Ατομική Διπλωματική Εργασία

ΔΙΑΧΕΙΡΙΣΗ ΠΡΟΤΙΜΗΣΕΩΝ ΧΡΗΣΤΗ ΓΙΑ ΙΔΙΩΤΙΚΟΤΗΤΑ ΣΕ ΥΠΗΡΕΣΙΕΣ ΔΙΑΔΙΚΤΥΟΥ ΜΕ ΕΠΙΓΝΩΣΗ ΠΛΑΙΣΙΟΥ ΧΡΗΣΗΣ.

Ιωσηφίνα Ιωάννου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2012

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Διαχείριση προτιμήσεων χρήστη για ιδιωτικότητα σε υπηρεσίες διαδικτύου με επίγνωση πλαισίου χρήσης.

Ιωσηφίνα Ιωάννου

Επιβλέπων Καθηγητής
Γεωργία Καπιτσάκη

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

Μάιος 2012

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω την επιβλέπων καθηγήτριά μου Δρ. Γεωργία Καπιτσάκη, Επίκουρη καθηγήτρια του τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου, για την ευκαιρία την οποία μου πρόσφερε να ασχοληθώ με την παρούσα ατομική διπλωματική εργασία. Σαν επιβλέπων λέκτοράς μου, μου έδωσε τις απαραίτητες κατευθυντήριες γραμμές για την ολοκλήρωση της εργασίας αυτής, αλλά παράλληλα μου έδωσε την ευκαιρία να εξασκήσω την κρίση μου και τις ικανότητές μου.

Περίληψη

Η ιδιωτικότητα των στοιχείων του χρήστη αλλά και των προτιμήσεών του αποτελεί ένα από τα μεγαλύτερα προβλήματα που απασχολούν τον κλάδο της Πληροφορικής στις μέρες μας. Είναι αναγκαίο να αναπτυχθούν συστήματα τα οποία να μπορούν να προφυλάσσουν πληροφορίες για τους χρήστες τους, τις οποίες δεν θέλουν να είναι προσβάσιμες από τρίτους.
Στην παρούσα ατομική διπλωματική εργασία κύριο στόχο αποτελεί η διαχείριση των προτιμήσεων του χρήστη για ιδιωτικότητα σε υπηρεσίες διαδικτύου με επίγνωση πλαισίου χρήσης. Ο έλεγχος της διαχείρισης των προτιμήσεων κάθε χρήστη θα μπορεί να τεθεί σε εφαρμογή και να ελεχθεί μέσω της υλοποίηση ενός συνόλου σεναρίων.
Η παρούσα ατομική διπλωματική εργασία θα πραγματοποιηθεί σαν επέκταση της αρχιτεκτονικής που ήδη υπάρχει γύρω από τις υπηρεσίες διαδικτύου με αποτέλεσμα να λαμβάνονται υπόψη οι προτιμήσεις του χρήστη για την ιδιωτικότητα του, χωρίς όμως να είναι απαραίτητο να τροποποιηθεί η υλοποίηση των υπηρεσιών διαδικτύου.

