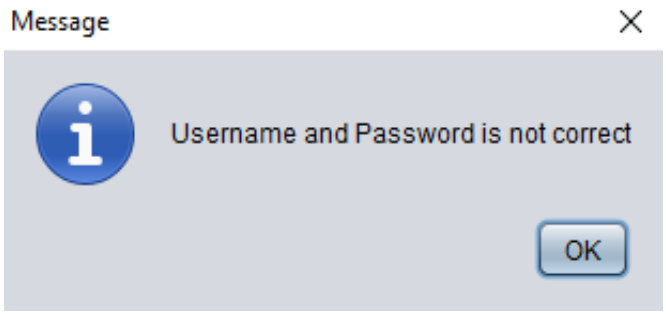
Πρόσβαση στο Σύστημα

Πιο κάτω φαίνεται το στιγμιότυπο της διεπαφής στην οποία ο χρήστης συμπληρώνει τα αντίστοιχα πεδία του username και του συνθηματικού του, όπου ταυτοποιούνται με τα δεδομένα στη βάση για να μπορέσει ο χρήστης να έχει πρόσβαση στο σύστημα.



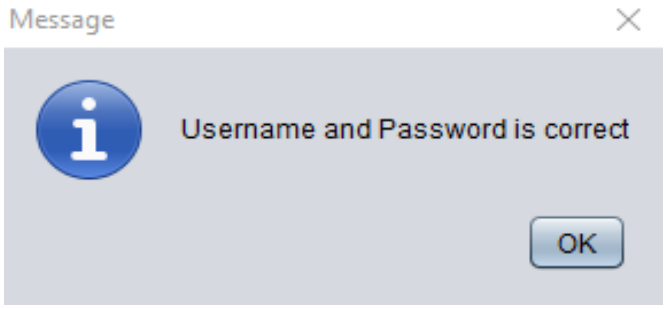
Σχήμα 5.4

Αν ο χρήστης εισάγει λάθος δεδομένα στο Username ή στο Password εμφανίζεται το πιο κάτω μήνυμα στην οθόνη το οποίο ενημερώνει το χρήστη ότι τα στοιχεία που έβαλε είναι λανθασμένα.



Σχήμα 5.5

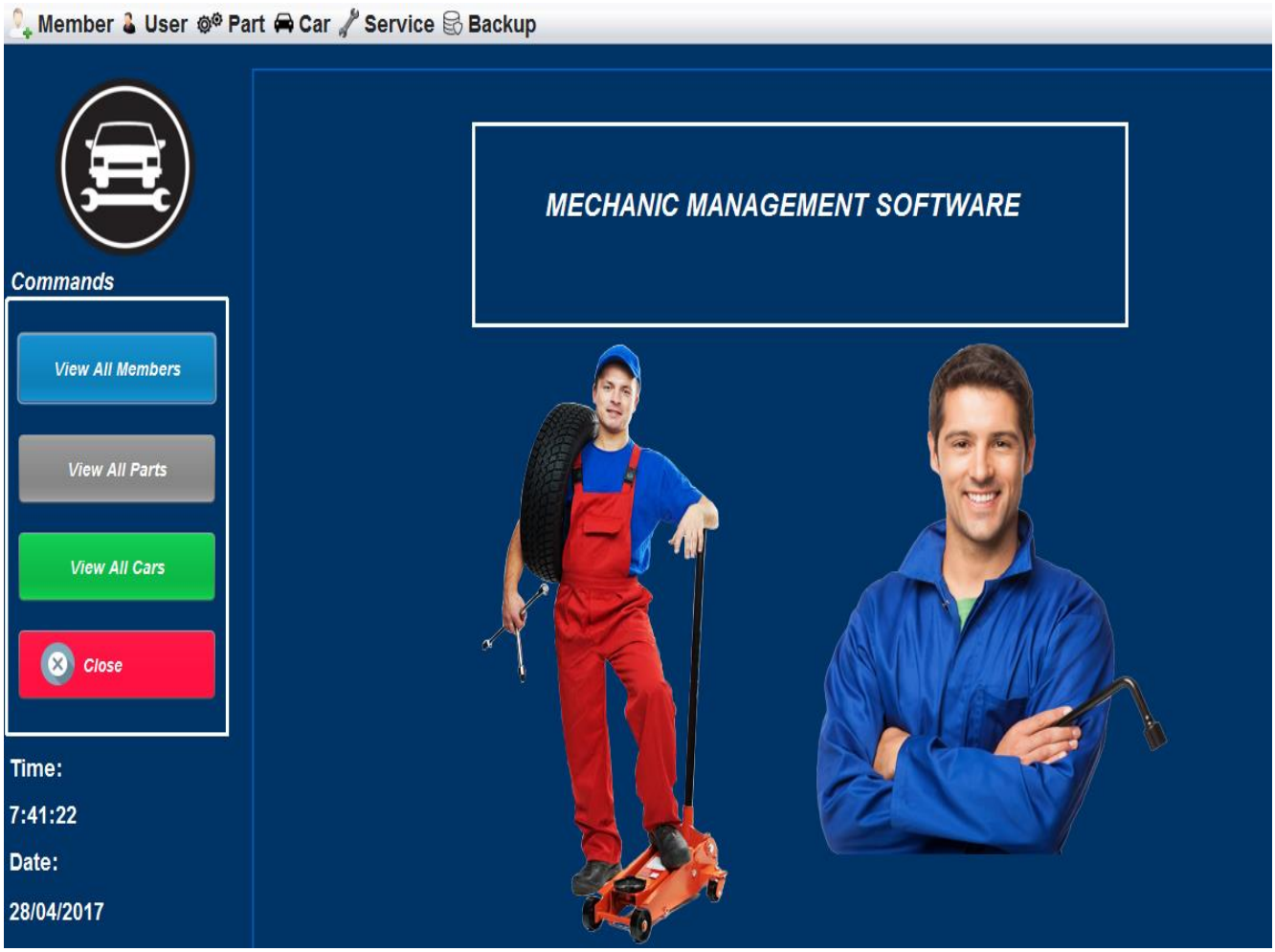
Σε αντίθετη περίπτωση που τα στοιχεία είναι σωστά εμφανίζεται το ακόλουθο μήνυμα το οποίο ενημερώνει το χρήστη ότι τα στοιχεία που έβαλε είναι ορθά και στη συνέχεια ανοίγει η διεπαφή με το αρχικό μενού του συστήματος.



Σχήμα 5.6

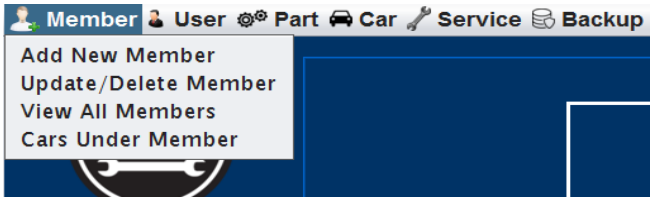
Αρχικό / Κύριο Μενού Συστήματος

Πιο κάτω φαίνεται το στιγμιότυπο οθόνης της διεπαφής για το αρχικό μενού του συστήματός μας όπου ο χρήστης μπορεί να μεταβεί σε οποιανδήποτε λειτουργία υποστηρίζει το σύστημα, είτε από το jpanel στα αριστερά της οθόνης που περιέχει κάποιες λειτουργίες με το πάτημα των κουμπιών, είτε από την μπάρα στο πάνω μέρος της οθόνης.

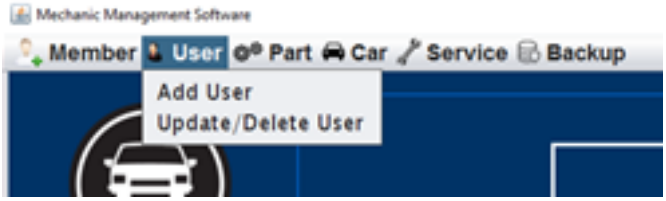


Σχήμα 5.7

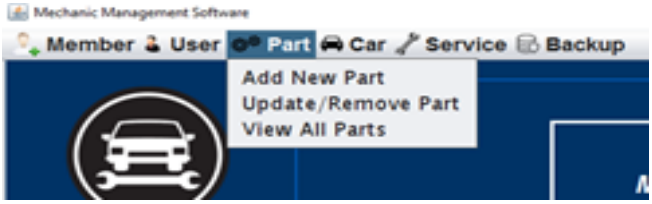
Πατώντας το καθένα από τα κουμπιά που βρίσκονται στη μπάρα μενού φτάνουμε στις επιλογές που φαίνονται στα σχήματα πιο κάτω:



Σχήμα 5.8

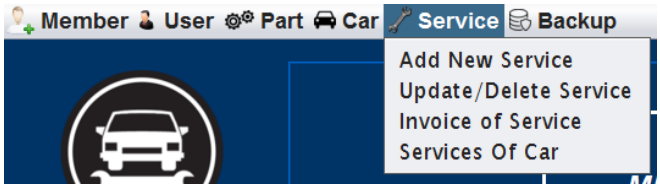


Σχήμα 5.9



Σχήμα 5.10

Σχήμα 5.11

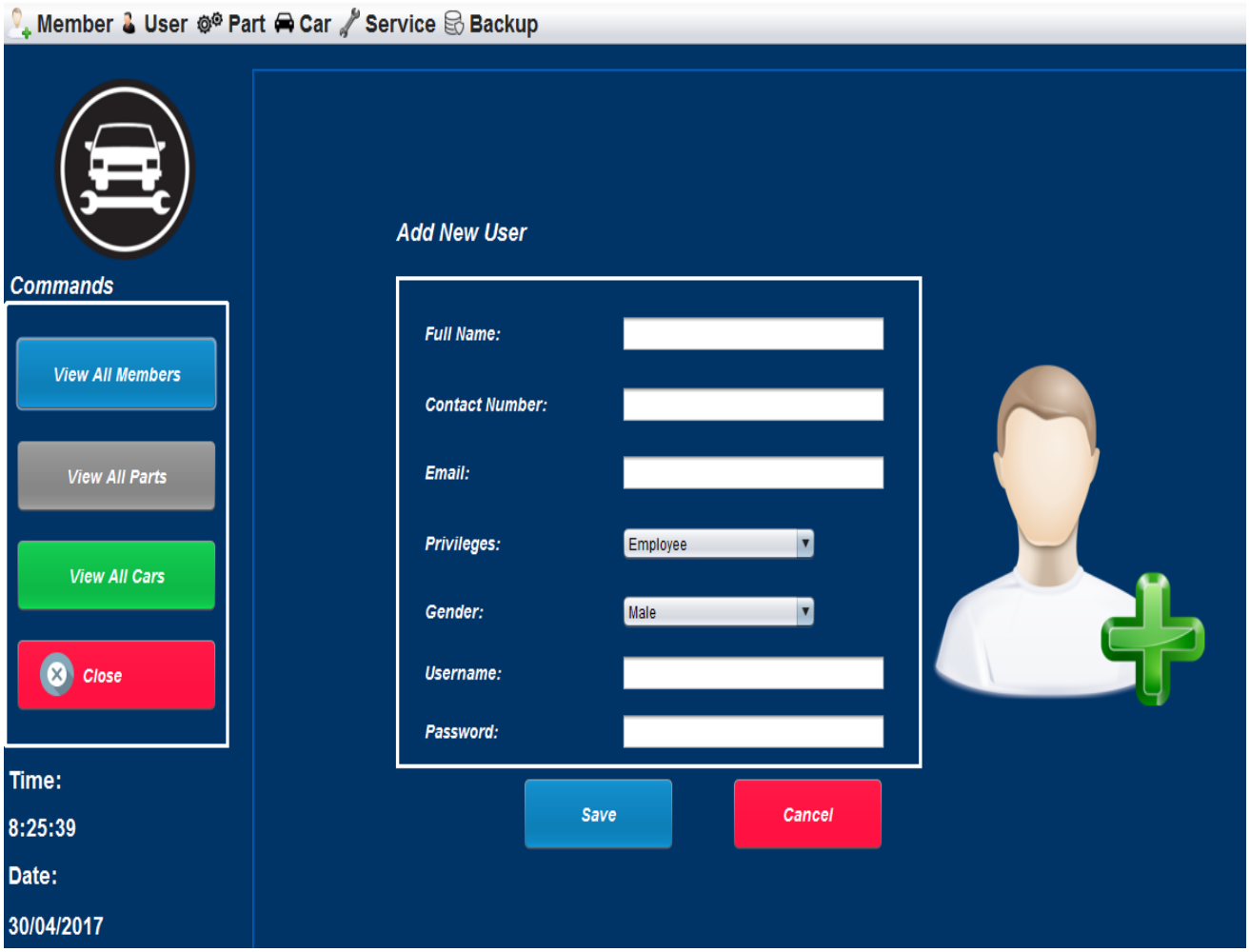


Σχήμα 5.12

Σχήμα 5.13 (Πατώντας το Backup δημιουργείται αντίγραφο του αρχείου .sqlite στο οποίο βρίσκεται το σχήμα της βάσης του συστήματος στο φάκελο Backup που βρίσκεται στο φάκελο του προγράμματος)

Προσθήκη Χρήστη

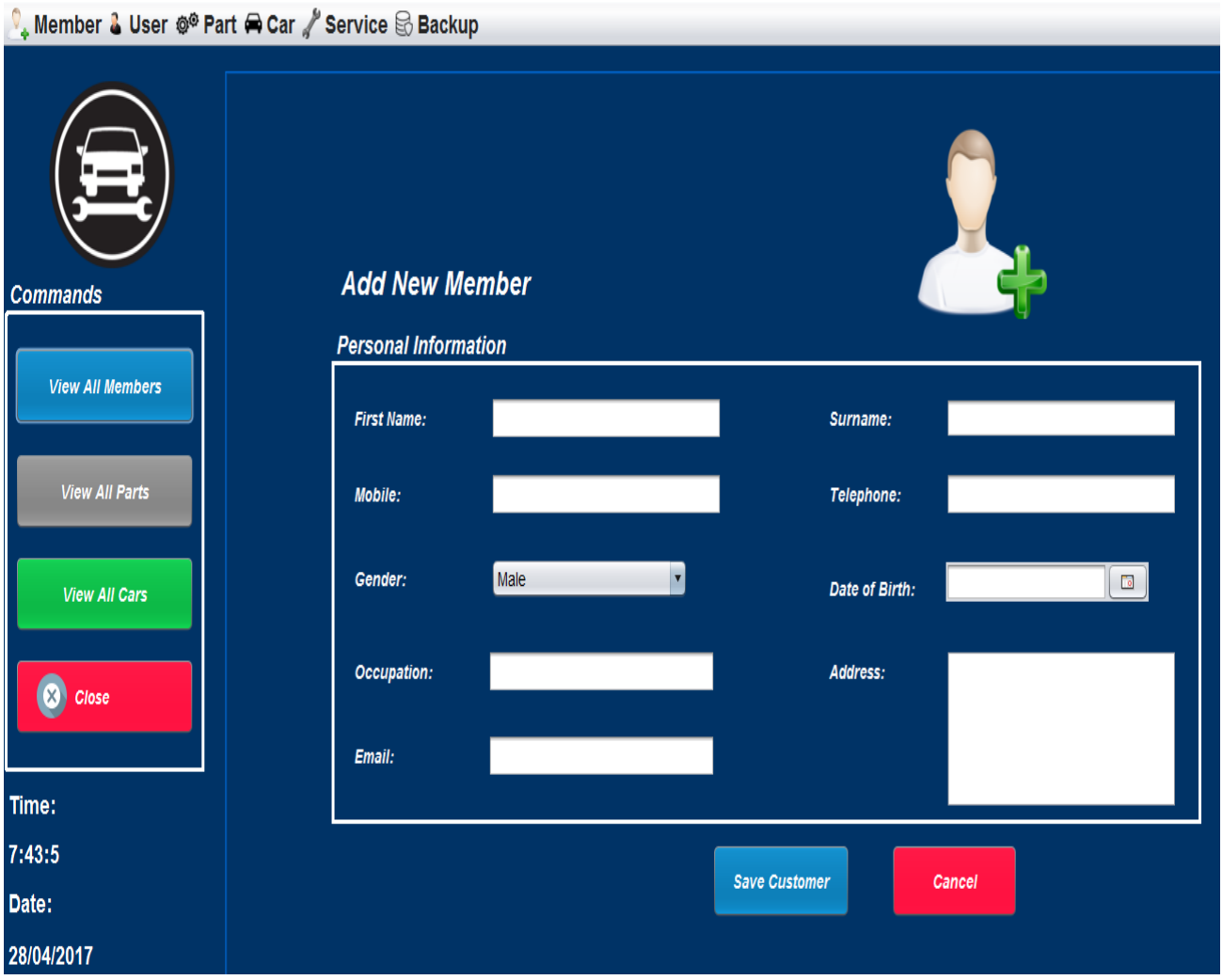
Πιο κάτω φαίνεται το στιγμιότυπο οθόνης της διεπαφής για τη λειτουργία κατά την οποία γίνεται η προσθήκη κάποιου χρήστη από κάποιο άλλο χρήστη. Εδώ πρέπει να συμπληρωθούν το όνομα του χρήστη, το τηλέφωνο επικοινωνίας, το φύλο, το ηλεκτρονικό του ταχυδρομείο, όπως επίσης και τα δικαιώματα που έχει. Ακόμη θα πρέπει να συμπληρωθούν τα πεδία για το username και password με τα οποία ο χρήστης θα μπορεί να εισέλθει στο σύστημα και τα οποία είναι μοναδικά για κάθε χρήστη.



Σχήμα 5.14

Προσθήκη Νέου Μέλους

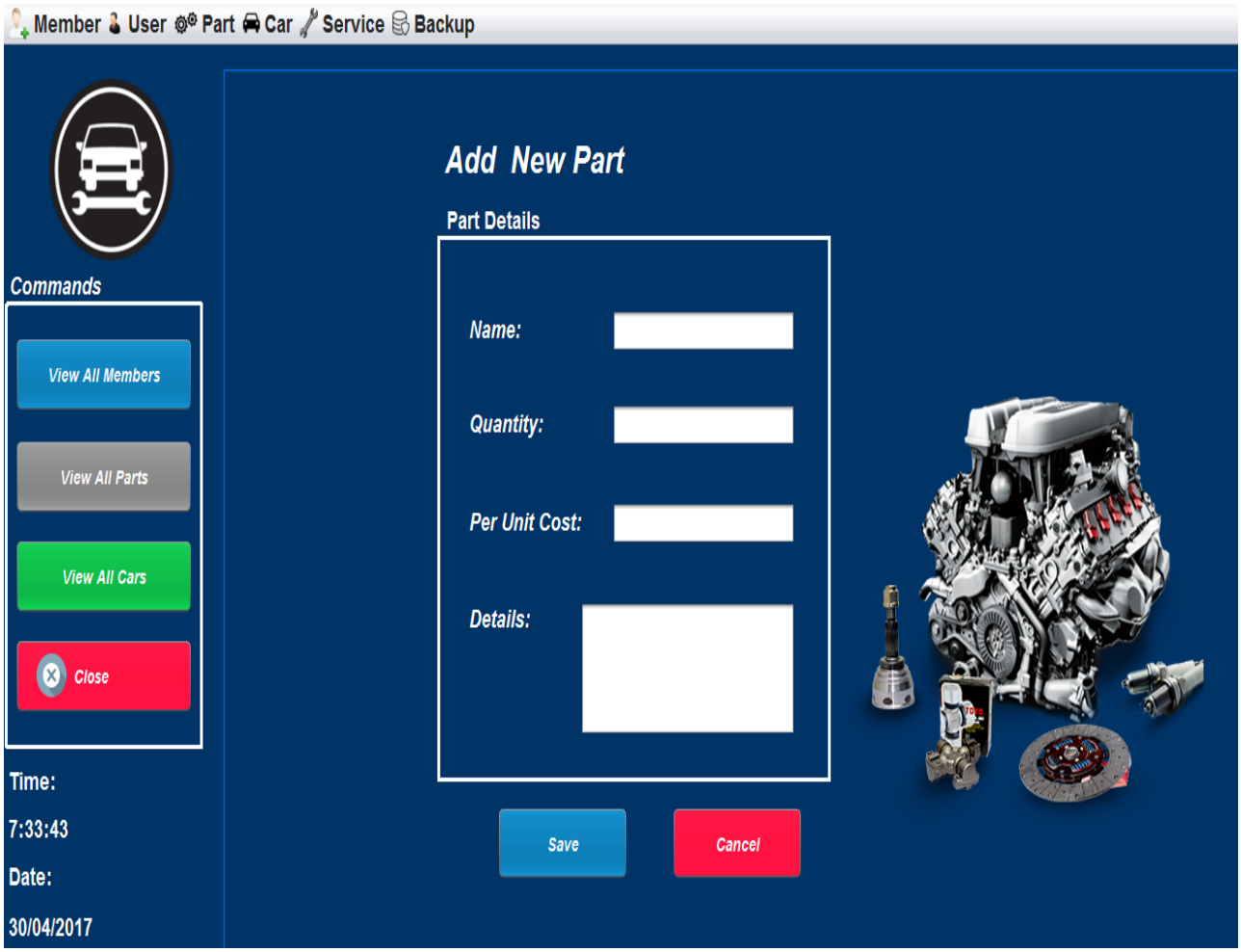
Σε αυτό το στιγμιότυπο παρουσιάζεται η λειτουργία κατά την οποία γίνεται η εισαγωγή ενός νέου μέλους στο σύστημα. Αφού καταχωρηθούν τα αντίστοιχα πεδία για το όνομα, το επίθετο, την ημερομηνία γέννησης, το φύλο, τον αριθμό τηλεφώνου, τον αριθμό του κινητού του τηλεφώνου, την διεύθυνσή και το email του, με το πάτημα του κουμπιού 'Save' γίνεται η καταχώρηση των δεδομένων του υπαλλήλου στη βάση.



Σχήμα 5.15

Προσθήκη Εξαρτήματος

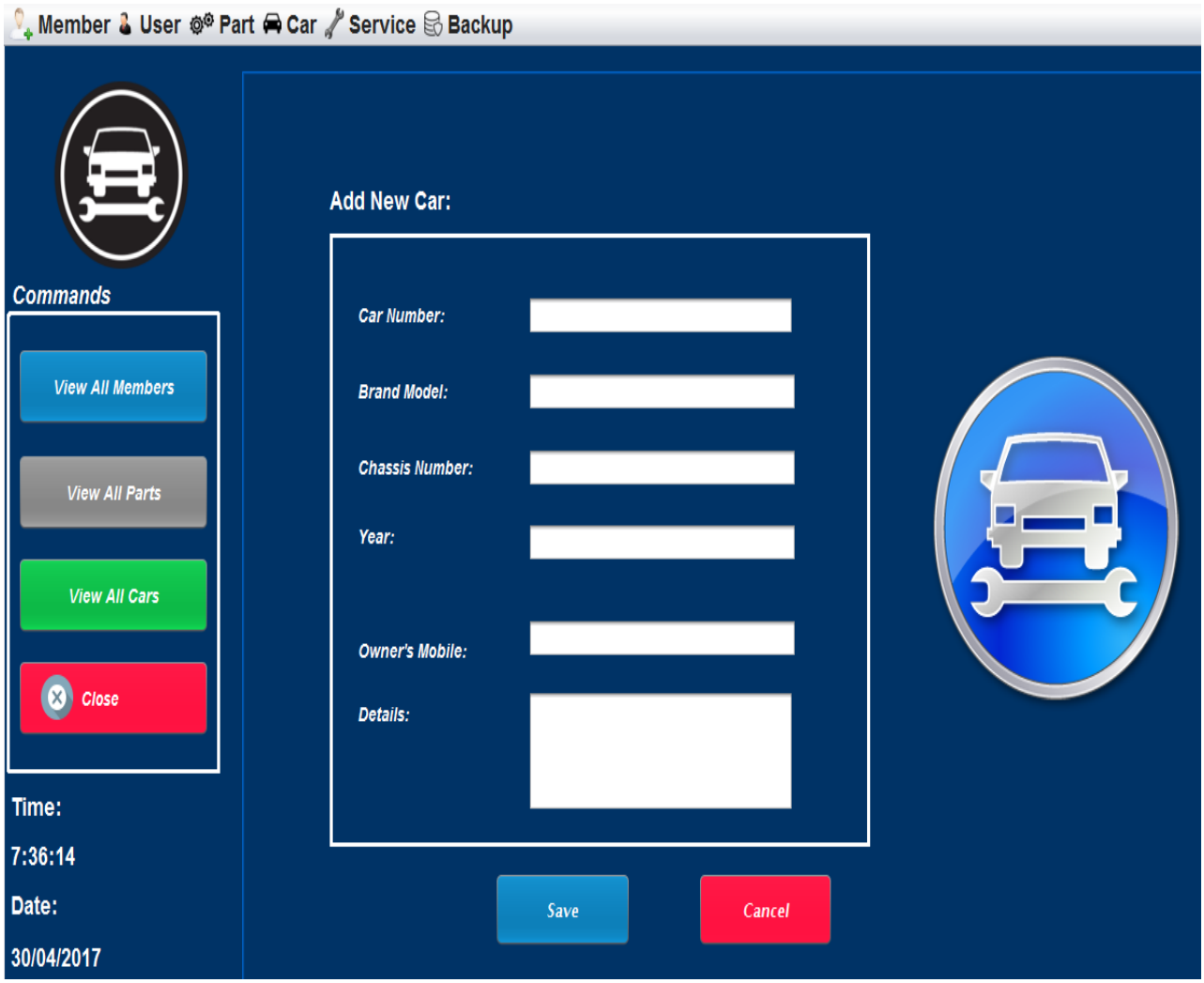
Στο πιο κάτω στιγμιότυπο παρουσιάζεται η λειτουργία κατά την οποία γίνεται η εισαγωγή κάποιου εξαρτήματος στο σύστημα. Αφού καταχωρηθούν τα αντίστοιχα πεδία για το όνομα, τη ποσότητα του και το ποσό που κοστίζει κάθε μονάδα και οι λεπτομέρειες που χαρακτηρίζουν το κάθε εξάρτημα, με το πάτημα του κουμπιού 'Save' γίνεται η καταχώρηση των δεδομένων του εξαρτήματος στη βάση.