Περιεχόμενα

Κεφάλαιο **1** **Εισαγωγή** **1**

1.1 Κίνητρο 1

1.2 Στόχοι 2

1.3 Τεχνολογίες Διαδικτύου 3

1.3.1 XML 3

1.3.2 XML Schema 3

1.3.3 HTTP 4

1.4 Υπηρεσίες Διαδικτύου 4

1.4.1 Υπηρεσίες Διαδικτύου και SOA 4

1.4.2 Σύνδεση SOAP, UDDI, WSDL 6

1.4.3 SOAP 7

1.4.4 UDDI 7

1.4.5 WSDL 8

1.5 Πλαίσιο Χρήσης 9

1.5.1 Ορισμός Πλαισίου Χρήσης 9

1.5.2 Πλαίσιο Χρήσης και Υπηρεσίες Διαδικτύου 9

1.6 Ιδιωτικότητα 10

1.6.1 Ορισμός Ιδιωτικότητας 10

1.6.2 Απαιτήσεις Ιδιωτικότητας 10

1.6.3 Προσωπικά Δεδομένα 11

Κεφάλαιο **2** **Υφιστάμενη Αρχιτεκτονική Υπηρεσιών Διαδικτύου**……. **12**

2.1 Σύστημα Διαχείρισης Πλαισίου Χρήσης 12

2.1.1 Apache Axis 2 12

2.1.2 Apache CXF 13

2.2 XML APIs 13

2.2.1 SAX 13

2.2.2 DOM 14

Κεφάλαιο **3** **Σχεδίαση του Συστήματος και Υλοποίηση** **16**
3.1 Ανάλυση Τεχνολογιών Υλοποίησης Σεναρίου 17
3.1.1 Apache Axis2 17
3.1.2 Eclipse Java EE IDE for Web Developers 17
3.1.3 Apache Tomcat 7 17
3.1.4 DOM 18
3.2 Ανάλυση web services 18
3.2.1 Welcoming Service 18
3.2.2 User Profile Service 18
3.2.3 Current Location Service 19
3.2.4 Buy Service 20
3.2.5 Books Store Service 20
3.2.6 Books List Service 21
3.3 Ανάλυση Parser 22
3.3.1 Επεξήγηση XML Schema (upp.xsd) 22
3.3.2 Επεξήγηση Privacy Preferences 23
3.4 Επεξήγηση Υφιστάμενης Αρχιτεκτονικής Axis 2 Handler 26
3.5 Ανάλυση Axis 2 Handler 27
3.5.1 Ανάλυση Plugins Σεναρίου 27
3.5.1.1 Books List Plugin 27
3.5.1.2 Buy Plugin 28
3.5.1.3 Welcome Plugin 29
3.5.2 Ανάλυση Providers Σεναρίου 30
3.5.2.1 Books List Provider 30
3.5.2.2 Books Store Provider 30
3.5.2.3 Buying Provider 30
3.5.2.4 Welcoming Provider 31

Κεφάλαιο 4 **Σχεδίαση Σεναρίου χρήσης επέκτασης του συστήματος** **32**
4.1 Ανάλυση Σεναρίου 32
4.2 Απαιτήσεις Συστήματος 34
4.2.1 Λειτουργίες Συστήματος 34
4.2.2 Περιορισμοί Συστήματος 34

4.3 Data Flow Diagrams 35
4.3.1 WelcomingService 35
4.3.2 UserProfileService 36
4.3.3 CurrentLocationService 42
4.3.4 BuyService 45
4.3.5 BooksStoreService 47
4.3.6 BooksListService 50
4.4 Διαγράμματα Κλάσεων 56
4.4.1 WelcomingService 56
4.4.2 UserProfileService 57
4.4.3 CurrentLocationService 59
4.4.4 BuyService 60
4.4.5 BooksStoreService 61
4.4.6 BooksListService 63

Κεφάλαιο **5** **Συμπεράσματα**…….. **66**
5.1 Προστασία προσωπικών δεδομένων στις υπηρεσίες διαδικτύου 66

Β ι β λ ι ο γ ρ α φ ί α …… **67**

Π α ρ ά ρ τ η μ α Α…….……..…….…… **Α-1**

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο 1
1.2 Στόχοι 2
1.3 Τεχνολογίες Διαδικτύου 3
1.3.1 XML 3
1.3.2 XML Schema 3
1.3.3 HTTP 4
1.4 Υπηρεσίες Διαδικτύου 4
1.4.1 Υπηρεσίες Διαδικτύου και SOA 4
1.4.2 Σύνδεση SOAP, UDDI, WSDL 6
1.4.3 SOAP 7
1.4.4 UDDI 7
1.4.5 WSDL 8
1.5 Πλαίσιο Χρήσης 9
1.5.1 Ορισμός Πλαισίου Χρήσης 9
1.5.2 Πλαίσιο Χρήσης και Υπηρεσίες Διαδικτύου 9
1.6 Ιδιωτικότητα 10
1.6.1 Ιδιωτικότητα 10
1.6.2 Απαιτήσεις Ιδιωτικότητας 10
1.6.3 Προσωπικά Δεδομένα 11