Σχήμα 5.16

Προσθήκη Νέου Αυτοκινήτου

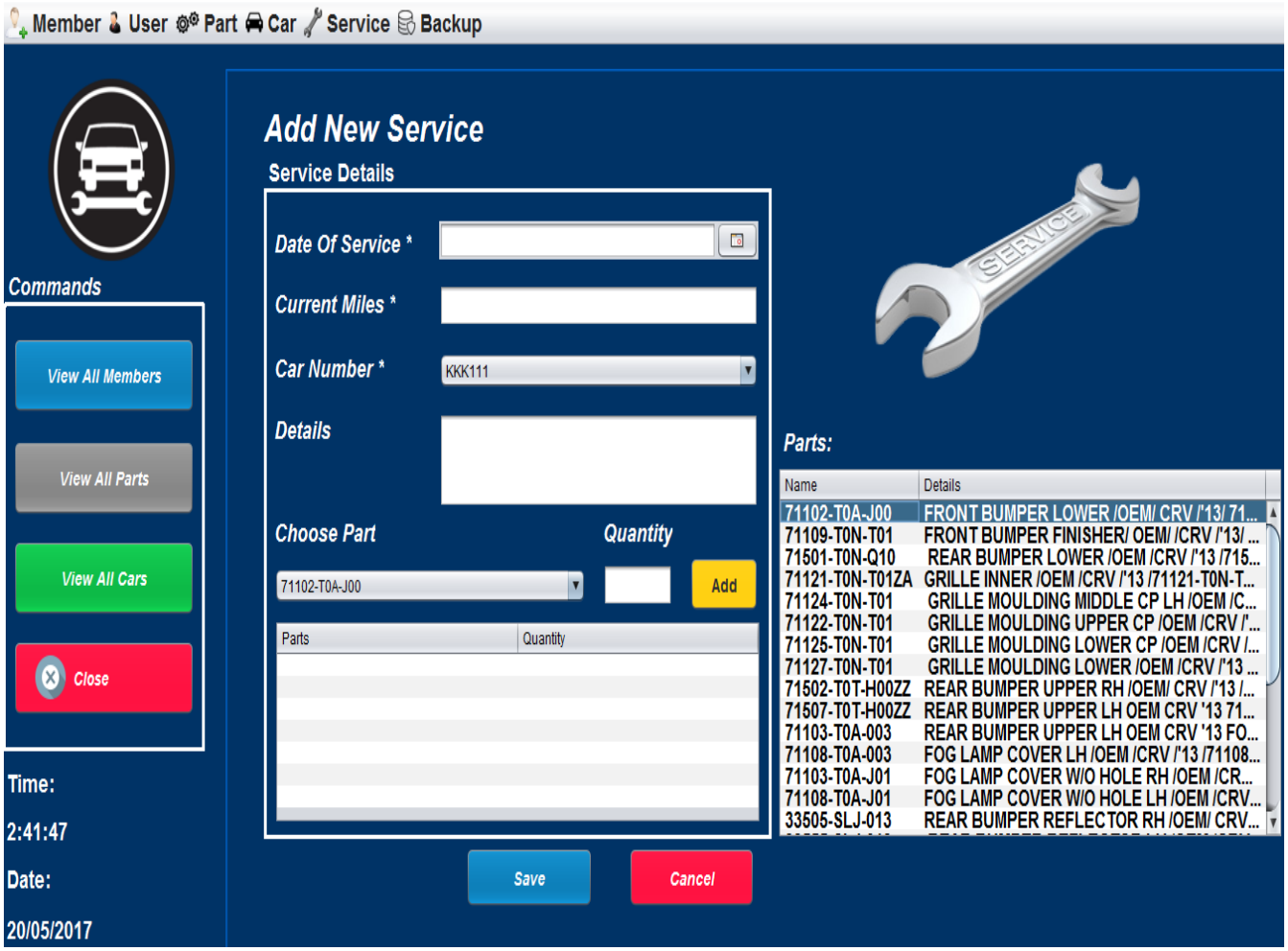
Σε αυτό το στιγμιότυπο παρουσιάζεται η λειτουργία κατά την οποία γίνεται η εισαγωγή ενός νέου αυτοκινήτου στο σύστημα. Αφού καταχωρηθούν τα αντίστοιχα πεδία για το car number, το brand model, το chassis number και το έτος κατασκευής του αυτοκινήτου, το κινητό τηλέφωνο του ιδιοκτήτη και τις λεπτομέρειες, πατώντας το κουμπί 'Save' γίνεται η καταχώρηση των στοιχείων στη βάση.



Σχήμα 5.17

Προσθήκη Νέου Service

Σε αυτό το στιγμιότυπο παρουσιάζεται η λειτουργία κατά την οποία γίνεται η εισαγωγή ενός νέου service στο σύστημα. Αφού καταχωρηθούν τα αντίστοιχα πεδία για την ημερομηνία που γίνεται το service, τα τρέχοντα μίλια του αυτοκινήτου, το car number και τις λεπτομέρειες, ο χρήστης μπορεί να καταχωρήσει τυχόν εξαρτήματα που χρησιμοποιηθήκαν επιλέγοντας τα εξαρτήματα, ακολούθως την ποσότητα για το κάθε εξάρτημα και add κάθε φορά. Κάτω δεξιά της οθόνης φαίνονται τα εξαρτήματα με τις λεπτομέρειες που τα περιγράφουν. Αφού τελειώσουμε με την καταχώρηση πατούμε το κουμπί 'Save' και καταχωρούνται τα στοιχεία στη βάση.



Σχήμα 5.18

Ενημέρωση / Διαγραφή Πελάτη/ Στοιχεία πελάτη σε PDF

Πιο κάτω παρουσιάζεται η λειτουργία στην οποία ο χρήστης μπορεί να αναζητήσει κάποιον πελάτη με βάση το όνομά του και τον αριθμό του κινητού του τηλέφωνου. Έτσι μπορεί να δει όλες τις προσωπικές πληροφορίες του πελάτη που βρίσκονται καταχωρημένα στο σύστημα, όπως όνομα, επίθετο, αριθμό κινητού και σταθερού τηλέφωνου, ημερομηνία γέννησης, το φύλο, τη διεύθυνσή του και το προσωπικό ηλεκτρονικό του ταχυδρομείο. Ακολούθως μπορεί να τροποποιήσει τα στοιχεία του μέλους πατώντας το κουμπί "Update Member", να διαγράψει το μέλος αν το επιθυμεί πατώντας το κουμπί "Delete Member", αλλά και να δημιουργήσει ένα αρχείο PDF με το προφίλ του μέλους πατώντας το "Member Into PDF" (Σχήμα 5.13.2).

Σχήμα 5.19.1

Παράδειγμα αρχείου PDF με το προφίλ του μέλους που επιλέγει ο χρήστης- ‘Member Into PDF’:

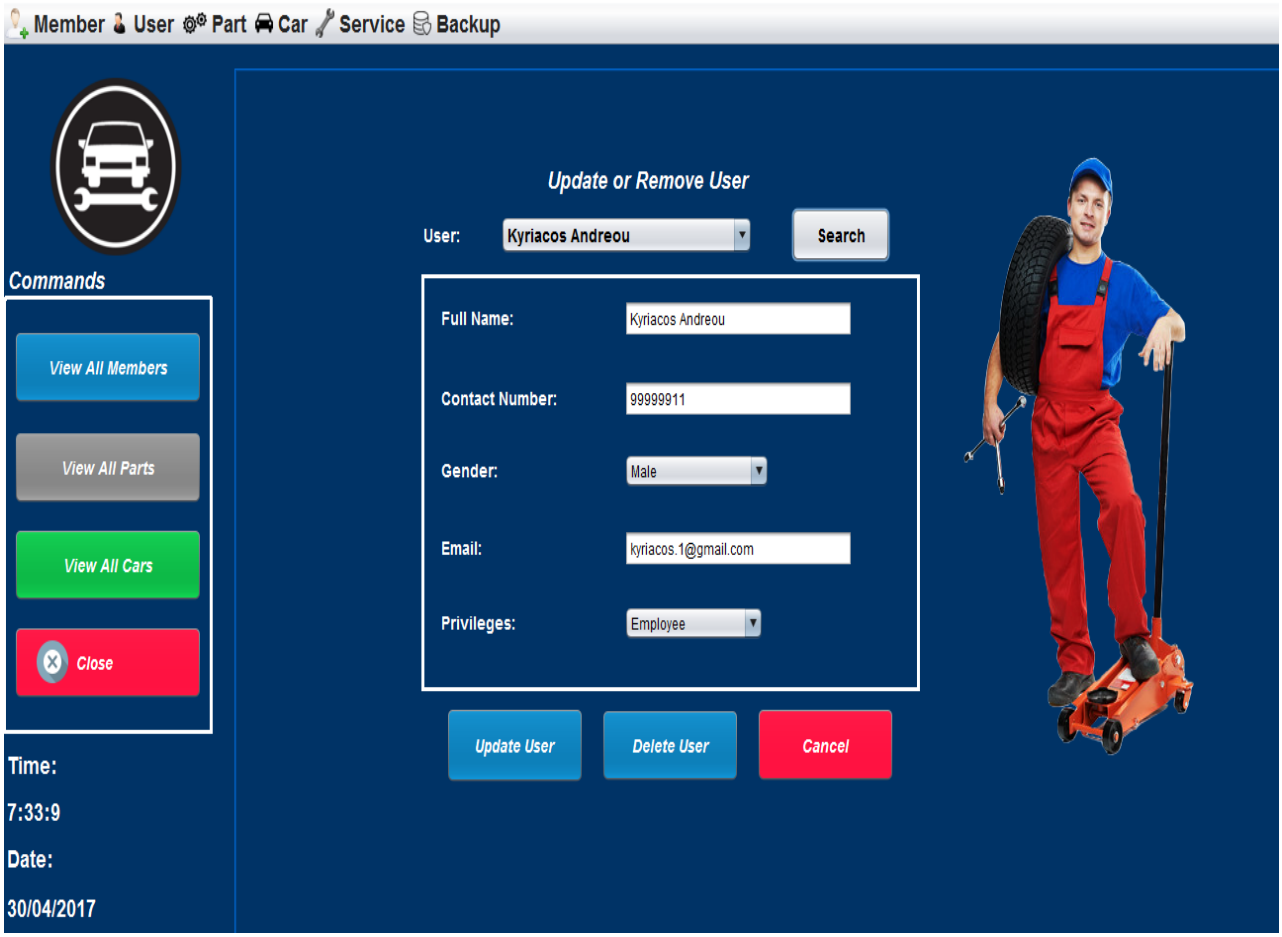
Personal Information:

Name:	Georgios
Surname:	Georgiou
Gender:	Male
Date of Birth:	12/04/1990
Email:	george.g@gmail.com
Occupation:	Student
Mobile:	99999999
Telephone:	
Address:	Address 1, Nicosia
Current Date:	21/05/2017

Σχήμα 5.19.2

Ενημέρωση / Διαγραφή Χρήστη

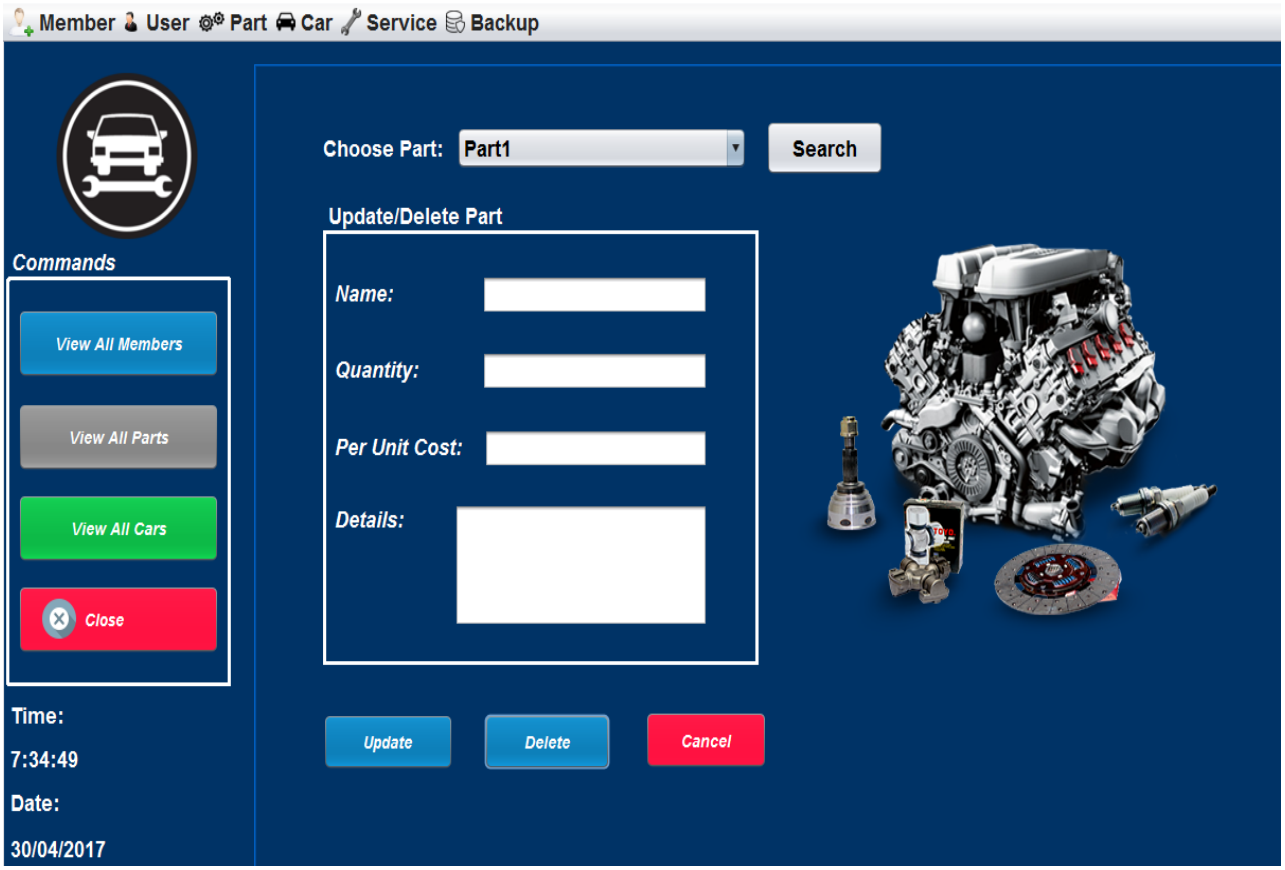
Πιο κάτω παρουσιάζεται η λειτουργία στην οποία ο χρήστης ο οποίος μπορεί να είναι και ο ιδιοκτήτης, μπορεί να αναζητήσει κάποιον άλλο χρήστη με βάση το ονοματεπώνυμο του. Έτσι μπορεί να δει όλες τις προσωπικές πληροφορίες που βρίσκονται καταχωρημένα στο σύστημα, όπως το ονοματεπώνυμο, το αριθμό κινητού τηλέφωνου, το φύλο, τη διεύθυνσή του, το προσωπικό ηλεκτρονικό του ταχυδρομείο και τα προνόμια που έχει. Ακολούθως μπορεί να τροποποιήσει τα στοιχεία του χρήστη πατώντας το κουμπί "Update User" και να διαγράψει το χρήστη από τη βάση αν το επιθυμεί πατώντας το κουμπί "Delete User ".



Σχήμα 5.20

Ενημέρωση / Διαγραφή Εξαρτήματος

Πιο κάτω παρουσιάζεται η λειτουργία στην οποία ο χρήστης έχει τη δυνατότητα να βρει ένα εξάρτημα το οποίο είναι καταχωρημένο στη βάση και να τροποποιήσει τα στοιχεία που περιγράφουν το εξάρτημα αυτό όπως επίσης και να διαγράψει το συγκεκριμένο εξάρτημα.



Σχήμα 5.21

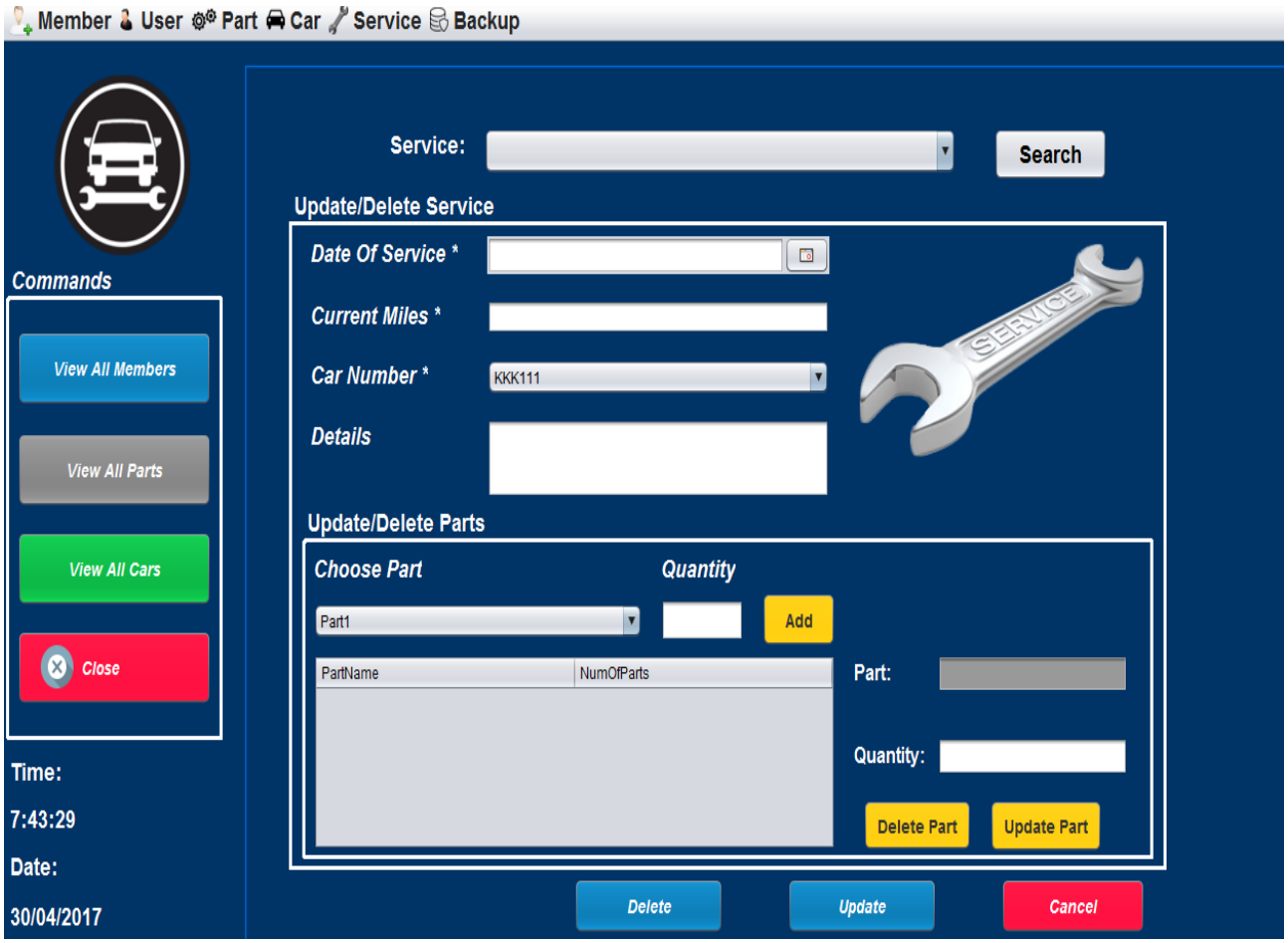
Ενημέρωση / Διαγραφή Αυτοκινήτου

Σε αυτό το στιγμιότυπο παρουσιάζεται η λειτουργία κατά την οποία γίνεται η επεξεργασία ή και η διαγραφή των στοιχείων ενός αυτοκινήτου . Τα στοιχεία αυτά όπως φαίνεται πιο κάτω είναι το car number, το brand model, το chassis number και το έτος κατασκευής του αυτοκινήτου, το κινητό τηλέφωνο του ιδιοκτήτη και τις λεπτομέρειες. Πατώντας το κουμπί 'Update Car' γίνεται η ενημέρωση των στοιχείων στη βάση ενώ με το 'Delete Car' γίνεται η διαγραφή του αυτοκινήτου.

Σχήμα 5.22

Ενημέρωση / Διαγραφή Service

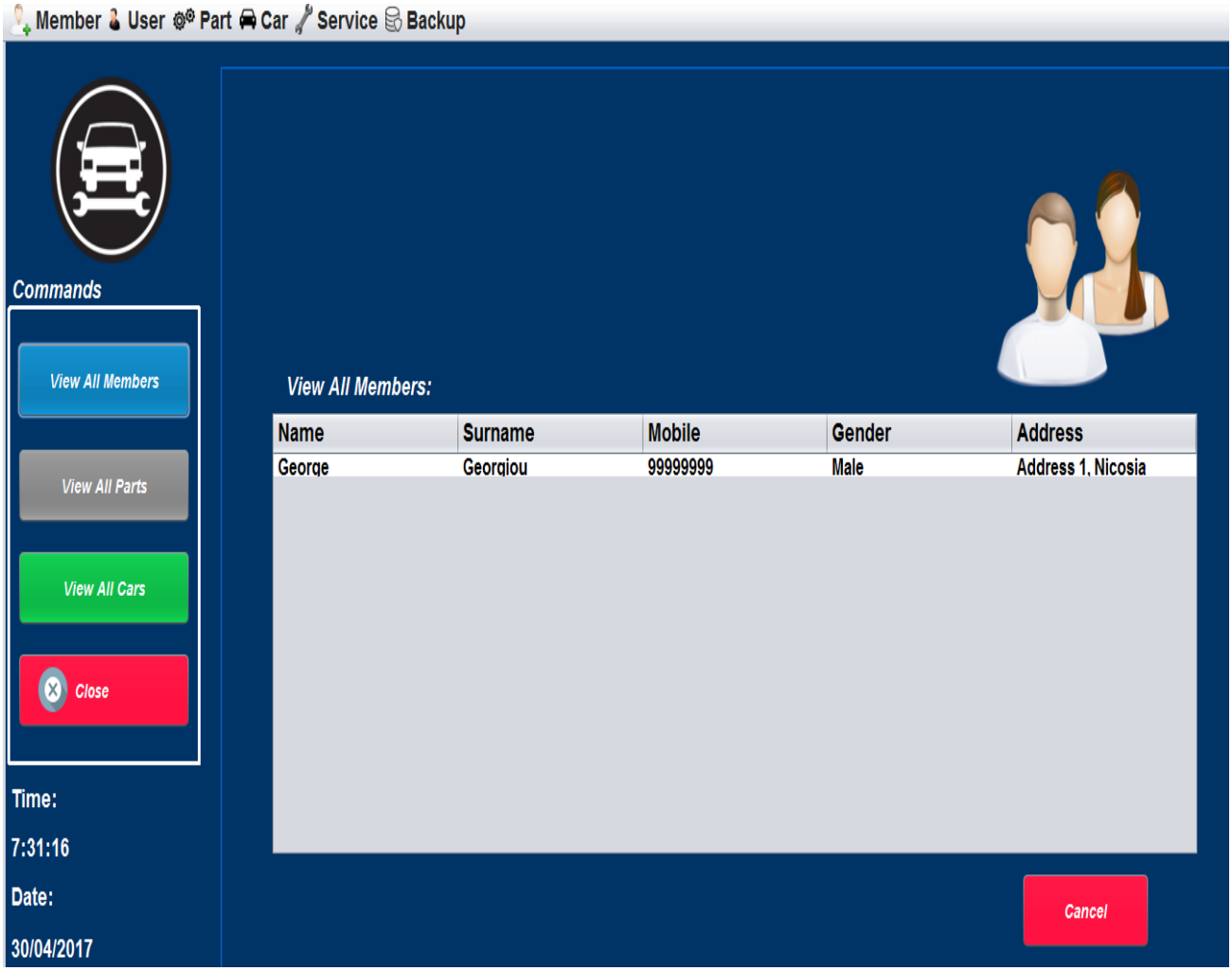
Σε αυτό το στιγμιότυπο παρουσιάζεται η λειτουργία κατά την οποία γίνεται η τροποποίηση ή και η διαγραφή ενός service στο σύστημα. Αφού ο χρήστης επιλέξει το service που επιθυμεί, εμφανίζονται τα πεδία για την ημερομηνία που γίνεται το service, τα τρέχοντα μίλια του αυτοκινήτου, το car number και τις λεπτομέρειες. Αν ο χρήστης θέλει να διαγράψει το service θα πρέπει να πατήσει το 'Delete' στο κάτω μέρος της οθόνης. Ο χρήστης μπορεί τροποποιήσει το καθένα από τα πεδία που αναφέρθηκαν πιο πάνω αλλά και τα τυχόν εξαρτήματα που φαίνονται στον πίνακα για τα parts, όπως επίσης να προσθέσει κι άλλα εξαρτήματα, αλλά και να διαγράψει. Για να προσθέσει εξάρτημα, απλά το επιλέγει, βάζει την ποσότητα κάτω από το 'Quantity' και πατά το 'add'. Πατώντας πάνω σε κάθε γραμμή του πίνακα μπορεί να αλλάξει την ποσότητα του κάθε εξαρτήματος αλλά και να διαγράψει το εξάρτημα με το 'Update Part' και 'Delete Part' αντίστοιχα. Αφού τελειώσουμε με τις αλλαγές πατούμε το κουμπί 'Update'.