1.1 Κίνητρο

Ζούμε σε μία εποχή όπου τα πάντα περιστρέφονται γύρω από τον άξονα του διαδικτύου, αφού το σημαντικότερο πράγμα που μπορεί να σε καθορίσει είναι οι γνώσεις σου γύρω από το διαδίκτυο αλλά και η πρόσβαση που έχεις σε αυτό. Ζούμε στην εποχή όπου όλες οι πληροφορίες αλλά και οι υπηρεσίες βρίσκονται κάποια κλικ μακριά από τον καθένα μας καθώς τα πάντα μπορούν να βρεθούν μέσα από το διαδίκτυο. Έχει γίνει βασική προϋπόθεση

στην ζωή μας ότι όλες οι συσκευές οι οποίες χρησιμοποιούμε στην καθημερινότητά μας πρέπει να έχουν πρόσβαση στο διαδίκτυο.
Οι προγραμματιστές παρέχουν τα λογισμικά τους ως Υπηρεσίες Διαδικτύου για να μπορούν οι χρήστες να έχουν εύκολη και γρήγορη πρόσβαση σε αυτά. Όμως αυτό δεν είναι αρκετό. Σε πολλές περιπτώσεις χρειάζεται να προσφέρουμε στους χρήστες υπηρεσίες οι οποίες μπορούν να λαμβάνουν υπόψη τις προτιμήσεις των ίδιων των χρηστών τους για να εκτελούν τις ανάλογες λειτουργίες. Οι προτιμήσεις του χρήστη καθώς και οι πληροφορίες που αφορούν τον χρήστη αναφέρονται ως πλαίσιο χρήσης ή αλλιώς context και αποτελεί ένα σημαντικό μέρος της ανάπτυξης των υπηρεσιών διαδικτύου.

1.2 Στόχοι

Η παρούσα ατομική διπλωματική εργασία έχει ως στόχο τη διαχείριση των προτιμήσεων του χρήστη για ιδιωτικότητα σε υπηρεσίες διαδικτύου με επίγνωση πλαισίου χρήσης. Ο έλεγχος της διαχείρισης των προτιμήσεων αυτών θα γίνει με την υλοποίηση ενός συνόλου σεναρίων. Το σενάριο το οποίο θα υλοποιηθεί θα λαμβάνει υπόψη τις προτιμήσεις του χρήστη σε σχέση με την ιδιωτικότητα των προσωπικών του δεδομένων, δηλαδή ποιες από τις υπηρεσίες διαδικτύου θα έχουν πρόσβαση και σε ποια από αυτά.
Επιπρόσθετα, η παρούσα ατομική διπλωματική εργασία στοχεύει στην επέκταση της υπάρχουσας αρχιτεκτονικής γύρω από τις υπηρεσίες διαδικτύου έτσι ώστε να λαμβάνονται υπόψη οι προτιμήσεις του χρήστη για την ιδιωτικότητα του, χωρίς όμως να είναι απαραίτητο να τροποποιηθεί η υλοποίηση των υπηρεσιών διαδικτύου.
Όπως προανέφερα στην παρούσα ατομική διπλωματική εργασία υλοποιήθηκε ένα σύνολο σεναρίων τα οποία λαμβάνουν υπόψη τις προτιμήσεις του χρήστη για την διαχείριση των προσωπικών του δεδομένων, χωρίς όμως να χρειάζεται να αλλαχθεί η δομή των υπηρεσιών διαδικτύου. Για την απόκρυψη των προσωπικών δεδομένων του χρήστη χρειάστηκε να εισαχθούν περιορισμοί για το ποιες πληροφορίες θα είναι προσβάσιμες από ποιες υπηρεσίες διαδικτύου, οι οποίοι θα εξηγηθούν με λεπτομέρεια σε επόμενο κεφάλαιο.

1.3 Τεχνολογίες Διαδικτύου

1.3.1 XML

Η Extensible Markup Language (XML) είναι μια γλώσσα, η οποία καθορίζει ένα σύνολο από κανόνες για την κωδικοποίηση των εγγράφων σε μορφή που είναι κατανοητή τόσο από τον άνθρωπο όσο και από τις μηχανές [1].

Οι στόχοι του σχεδιασμού της γλώσσας XML επικεντρώνονται στην απλότητα, στη γενικότητα καθώς και στη χρηστικότητα μέσω του Διαδικτύου. Παρόλο που ο σχεδιασμός της γλώσσας XML αρχικά είχε επικεντρωθεί σε έγγραφα, η γλώσσα αυτή χρησιμοποιείται ευρέως για την αναπαράσταση των διαφόρων δεδομένων, για παράδειγμα σε υπηρεσίες διαδικτύου [1].

Επιπρόσθετα, η γλώσσα XML αποτελεί ένα ευέλικτο τρόπο για να δημιουργηθούν μορφοποιήσεις και ώστε να είναι δυνατόν να παρέχεται τόσο η μορφή όσο και τα στοιχεία στο διαδίκτυο.

Η XML, σύμφωνα με το W3C, είναι παρόμοια με τη γλώσσα HTML η οποία αποτελεί την γλώσσα των ιστοσελίδων σήμερα. Η XML και HTML είναι γλώσσες οι οποίες περιέχουν σύμβολα για να περιγράψουν το περιεχόμενο μιας σελίδας ή αρχείου. Η γλώσσα XML χρησιμοποιείται για να περιγράψει το περιεχόμενο, δηλαδή το τι δεδομένα περιγράφονται. Επίσης, ένα αρχείο XML μπορεί να επεξεργαστεί μόνο ως δεδομένο από ένα πρόγραμμα.

Η XML γλώσσα είναι "επεκτάσιμη", επειδή περιέχει απεριόριστα σύμβολα. Στην πραγματικότητα η γλώσσα XML αποτελεί ένα απλούστερο και ευκολότερο στη χρήση του υποσύνολο της Standard Generalized Markup Language (SGML) [2].

1.3.2 XML Schema

Ένα XML schema αποτελεί μια περιγραφή του τύπου του XML εγγράφου. Συνήθως περιέχει τους περιορισμούς σχετικά με τη δομή και το περιεχόμενο των εγγράφων του συγκεκριμένου τύπου, πέρα από τους βασικούς συντακτικούς περιορισμούς που καθορίζονται από την ίδια την XML. Γενικά οι συγκεκριμένοι περιορισμοί εκφράζονται με κάποιο συνδυασμό των γραμματικών κανόνων που διέπουν τη σειρά των στοιχείων, οι τύποι δεδομένων που διέπουν το περιεχόμενο των στοιχείων και ιδιοχαρακτηριστικών τους, καθώς επίσης και πιο εξειδικευμένους κανόνες, όπως είναι η μοναδικότητα των περιορισμών ακεραιότητας των αναφορών [3] .

Κύριοι στόχοι του XML schema αποτελεί αρχικά το να μπορέσουν να εκφραστούν μέσα στο πρότυπο αρχές αντικειμενοστραφούς σχεδιασμού οι οποίες μπορούν να βρεθούν σε όλες τις αντικειμενοστραφείς γλώσσες προγραμματισμού. Επίσης να παρέχεται υποστήριξη για σύνθετους τύπους δεδομένων παρόμοια με την υποστήριξη που υπάρχει στις περισσότερες σχεσιακές βάσεις δεδομένων [4] .

1.3.3 HTTP

Το Hypertext Transfer Protocol (HTTP) είναι ένα πρωτόκολλο το οποίο χρησιμοποιείται από τα προγράμματα περιήγησης του διαδικτύου για να τους δίνει δυνατότητα πρόσβασης σε διάφορες σελίδες. Το HTTP είναι ένα client/server πρωτόκολλο το οποίο χρησιμοποιείται για να μεταφέρει μηνύματα μεταξύ του διακομιστή και του πελάτη σε περιβάλλοντα υπηρεσιών διαδικτύου [5] .

Οι υπηρεσίες διαδικτύου χρησιμοποιούν δύο από τα είδη των μηνυμάτων HTTP το GET και το POST. Το μήνυμα GET χρησιμοποιείται για να αναζητήσει και να επιστρέψει ένα πόρο ο οποίος καθορίζεται από το URI, δηλαδή εάν χρειαζόμαστε να κατεβάσουμε αρχεία από τους διακομιστές. Το μήνυμα POST χρησιμοποιείται για την ενημέρωση ενός πόρου ή την παροχή εισροής σε κάποιο πρόγραμμα επεξεργασίας στον εξυπηρετητή, δηλαδή χρησιμοποιείται από τον πελάτη όταν θέλουμε να τρέξουμε συγκεκριμένο κώδικα εφαρμογής στον εξυπηρετητή.