Σχήμα 5.23

Προβολή όλων των μελών

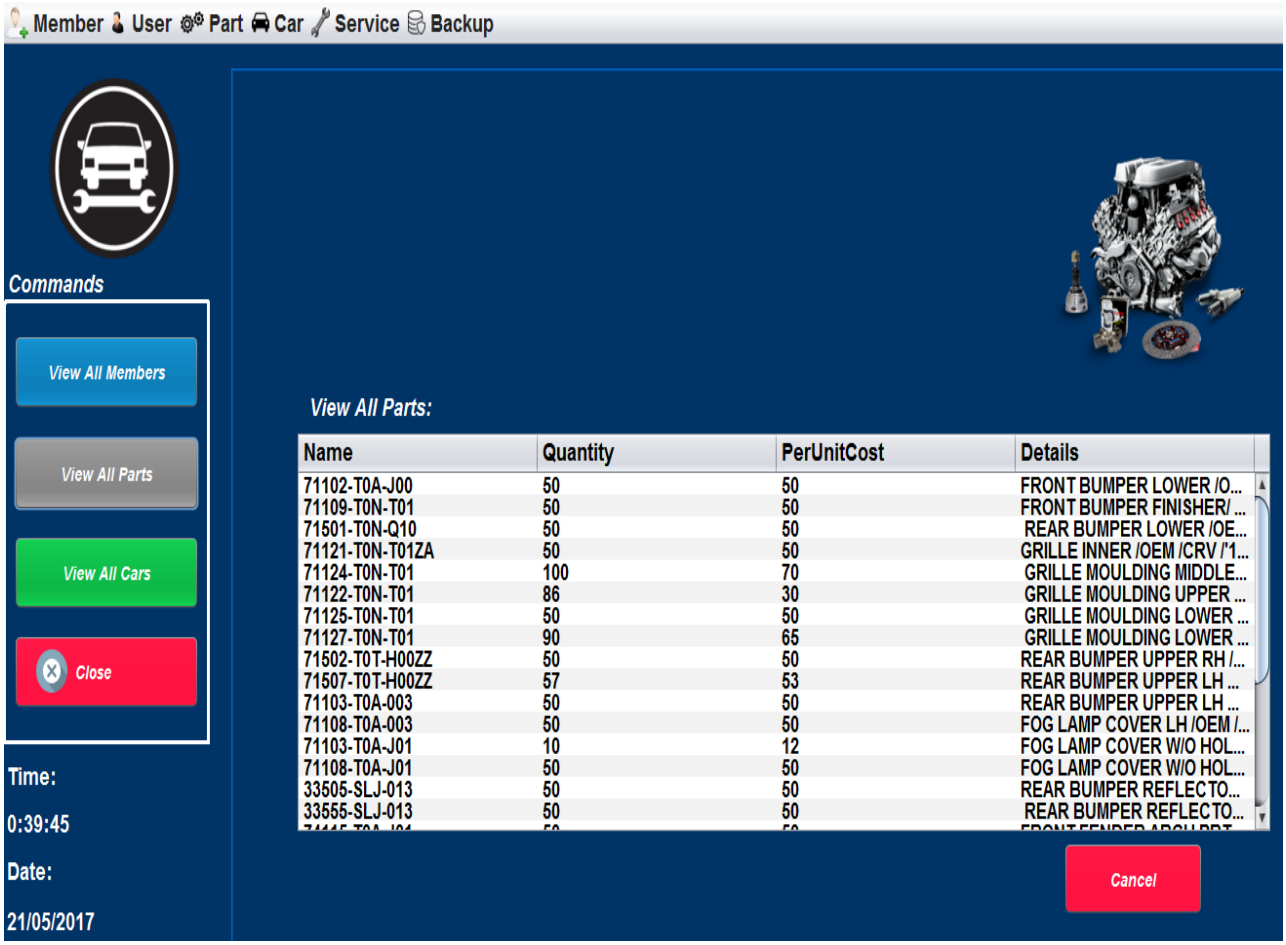
Πιο κάτω φαίνεται το στιγμιότυπο οθόνης του συστήματος στο οποίο ο χρήστης μπορεί να δει τα στοιχεία όλων των μελών που είναι καταχωρημένα στη βάση όπως το όνομα, το επίθετο, το φύλο, τον αριθμό του κινητού του τηλεφώνου και την διεύθυνσή.



Σχήμα 5.24

Προβολή όλων των εξαρτημάτων

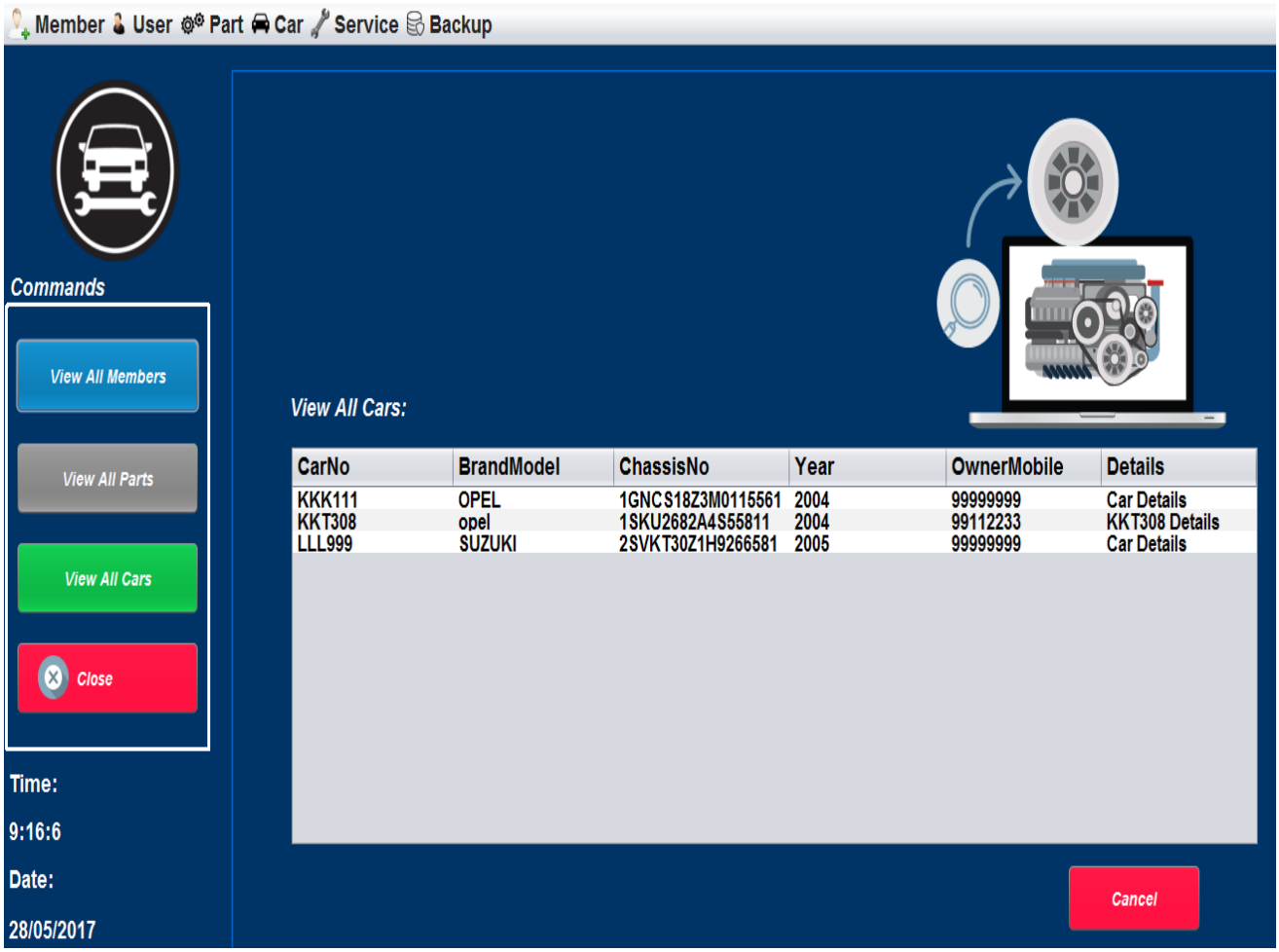
Πιο κάτω φαίνεται το στιγμιότυπο οθόνης του συστήματος στο οποίο ο χρήστης μπορεί να δει όλα τα εξαρτήματα που είναι καταχωρημένα στη βάση και τα στοιχεία τους τα οποία είναι το όνομα του κάθε εξαρτήματος, την ποσότητα του, το ποσό του κοστίζει και κάποιες λεπτομέρειες.



Σχήμα 5.25

Προβολή όλων των αυτοκινήτων

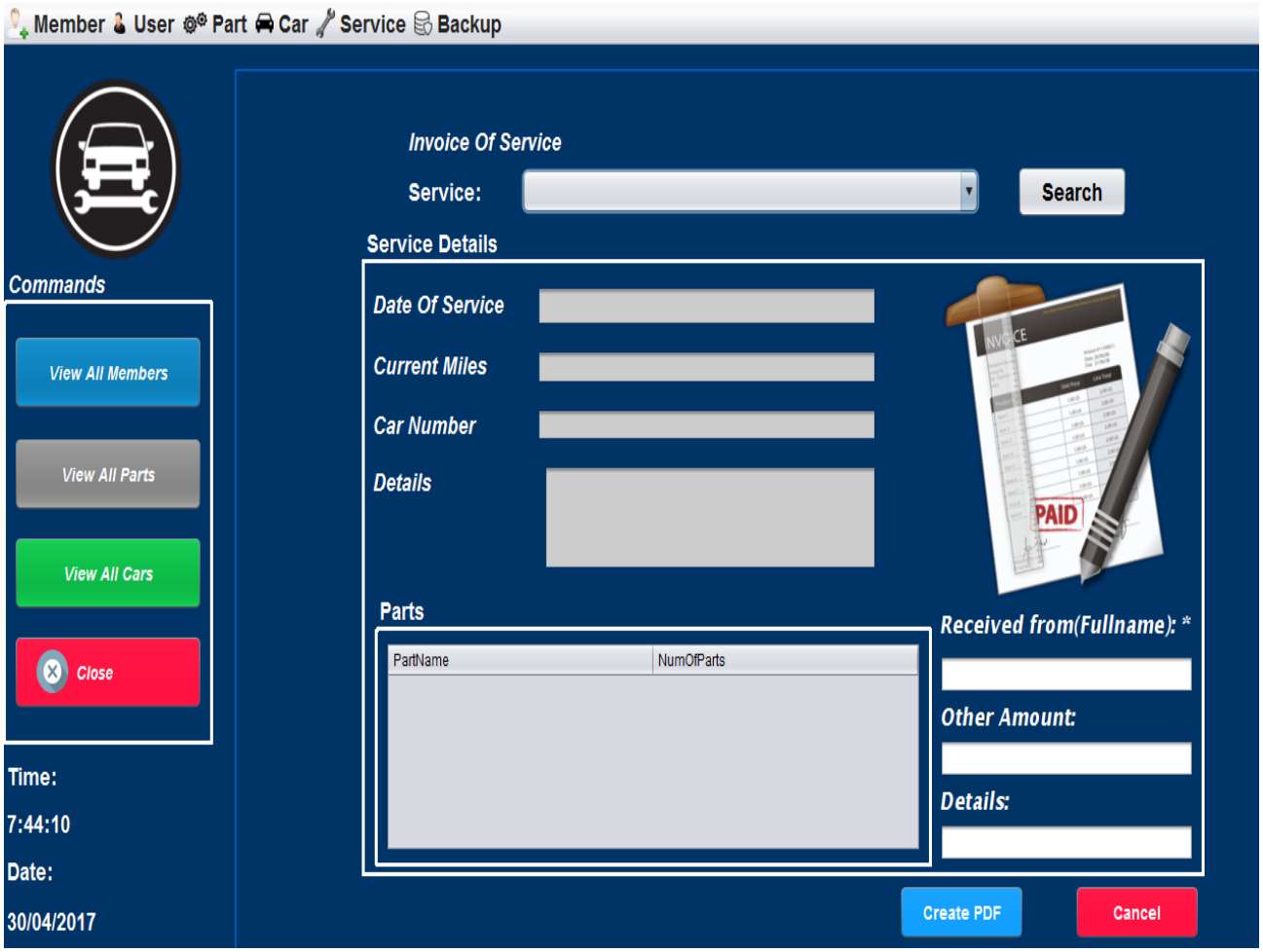
Πιο κάτω φαίνεται το στιγμιότυπο οθόνης του συστήματος στο οποίο ο χρήστης μπορεί να δει όλα τα αυτοκίνητα που είναι καταχωρημένα στη βάση και τα στοιχεία τους. Τα στοιχεία αυτά είναι το car number, το brand model, το chassis number, το έτος κατασκευής του αυτοκινήτου, το κινητό τηλέφωνο του ιδιοκτήτη και κάποιες λεπτομέρειες.



Σχήμα 5.26

Απόδειξη Πληρωμής Service

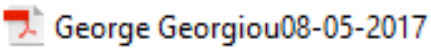
Σε αυτό το στιγμιότυπο παρουσιάζεται η λειτουργία κατά την οποία γίνεται η δημιουργία της απόδειξης πληρωμής του service του αυτοκινήτου την οποία μπορεί να αποθηκεύσει ο χρήστης αλλά και να τυπώσει. Αφού ο χρήστης επιλέξει το service που επιθυμεί, εμφανίζονται τα πεδία για την ημερομηνία που γίνεται το service, τα τρέχοντα μίλια του αυτοκινήτου, το car number και τις λεπτομέρειες. Ο χρήστης βλέπει επίσης στον πίνακα για τα parts τα τυχόν εξαρτήματα που χρησιμοποιήθηκαν. Πριν πατήσει το κουμπί 'Create PDF' θα πρέπει να βάλει στο κουτί κάτω από το 'Received from(Fullname):*' το όνομα αυτού στον οποίο θα βγει η απόδειξη. Επίσης μπορεί να βάλει κάποιο άλλο ποσό το οποίο θα χρεώσει και να γράψει λεπτομέρειες γι' αυτό. Αμέσως μόλις πατήσει το 'Create PDF' ανοίγει σε PDF το αρχείο αυτό στη μορφή Fullname+Date+'.pdf' το οποίο αποθηκεύεται σε φάκελο με όλες τις αποδείξεις έτσι ώστε να μπορεί ο χρήστης να βρει οποιανδήποτε εύκολα.