1.4 Υπηρεσίες Διαδικτύου

1.4.1 Υπηρεσίες Διαδικτύου (Web Services) και SOA

Μία Service Oriented Architecture (SOA) είναι ένα σύνολο αρχών και μεθοδολογιών με σκοπό τον σχεδιασμό και την ανάπτυξη λογισμικού, το οποίο έχει την μορφή των διαλειτουργικών υπηρεσιών. Οι υπηρεσίες αυτές είναι αυστηρά καθορισμένες λειτουργίες που έχουν κατασκευαστεί ως συστατικά στοιχεία του λογισμικού. Οι υπηρεσίες αυτές μπορεί να είναι διακριτά κομμάτια κώδικα ή ακόμα και δομές δεδομένων, τα οποία έχουν την δυνατότητα να επαναχρησιμοποιηθούν για διαφορετικούς σκοπούς. Οι αρχές σχεδιασμού των SOA χρησιμοποιούνται και κατά τη διάρκεια των φάσεων της ανάπτυξης συστημάτων. Η SOA παρέχει έναν τρόπο για τους καταναλωτές των υπηρεσιών, όπως είναι οι εφαρμογές διαδικτύου, να έχουν επίγνωση των διαθέσιμων SOA-based υπηρεσιών [5].

Μία υπηρεσία διαδικτύου σύμφωνα με τον ορισμό που δίνεται από το W3C (World Wide Web Consortium) είναι ένα σύστημα το οποίο σχεδιάστηκε για να υποστηρίζει διαλειτουργική, μηχανή προς μηχανή, αλληλεπίδραση μέσω δικτύου. Άλλα συστήματα μπορούν να αλληλεπιδρούν με την υπηρεσία διαδικτύου κάνοντας χρήση SOAP μηνυμάτων, της γλώσσας UDDI ή της γλώσσας WSDL, τα οποία καθορίζουν στην περιγραφή τους τον τρόπο της αλληλεπίδρασης. Τα SOAP μηνύματα μεταφέρονται μέσω του πρωτοκόλλου HTTP και χρησιμοποιούν την XML (eXtensible Markup Language) [1].

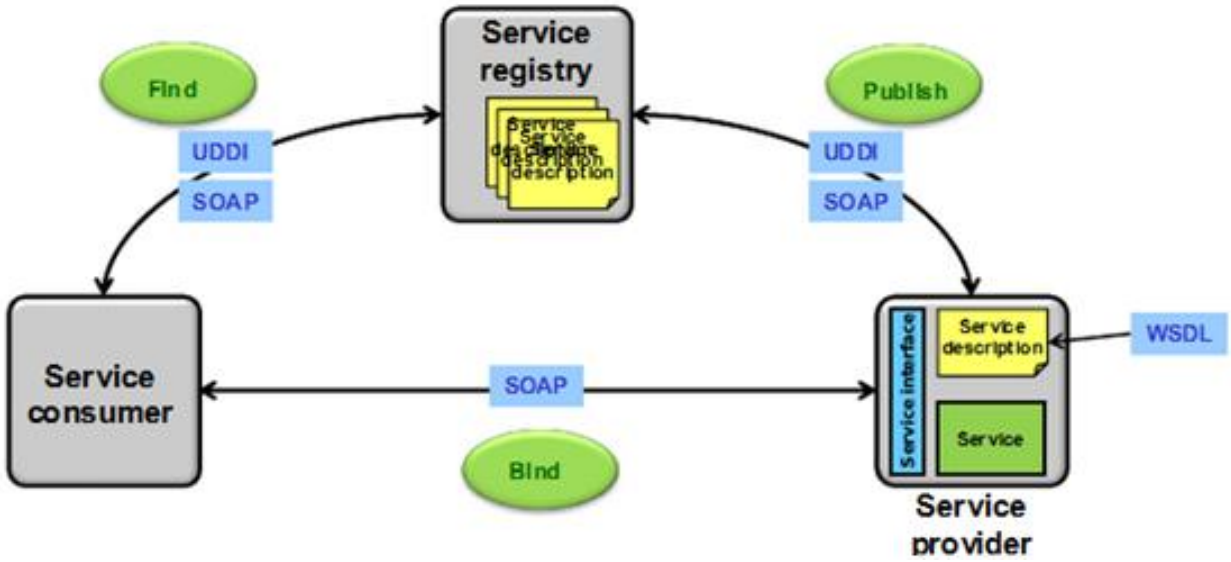
Σύμφωνα με ορισμό ο οποίος δόθηκε από την IBM μπορούμε να πούμε ότι υπηρεσία διαδικτύου είναι μια τεχνολογία που επιτρέπει στις εφαρμογές να επικοινωνούν μεταξύ τους ανεξαρτήτως πλατφόρμας και γλώσσας προγραμματισμού. Μία υπηρεσία διαδικτύου είναι μια διεπαφή λογισμικού, η οποία περιγράφει μια συλλογή από λειτουργίες οι οποίες μπορούν να προσεγγιστούν από το δίκτυο μέσω πρότυπων μηνυμάτων XML. Επιπρόσθετα, χρησιμοποιεί πρότυπα βασισμένα σε γλώσσα XML (eXtensible Markup Language) για να περιγράψει μία λειτουργία προς εκτέλεση, καθώς και τα δεδομένα προς ανταλλαγή με κάποια άλλη εφαρμογή. Μια ομάδα από υπηρεσίες διαδικτύου οι οποίες έχουν τη δυνατότητα να αλληλεπιδρούν μεταξύ τους, καθορίζουν μια εφαρμογή υπηρεσιών διαδικτύου [2].