Σχήμα 5.27

Παράδειγμα αρχείου PDF για απόδειξη

-Παράδειγμα για το πως φαίνεται το αρχείο:



Σχήμα 5.22

-Παράδειγμα για το περιεχόμενο αρχείου:

RECEIPT

COMPANY NAME
Address, City, ST Zip Code
Phone|Fax

Equipments x Quantity	Per Unit Cost	Amount
Part1 x1	10.0	10.0
Part2 x1	20.0	20.0
Part3 x2	15.0	30.0

Other Amount:	0
Other Amount Details:	-
Reveived from	George Georgiou
the sum of Euro	60.0

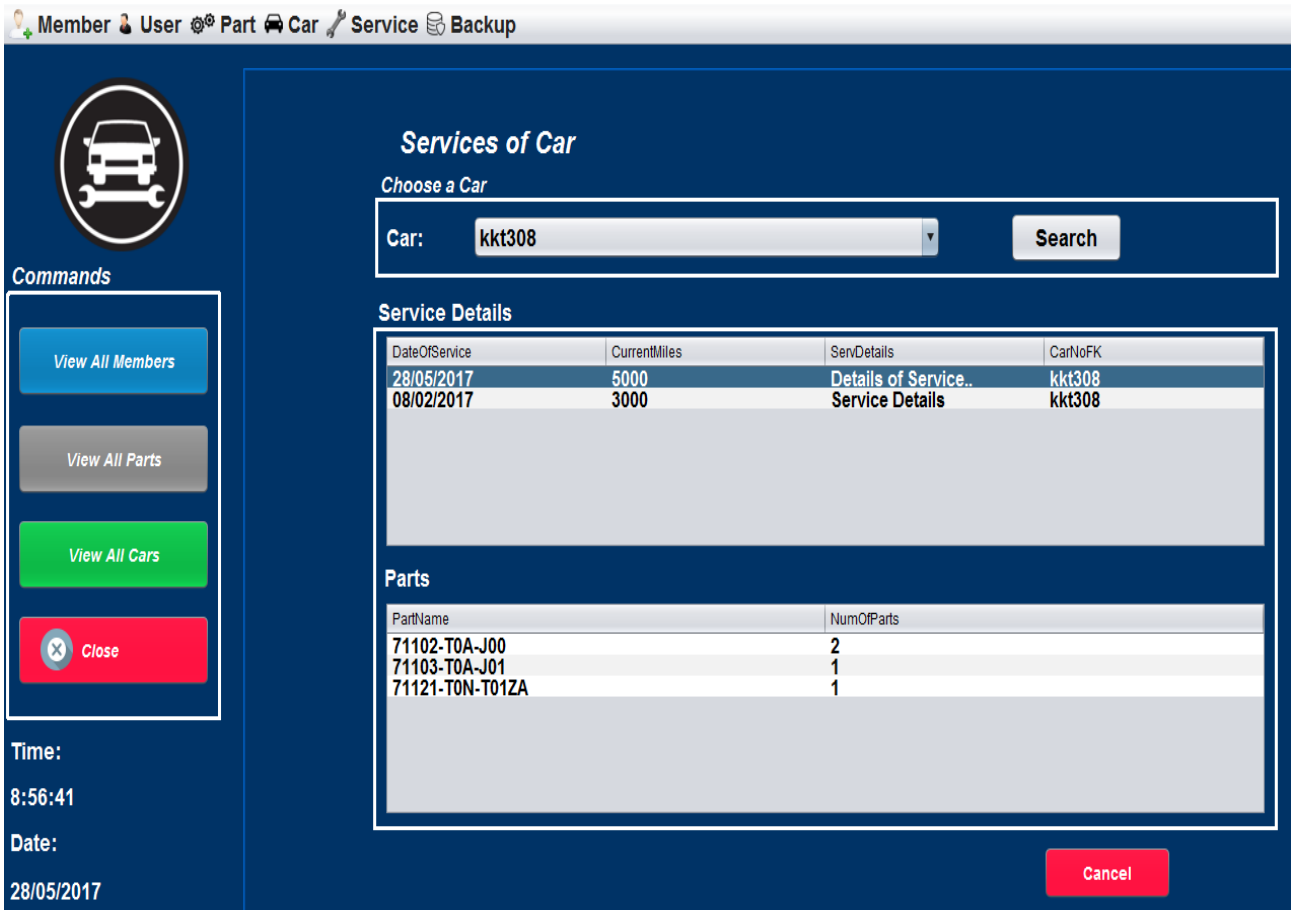
Date: 08/05/2017 Receiver: .....

Thank you for joining us

Σχήμα 5.28

Services Αυτοκινήτου

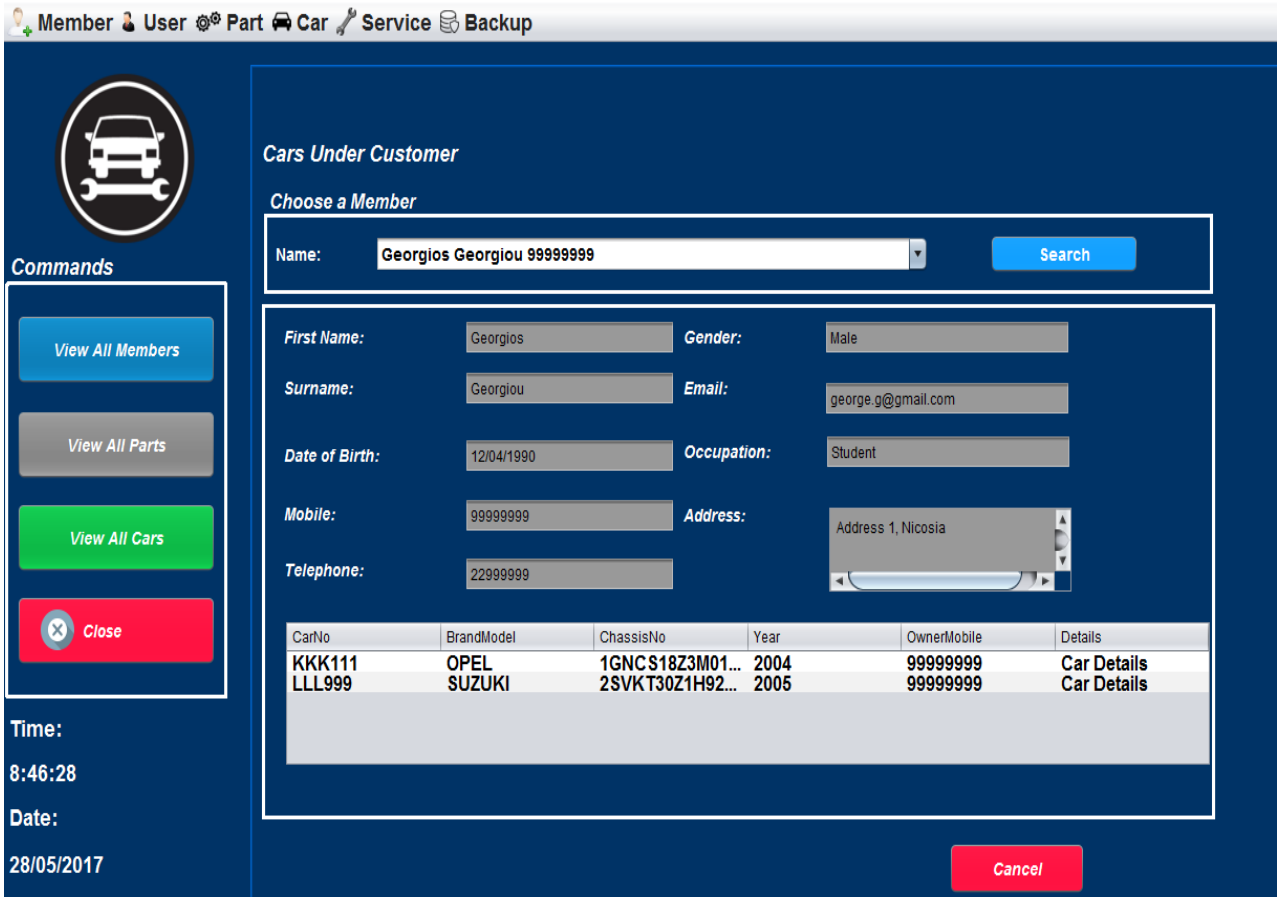
Πιο κάτω φαίνεται το στιγμιότυπο οθόνης του συστήματος στο οποίο ο χρήστης μπορεί να δει όλα τα services του κάθε αυτοκινήτου. Ο χρήστης επιλέγοντας το αυτοκίνητο που επιθυμεί και πατώντας το search, εμφανίζονται σε πίνακα τα services του αυτοκινήτου με την ημερομηνία του κάθε service, τα μίλια και οι λεπτομέρειες. Επίσης πατώντας πάνω στο κάθε service στον πίνακα, εμφανίζονται τα εξαρτήματα που χρησιμοποιήθηκαν στο συγκεκριμένο service στον αντίστοιχο πίνακα πιο κάτω.



Σχήμα 5.29

Αυτοκίνητα κάτω από κάποιο πελάτη

Πιο κάτω φαίνεται το στιγμιότυπο οθόνης του συστήματος στο οποίο ο χρήστης μπορεί να δει όλα τα αυτοκίνητα κάποιου πελάτη. Ο χρήστης επιλέγοντας τον πελάτη που επιθυμεί και πατώντας το search, εμφανίζονται τα στοιχεία του πελάτη και σε πίνακα τα στοιχεία των αυτοκινήτων που έχει. Τα στοιχεία που εμφανίζονται στον πίνακα είναι το car number, το brand model, το chassis number, το έτος κατασκευής του αυτοκινήτου, το κινητό τηλέφωνο του ιδιοκτήτη και κάποιες λεπτομέρειες.



Σχήμα 5.30

Κεφάλαιο 6

Συμπεράσματα και Μελλοντική Εργασία

6.1 Συμπεράσματα

Συμπερασματικά μέσα από την μέχρι τώρα εμπειρία που έχω αποκτήσει μπορώ να παραθέσω τα ακόλουθα. Η τεχνολογία αναπόφευκτα έχει κατακλύσει τη ζωή μας σε όλα τα φάσματα αυτής. Από την επιχειρηματική σκοπιά και θεώρηση, τη σήμερον ημέρα είναι απαραίτητο η κάθε επιχείρηση να μπορεί και να είναι σε θέση να αντιληφθεί αυτή την ανάγκη και να εναρμονιστεί με τις σύγχρονες τεχνικές που προσφέρει η τεχνολογία για να μπορεί να αναβαθμίζεται καθημερινά και να καθίσταται ανταγωνιστική στο σύγχρονο περιβάλλον που εξελίσσεται ραγδαία. Στην παρούσα εργασία είμαι πεπεισμένη ότι δώσαμε μια καθόλα επαναστατική λύση στον εργοδότη, στα υφιστάμενα προβλήματα που τον ταλάνιζαν και τον βασάνιζαν. Μέσα από το σύστημα αυτό, αυτοματοποιήσαμε λειτουργίες και διαδικασίες ικανές να διαφυλάξουν πολύτιμο χρόνο, χώρο και στο τέλος της ημέρας ανθρώπινο δυναμικό. Όλα αυτά εύκολα μεταφράζονται σε κέρδος αφού γίνεται τόσο μεγάλη εξοικονόμηση πόρων. Τέλος να αναφέρω ότι η καινοτομία δεν είναι κάτι στατικό. Θα ήταν ανόητο να θεωρείτο τέτοια. Είναι κάτι δυναμικό που κάθε φορά προκύπτει από την σωστή μελέτη και εντοπισμό των αναγκών και των προβλημάτων που δημιουργούνται. Έτσι επέρχεται σαν επισφράγισμα να λύσει και να εξαλείψει κάθε προηγούμενο κενό και αστοχία, μετατρέποντας τα σε απλές διαδικασίες και λειτουργίες για όλους. Τότε τέλος, μπορούμε να μιλάμε για μια καλύτερη ποιότητα εργασίας για όλους με εργονομία και ευκολία.