Σκοπός μίας υπηρεσίας διαδικτύου δεν είναι μόνο να παρέχει μια λειτουργική επικοινωνία για οποιαδήποτε εφαρμογή, αλλά και να την προσφέρει με ενιαίο, ανοικτό, καλά ορισμένο και επεκτάσιμο τρόπο.

Το κυριότερο πρόβλημα το οποίο υπήρχε στις τεχνολογίες ήταν η ισχυρή συνδεσιμότητα (tight-coupling). Μια εφαρμογή η οποία καλούσε μια απομακρυσμένη εφαρμογή έπρεπε να ήταν αυστηρά δεμένη με αυτή μέσω της κλήσης λειτουργίας (function call) την οποία θα εκτελούσε, αλλά και με τις παραμέτρους της. Στα περισσότερα συστήματα πριν από τις υπηρεσίες διαδικτύου, ο μοναδικός τρόπος επικοινωνίας ήταν μια σταθερή διεπαφή με περιορισμένη ευελεξία και προσαρμοστικότητα στα διάφορα περιβάλλοντα ή τις ανάγκες του ίδιου του χρήστη [3].

Οι υπηρεσίες διαδικτύου χρησιμοποιούν τη γλώσσα XML, η οποία μπορεί να περιγράψει οποιαδήποτε δεδομένα με τρόπο ο οποίος είναι ανεξάρτητος από την πλατφόρμα στην οποία τρέχουν, για ανταλλαγή των δεδομένων αυτών μεταξύ των διαφόρων συστημάτων. Με τη χρήση των υπηρεσιών διαδικτύου μπορούμε να έχουμε εφαρμογές με χαλαρή συνδεσιμότητα (loosely-coupled). Επιπλέον, οι υπηρεσίες διαδικτύου μπορούν δυναμικά να επαναξιολογήσουν, να τροποποιήσουν ή να χειριστούν τύπους δεδομένων [4].

1.4.2 Σύνδεση SOAP, UDDI και WSDL



Σχήμα 1.1

Στο Σχήμα 1.1 βλέπουμε την σύνδεση του SOAP, UDDI και WSDL [27].

Αρχικά ένας τελικός χρήστης υπηρεσίας (Service Consumer) για να βρει την υπηρεσία διαδικτύου την οποία ψάχνει χρησιμοποιείται το μέρος του SOAP μηνύματος, που αποστέλλει, το οποίο είναι γραμμένο σε UDDI. Ουσιαστικά παίρνει το μέρος του SOAP μηνύματος που είναι γραμμένο σε UDDI και το ελέγχει με το Service Description σε WSDL για να βρει την κατάλληλη υπηρεσία [27].