6.2 Μελλοντική Εργασία

Στην παρούσα διπλωματική εργασία, για τις ανάγκες αυτής που αναδύθηκαν μέσω της αλληλεπίδρασής που είχα με του επαγγελματίες του χώρου, ανάπτυξα μια Desktop εφαρμογή. Αυτή θα εγκατασταθεί στον υπολογιστή της εταιρείας ούτως ώστε να μπορούν να γίνονται όλες οι λειτουργίες που αναφέρθηκαν και αναλύθηκαν πιο πάνω σε άλλα κεφάλαια.
Σε μία μελλοντική προέκταση του συστήματος αυτού θα μπορούσαμε να υλοποιήσουμε περισσότερες λειτουργίες ή να κάνουμε ορισμένες τροποποιήσεις. Μία από αυτές θα μπορούσε όλο το σύστημα να υλοποιηθεί υπό τη μορφή ιστοσελίδας για να μπορούν να αρθούν οι οποιοιδήποτε περιορισμοί ως προς τη προσβασιμότητα. Μέσα σε αυτό το πλαίσιο θα μπορεί επίσης να αναπτυχθούν καινοτόμες εφαρμογές, αφού πλέον θα μπορούμε να έχουμε και την ίδια την πρόσβαση στο σύστημα του κάθε πελάτη. Για παράδειγμα θα μπορεί να υπάρχει ένα ημερολόγιο χρονοπρογραμματισμού για τις διαθέσιμες ημέρες και ώρες που θα μπορεί online να κάνει κράτηση ο κάθε πελάτης για το service του αυτοκινήτου του. Επίσης η βιωσιμότητα τέτοιου είδους επιχειρήσεων έγκειται στην ύπαρξη μιας σωστής επικοινωνιακής, διαρκής αλυσίδας μάρκετινγκ. Για το λόγο αυτό μέσω της χρησιμοποίησης ενός sms gateway server θα μπορούν να αποστέλλονται μηνύματα στο προσωπικό τηλέφωνο των πελατών. Αυτό θα γίνεται αυτόματα χωρίς επιβάρυνση του ιδιοκτήτη. Η συγκεκριμένη υπηρεσία θα μπορεί να ενεργοποιείται για την υπενθύμιση του δεδομένου κάθε φορά service καθώς και οποιασδήποτε άλλης ανάγκης προκύψει.

Βιβλιογραφία / Πηγές

[1] https://netbeans.org/

[2] http://launch4j.sourceforge.net/

[3] http://www.dmst.aueb.gr/dds/isdi/graph/indexw.htm

[4] https://www.tutorialspoint.com/jdbc/jdbc-introduction.htm

[5] https://www.w3.org/standards/webdesign/htmlcss

[6] https://download.fpiautoparts.com/

[7] https://addons.mozilla.org/en-US/firefox/addon/sqlite-manager/

Παράρτημα Α

Κώδικας του αρχείου javaconnect.java που το χρησιμοποιούμε για να συνδεόμαστε στη βάση κάθε φορά:

```
/**
 *
 * @author Maria
 */
import java.sql.*;
import javax.swing.*;
import org.sqlite.SQLiteConfig;

public class javaconnect {

    Connection conn = null;
   public static Connection ConnecrDB(){
      try{
         Class.forName("org.sqlite.JDBC");
         Connection conn = null;
         try {
      SQLiteConfig config = new SQLiteConfig();
      config.enforceForeignKeys(true);
     conn = DriverManager.getConnection(

"jdbc:sqlite:C:\\Users\\Maria\\Desktop\\Mechanic_Management_Software19\\MechanicSoft
ware.sqlite",
             config.toProperties());
   } catch (SQLException ex) {}

         return conn;
```

```
        }catch(Exception e){
            JOptionPane.showMessageDialog(null,e);
            return null;
        }
    }


}
```

Παράρτημα Β

Κώδικας για την εισαγωγή ενός μέλους στη βάση:

```
import java.awt.HeadlessException;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.swing.JButton;
import javax.swing.JOptionPane;



/**
 * @author Maria
 */
public class AddNewMember extends javax.swing.JPanel implements ActionListener{

    Connection conn = null;
    ResultSet rs  = null;
    PreparedStatement pst = null;
    int IDCodeMember = 0;
    String Name, Surname,Mobile ;

    private MyFormListenerAddNewMember formListenerAddNewMember;
    public void setFormListenerAddNewMember(MyFormListenerAddNewMember
formListenerAddNewMember) {
        this.formListenerAddNewMember = formListenerAddNewMember;
    }
```

```
    public AddNewMember() {
        initComponents();
        cmd_save.addActionListener(this);
        cmd_main_menu.addActionListener(this);
            }

    private void cmd_main_menuActionPerformed(java.awt.event.ActionEvent evt) {

    }

    private void cmd_saveActionPerformed(java.awt.event.ActionEvent evt) {

    }

    private void txt_mobileActionPerformed(java.awt.event.ActionEvent evt) {

    }

     @Override
    public void actionPerformed(ActionEvent e) {

        JButton source = (JButton) e.getSource();

        if (formListenerAddNewMember != null) {
            if (source == cmd_save) {

if(txt_name.getText().isEmpty()  ||  txt_surname.getText().isEmpty()  ||
txt_mobile.getText().isEmpty() ||
  txt_DateOfBirth.getDate()  ==  null  ||  Txt_Email.getText().isEmpty()  ||
Txt_Occupation.getText().isEmpty() ||
     Txt_Address.getText().isEmpty() )
```

```
{
   JOptionPane.showMessageDialog(null, "You have to fill all the fields");
}
else{              try{

 conn = javaconnect.ConnecrDB();

String sql = "Insert into Personal_Information (Name,Surname,Mobile,"
       + "Telephone,DateOfBirth,Gender,Email,Occupation,Address) "
       + "values (?,?,?,?,?,?,?,?,?)";
          pst=conn.prepareStatement(sql);

java.sql.Date sqldate = new java.sql.Date(txt_DateOfBirth.getDate().getTime());
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
String sqldate2 = sdf.format(sqldate);

          pst.setString(1, txt_name.getText());
          pst.setString(2, txt_surname.getText());
          pst.setString(3, txt_mobile.getText());
          pst.setString(4, txt_telephone.getText());
          pst.setString (5, sqldate2);
          String gender = (String) txt_Gender.getSelectedItem();
          pst.setString(6, gender);
          pst.setString(7, Txt_Email.getText());
          pst.setString(8, Txt_Occupation.getText());
          pst.setString(9, Txt_Address.getText());

               pst.execute();

       JOptionPane.showMessageDialog(null, "Member successfully added");


        Mobile = txt_mobile.getText();
        Name = txt_name.getText();
```

```
        Surname = txt_surname.getText();

       txt_name.setText(null);
       txt_surname.setText(null);
       txt_mobile.setText(null);
       txt_telephone.setText(null);
       Txt_Email.setText(null);
       Txt_Occupation.setText(null);
       Txt_Address.setText(null);
       txt_DateOfBirth.setDate(null);

        String sql2 ="SELECT last_insert_rowid() ";
        pst=conn.prepareStatement(sql2);
        rs = pst.executeQuery();

           if(rs.next()){
             int numero = rs.getInt(1);
              IDCodeMember = numero ;

           }


              pst.close();
              conn.close();
    }


       catch(SQLException | HeadlessException ee){

           JOptionPane.showMessageDialog(null, ee);
       }
formListenerAddNewMember.formEventPerformed(MyFormListenerAddNewMember.Type
.yes_no,
       IDCodeMember,Name,Surname,Mobile);
```

```
            }
            }
        }

        if(source == cmd_main_menu){
                txt_name.setText(null);
                txt_surname.setText(null);
                txt_telephone.setText(null);
                Txt_Email.setText(null);
                Txt_Occupation.setText(null);
                Txt_Address.setText(null);
                txt_DateOfBirth.setDate(null);
                txt_mobile.setText(null);

formListenerAddNewMember.formEventPerformed(MyFormListenerAddNewMember.Type
.cancel_main_menu,0,
        null,null,null);
        }
    }

}
```

Παράρτημα Γ

Κώδικας για τη λειτουργία Update και Delete Service:

```
import java.awt.HeadlessException;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import javax.swing.JButton;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import net.proteanit.sql.DbUtils;

/**
 *
 * @author Maria
 */


public class UpdateDeleteService extends javax.swing.JPanel implements ActionListener{

    Connection conn = null;
    ResultSet rs  = null;
    PreparedStatement pst = null;
    ArrayList<Integer> codearray = new ArrayList<Integer>();

 int row;
```

```
        String IDServ;

        String add1;

        String add3;

        String add4;

        String add6;
    /**
     * Creates new form UpdateDeleteEquipments
     */

    private MyFormListenerUpdateDeleteService formListenerUpdateDeleteService;

    public void setFormListenerUpdateDeleteService(MyFormListenerUpdateDeleteService
formListenerUpdateDeleteService) {
        this.formListenerUpdateDeleteService = formListenerUpdateDeleteService;
    }


    public UpdateDeleteService() {
        initComponents();
   //     conn = javaconnect.ConnecrDB();
        cmd_update_service.addActionListener(this);
        cmd_delete_service.addActionListener(this);
        cmd_main_menu.addActionListener(this);
        fillServiceCombo();
            fillPartsCombo();
    }
 public void fillPartsCombo(){
         try
    {
 String sql1 = "select *  from Equipments";
```

```
 conn = javaconnect.ConnecrDB();
          pst=conn.prepareStatement(sql1);
            rs = pst.executeQuery();

   txt_PartsCombo.removeAllItems();
   txt_PartsCombo.removeAllItems();

    while(rs.next()){
       txt_PartsCombo.addItem(rs.getString("Name"));
    }
   pst.close();
   rs.close();
   conn.close();
    }
    catch(Exception e)
    {
       JOptionPane.showMessageDialog(null, e);
    }
         finally{
try{
           }
catch(Exception e){
           }
}
    }
    public void fillServiceCombo(){
       ServiceSearchCombo.removeAllItems();

       try{

           conn = javaconnect.ConnecrDB();
           String sql = "SELECT * FROM Service ORDER BY CarNoFK, DateOfService
DESC, CurrentMiles" ;
           pst=conn.prepareStatement(sql);
```

```
            rs = pst.executeQuery();

                while(rs.next()){
                String resu =("ID: ") + rs.getString("IDService")+(" | ") +
rs.getString("CarNoFK")+(" | ") +rs.getString("DateOfService")+(" | Current Miles: ")
+rs.getString("CurrentMiles");
                ServiceSearchCombo.addItem(resu);
                codearray.add(rs.getInt("IDService"));

               }
                rs.close();
                pst.close();
                conn.close();
             }
          catch(Exception e){
             JOptionPane.showMessageDialog(null, e);
          }

      }
        public void fill_Parts_table(){

            try{
            java.sql.Date sqldate1000 = new
java.sql.Date(txt_DateOfService.getDate().getTime());
            SimpleDateFormat sdfk = new SimpleDateFormat("dd/MM/yyyy");
            String sqldate2000 = sdfk.format(sqldate1000);
            String cartext =(String) txt_CarCombo.getSelectedItem();

           conn = javaconnect.ConnecrDB();
          String sql1 = "Select Distinct PartName, NumOfParts from ServiceParts where
CurrentMilesFK = "+ txt_miles1.getText()
                        + " and CarNoFK = '" + cartext +"'"  +  " and DateOfServiceFK = '"+
sqldate2000 +"'" ;
```

```
        pst=conn.prepareStatement(sql1);
        rs = pst.executeQuery();

            table_parts.setModel(DbUtils.resultSetToTableModel(rs));

            pst.close();
            rs.close();

        }
        catch(SQLException | HeadlessException ee){
           JOptionPane.showMessageDialog(null, ee);
        }

    }


  public void fillCarCombo(){
         try
    {
String sql1 = "select *  from Car";
conn = javaconnect.ConnecrDB();
          pst=conn.prepareStatement(sql1);
            rs = pst.executeQuery();

 txt_CarCombo.removeAllItems();
  txt_CarCombo.removeAllItems();

    while(rs.next()){
       txt_CarCombo.addItem(rs.getString("CarNo"));
       //txt_CarCombo.addItem(rs.getString("CarNo"));
    }


  pst.close();
```

```
rs.close();
conn.close();
 }
 catch(Exception e)
 {
    JOptionPane.showMessageDialog(null, e);
 }
      finally{
        try{



        }
          catch(Exception e){
               }
     }
 }

 private void cmd_search_serviceActionPerformed(java.awt.event.ActionEvent evt) {
     try
{ String serv = (String) ServiceSearchCombo.getSelectedItem();
    String[] IDService = serv.split("\\s+");
    IDServ=IDService[1];

        conn = javaconnect.ConnecrDB();

    String sql = "SELECT  * FROM Service where IDService='"+ IDService[1] +"'";
         pst=conn.prepareStatement(sql);
         rs = pst.executeQuery();

          if(rs.next()){

         add1 = rs.getString("CurrentMiles");
        txt_miles1.setText(add1);
```

```
        add3 = rs.getString("ServDetails");
        txt_Serv_Details.setText(add3);

        add4 = rs.getString("CarNoFK");
        txt_CarCombo.setSelectedItem(add4);

        add6 = rs.getString("DateOfService");
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
        Date add66 = sdf.parse(add6);
        txt_DateOfService.setDate(add66);

        fillPartsCombo();
        fill_Parts_table();

         }
           rs.close();
            pst.close();
            conn.close();
          }
catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
        txt_UpDel_Part.setEditable(false);
}

private void cmd_update_serviceActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
}

private void cmd_delete_serviceActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void txt_CarComboActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    // TODO add your handling code here:
    }

    private void txt_PartsComboActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void cmd_add_partsActionPerformed(java.awt.event.ActionEvent evt) {

            if( txt_miles1.getText().isEmpty()  || txt_quantity.getText().isEmpty() ) {
                JOptionPane.showMessageDialog(null, "The fields  for Parts, Quantity and the fields with * must be completed");
            }               else{


            try{
               java.sql.Date sqldate10 = new java.sql.Date(txt_DateOfService.getDate().getTime());
           SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
           String sqldate20 = sdf.format(sqldate10);

String sql3= "Insert into ServiceParts (DateOfServiceFK, CurrentMilesFK, CarNoFK, PartName, NumOfParts) values (?, ?,?,?,?)";
           conn = javaconnect.ConnecrDB();
           pst=conn.prepareStatement(sql3);

            pst.setString(1, sqldate20);
            pst.setString(2, txt_miles1.getText());
            String Cars = (String) txt_CarCombo.getSelectedItem();
            pst.setString(3, Cars);
            String Parts = (String) txt_PartsCombo.getSelectedItem();
            pst.setString(4, Parts);
            pst.setString(5, txt_quantity.getText());
            pst.execute();
```

```
            JOptionPane.showMessageDialog(null, "Part(s) added");
            fill_Parts_table();




            pst.close();
            conn.close();

            txt_quantity.setText(null);



             }

             catch(SQLException | HeadlessException ee){

           JOptionPane.showMessageDialog(null, ee);
       }

//formListenerAddService.formEventPerformed(MyFormListenerAddService.Type.hello);


          }


    }

    private void ServiceSearchComboActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
```

```
    private void cmd_updateActionPerformed(java.awt.event.ActionEvent evt) {

if(txt_UpDel_Part.getText().isEmpty() || txt_UpDel_quantity.getText().isEmpty()){

        JOptionPane.showMessageDialog(null, "You have to choose a Part");
      }
      else{
        int code = codearray.get(row);
        try{
        java.sql.Date sqldate10 = new java.sql.Date(txt_DateOfService.getDate().getTime());
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
        String sqldate20 = sdf.format(sqldate10);

        String cartext =(String) txt_CarCombo.getSelectedItem();
           String sql = "UPDATE ServiceParts set NumOfParts = '" +
txt_UpDel_quantity.getText() +"'" + " where CurrentMilesFK = "+ txt_miles1.getText()
                    + " and CarNoFK = '" + cartext +"'"  +  " and DateOfServiceFK = '"
                + sqldate20 +"'" +  " and PartName = '"+ txt_UpDel_Part.getText() +"'" ;
           conn = javaconnect.ConnecrDB();
           pst = conn.prepareStatement(sql);
          // pst.setInt(1,code);
           pst.execute();
              rs.close();
              pst.close();
           JOptionPane.showMessageDialog(null, "Quantity of part '"+
txt_UpDel_Part.getText()+" ' changed");

           txt_UpDel_Part.setText(null);
           txt_UpDel_quantity.setText(null);



           //show_users_jtable(ExeDay);
        } catch(Exception e){
           JOptionPane.showMessageDialog(null, e);
```

```
        }
        finally{
            try{
                rs.close();
                pst.close();

            }
            catch(Exception e){
            }
        }
    }
  fill_Parts_table();
 }

 private void delete_exerciseActionPerformed(java.awt.event.ActionEvent evt) {

    if(txt_UpDel_Part.getText().isEmpty() || txt_UpDel_quantity.getText().isEmpty()){

        JOptionPane.showMessageDialog(null, "You have to choose a Part");
    }
    else{
        int code = codearray.get(row);
        try{
        java.sql.Date sqldate10 = new java.sql.Date(txt_DateOfService.getDate().getTime());
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
        String sqldate20 = sdf.format(sqldate10);
        String cartext =(String) txt_CarCombo.getSelectedItem();
            String sql = "Delete from ServiceParts where CurrentMilesFK = "+
txt_miles1.getText()
                    + " and CarNoFK = '" + cartext +"'"  +  " and DateOfServiceFK = '"
                + sqldate20 +"'" +  " and PartName = '"+ txt_UpDel_Part.getText() +"'" ;
            conn = javaconnect.ConnecrDB();
            pst = conn.prepareStatement(sql);
           // pst.setInt(1,code);
```

```
            pst.execute();
               rs.close();
               pst.close();
            JOptionPane.showMessageDialog(null, "Part deleted");

            txt_UpDel_Part.setText(null);
            txt_UpDel_quantity.setText(null);


            //show_users_jtable(ExeDay);
         } catch(Exception e){
            JOptionPane.showMessageDialog(null, e);
         }
         finally{
            try{
               rs.close();
               pst.close();

            }
            catch(Exception e){
            }
         }
      }
    fill_Parts_table();
   }

   private void table_partsMouseClicked(java.awt.event.MouseEvent evt) {
try{
          row = table_parts.getSelectedRow();
         txt_UpDel_Part.setText((table_parts.getModel().getValueAt(row,  0).toString())); // =
(table_programs.getModel().getValueAt(row, 0).toString());
           txt_UpDel_quantity.setText(table_parts.getModel().getValueAt(row, 1).toString()); //
= (table_programs.getModel().getValueAt(row, 1).toString());
```

```
  }
  catch(Exception e){
     JOptionPane.showMessageDialog(null, e);
  }

}

private void cmd_main_menuActionPerformed(java.awt.event.ActionEvent evt) {
   // TODO add your handling code here:
}

 @Override
public void actionPerformed(ActionEvent e) {

   // We can use the event "source" to figure
   // out which button was pressed.
   JButton source = (JButton) e.getSource();

   if (formListenerUpdateDeleteService != null) {
      if (source == cmd_update_service) {

if(txt_DateOfService.getDate().toString().isEmpty()|| txt_miles1.getText().isEmpty()){

      JOptionPane.showMessageDialog(null, "You have to choose a Service");
   }
 else{


         try{
             conn = javaconnect.ConnecrDB();
      java.sql.Date sqldate1 = new java.sql.Date(txt_DateOfService.getDate().getTime());
      SimpleDateFormat sdfr = new SimpleDateFormat("dd/MM/yyyy");
      String sqldate24 = sdfr.format(sqldate1);
         String cartext =(String) txt_CarCombo.getSelectedItem();
```

```
String miles =  txt_miles1.getText();


String sql2 = "UPDATE ServiceParts set  CurrentMilesFK = "'+ txt_miles1.getText()
        + '" , CarNoFK = '" + txt_CarCombo.getSelectedItem() + '" ,
DateOfServiceFK = '"
    + sqldate24 +"' where CurrentMilesFK = "'+ add1
        + '" and CarNoFK = '" + add4 +  '" and DateOfServiceFK = "'+ add6 + "'" ;

String sql = "UPDATE Service set DateOfService = '" + sqldate24  + "' , CurrentMiles
= "'+  txt_miles1.getText()
        + '" , CarNoFK = '" + txt_CarCombo.getSelectedItem()  +  '" , ServDetails
= "'+ txt_Serv_Details.getText()
    + '" where IDService= '" + IDServ +"'";


pst=conn.prepareStatement(sql2);
pst.execute();
 pst.close();
  conn.close();

 conn = javaconnect.ConnecrDB();
pst=conn.prepareStatement(sql);
pst.execute();

   pst.close();
  conn.close();
  JOptionPane.showMessageDialog(null, "Service updated");


  txt_quantity.setText(null);
  txt_miles1.setText(null);
  txt_Serv_Details.setText(null);
   txt_DateOfService.setCalendar(null);
```

```
        DefaultTableModel dm = (DefaultTableModel)table_parts.getModel();
        dm.getDataVector().removeAllElements();
        dm.fireTableDataChanged();

          }

          catch(SQLException | HeadlessException ee){

        JOptionPane.showMessageDialog(null, ee);
      }


formListenerUpdateDeleteService.formEventPerformed(MyFormListenerUpdateDeleteServi
ce.Type.hello);


        }

        }

        else if(source == cmd_delete_service){

            if(txt_DateOfService.getDate().toString().isEmpty()||
txt_miles1.getText().isEmpty()){

        JOptionPane.showMessageDialog(null, "You have to choose a Service");
      }
    else{


          try{
              conn = javaconnect.ConnecrDB();
```

```
String sql2 = "DELETE FROM ServiceParts where CurrentMilesFK = "'+ add1
                    + "' and CarNoFK = "' + add4 +  "' and DateOfServiceFK = "'
                + add6 + """ ;
 String sql = "DELETE FROM Service where IDService= "' + IDServ +"""";


        pst=conn.prepareStatement(sql2);
         pst.execute();
        pst=conn.prepareStatement(sql);

         pst.execute();

           pst.close();
           conn.close();

           JOptionPane.showMessageDialog(null, "Service deleted");

           txt_quantity.setText(null);
           txt_miles1.setText(null);
           txt_Serv_Details.setText(null);
           txt_DateOfService.setCalendar(null);
         DefaultTableModel dm = (DefaultTableModel)table_parts.getModel();
         dm.getDataVector().removeAllElements();
        dm.fireTableDataChanged();

            }

            catch(SQLException | HeadlessException ee){

         JOptionPane.showMessageDialog(null, ee);
    }
```

```
formListenerUpdateDeleteService.formEventPerformed(MyFormListenerUpdateDeleteServi
ce.Type.hello);



        }
        }
         else if(source == cmd_main_menu){



           txt_quantity.setText(null);
         txt_miles1.setText(null);
         txt_Serv_Details.setText(null);
         txt_DateOfService.setCalendar(null);
          DefaultTableModel dm = (DefaultTableModel)table_parts.getModel();
        dm.getDataVector().removeAllElements();
        dm.fireTableDataChanged();

formListenerUpdateDeleteService.formEventPerformed(MyFormListenerUpdateDeleteServi
ce.Type.goodbye);
         }
   }
   }


}
```

Παράρτημα Δ

Κώδικας αρχείου ViewAllCars το οποίο εμφανίζει όλα τα αυτοκίνητα που είναι καταχωρημένα στη βάση:

```
/**
 *
 * @author Maria
 */
import java.awt.Font;
import java.awt.HeadlessException;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JButton;
import javax.swing.JOptionPane;
import javax.swing.table.JTableHeader;
import net.proteanit.sql.DbUtils;


public class ViewAllCars extends javax.swing.JPanel implements ActionListener {
    Connection conn = null;
   ResultSet rs  = null;
   PreparedStatement pst = null;

    private MyFormListenerViewCars formListenerViewCars;

  public void setFormListenerViewAllCars(MyFormListenerViewCars
formListenerViewCars) {
      this.formListenerViewCars = formListenerViewCars;
```

```
}

/**
 * Creates new form ViewAllCars
 */
public ViewAllCars() {
    initComponents();
    cmd_main_menu.addActionListener(this);
    tableheader();
   filltable();
}

private void cmd_main_menuActionPerformed(java.awt.event.ActionEvent evt) {

}

 public void tableheader(){
   Font f = new Font("Arial", Font.BOLD, 18);
  JTableHeader header = ViewAllCars_table.getTableHeader();
  header.setFont(f);
}

public void filltable(){

      try{
    String sql1 = "select CarNo,BrandModel,ChassisNo,Year,OwnerMobile from Car";
    conn = javaconnect.ConnecrDB();
    pst=conn.prepareStatement(sql1);
    rs = pst.executeQuery();

        ViewAllCars_table.setModel(DbUtils.resultSetToTableModel(rs));

        pst.close();
        rs.close();
```

```
            conn.close();
        }
        catch(SQLException | HeadlessException ee){
           JOptionPane.showMessageDialog(null, ee);
        }

    }

    public void actionPerformed(ActionEvent e) {

        JButton source = (JButton) e.getSource();

        if (formListenerViewCars != null) {
            if (source == cmd_main_menu) {

formListenerViewCars.formEventPerformed(MyFormListenerViewCars.Type.main);

            }

        }
    }

}
```

Παράρτημα Ε

Κώδικας του αρχείου Login για να εισέλθει κάποιος στο σύστημα:

```
/**
 *
 * @author Maria
 */

import java.awt.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.ParseException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.*;

public class login extends javax.swing.JFrame {
            Connection conn = null;
            ResultSet rs  = null;
            PreparedStatement pst = null;
  public login() {
        initComponents();


    }

    private void cmd_loginActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

        String sql="select * from Login where Username=? and Password=?";
        try{

            conn = javaconnect.ConnecrDB();
            pst=conn.prepareStatement(sql);
            pst.setString(1, txt_username.getText());
            pst.setString(2, txt_password.getText());

            rs=pst.executeQuery();
```

```
        if(rs.next()){
            JOptionPane.showMessageDialog(null, "Username and Password is correct");
              try {
                Toolkit toolkit =  Toolkit.getDefaultToolkit ();
                Dimension dim = toolkit.getScreenSize();
                MainFrame c;
        try {
            c = new MainFrame();
            c.setVisible(true);
        } catch (ParseException ex) {
            Logger.getLogger(login.class.getName()).log(Level.SEVERE, null, ex);
        }

            } catch (SQLException ex) {
                 Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            this.setVisible(false);

           /* login s= new login();
            s.setVisible(true);*/
        }

        else{
            JOptionPane.showMessageDialog(null, "Username and Password is not correct");
        }


         pst.close();
         rs.close();
         conn.close();
    }
    catch (Exception e){
        JOptionPane.showMessageDialog(null,e);


    }




}
```

```
    private void txt_passwordActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void txt_usernameActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(login.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(login.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(login.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(login.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
```

```
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new login().setVisible(true);
            }
        });

    }

}
```