Ακολούθως για να δημοσιευθεί μια υπηρεσία διαδικτύου στον παροχέα υπηρεσίας (Service Provider) χρησιμοποιεί και πάλι το μέρος του SOAP μηνύματος, που είναι γραμμένο σε UDDI, το οποίο αποστέλλεται. Επιπρόσθετα, για να δημοσιευτεί μία υπηρεσία διαδικτύου χρειάζεται και μία περιγραφή της (service description), η οποία είναι γραμμένη σε WSDL γλώσσα [27].

Τέλος, για να ενωθεί ο Service Consumer με τον Service Provider χρειάζεται να αποστείλει SOAP request μήνυμα [27].

1.4.3 SOAP

Το Simple Object Access Protocol (SOAP) αποτελεί ένα πρωτόκολλο για την ανταλλαγή δομημένων πληροφοριών στη διαμόρφωση των υπηρεσιών διαδικτύου αλλά και για την κλήση απομακρυσμένων διαδικασιών (RPCs). Το SOAP βασίζεται στην XML για τη μορφή των μηνυμάτων και στο HTTP και SMTP για τη μετάδοση των μηνυμάτων αυτών.

Το SOAP παρέχει τρία βασικά χαρακτηριστικά που είναι η επεκτασιμότητα, η ουδετερότητα και η ανεξαρτησία [6]. Αρχικά το SOAP είναι πρωτόκολλο το οποίο χαρακτηρίζεται από την επεκτασιμότητα του, η οποία αποτελεί απαίτηση για ένα πρωτόκολλο μεταφοράς για υπηρεσίες διαδικτύου. Επιπρόσθετα καθώς το SOAP αποτελεί επέκταση της XML παρέχεται η δυνατότητα όταν εκτελεστούν απομακρυσμένες κλήσεις να εξαλείφει την απαίτηση ότι και τα δύο συστήματα πρέπει να λειτουργούν με την ίδια πλατφόρμα ή να συντάσσουν στην ίδια γλώσσα προγραμματισμού.

Συνοψίζοντας, το πρωτόκολλο SOAP είναι ένα πρωτόκολλο, βασισμένο στην XML, το οποίο χρησιμοποιείται για την αποστολή μηνυμάτων και για την εγκαθίδρυση απομακρυσμένων διαδικασιών. Χρησιμοποιώντας το SOAP, τα δεδομένα μπορούν να αποστέλλονται χωρίς να λαμβάνεται υπόψη το πρωτόκολλο μεταφοράς.

1.4.4 UDDI

Το Universal Description, Discovery and Integration (UDDI) είναι μία γλώσσα βασισμένη στην XML, η οποία όμως είναι ανεξάρτητη από την πλατφόρμα στην οποία τρέχει και χρησιμοποιείται κυρίως από επιχειρήσεις σε όλο τον κόσμο [7]. Το UDDI βασίζεται σε ένα σύνολο από βιομηχανικά πρότυπα, τα οποία αποτελούνται από το HTTP, την XML, το XML Schema και το SOAP. Με τον τρόπο αυτό το UDDI παρέχει μία διαλειτουργική, θεμελιώδη υποδομή για ένα περιβάλλον λογισμικού το οποίο είναι προσανατολισμένο στις υπηρεσίες.

Επιπρόσθετα, το UDDI επικεντρώνεται στην εύρεση υπηρεσιών διαδικτύου οι οποίες υποστηρίζουν την περιγραφή και την ανακάλυψη των εταιριών/οργανισμών, των παροχέων των υπηρεσιών διαδικτύου, των υπηρεσιών διαδικτύου οι οποίες είναι διαθέσιμες και τέλος των τεχνικών διεπαφών οι οποίες μπορούν να χρησιμοποιηθούν.

Τέλος, το UDDI μπορεί να χαρακτηριστεί σαν χώρος αποθήκευσης περιγραφών υπηρεσιών διαδικτύου, ο οποίος ορίζει το μοντέλο δεδομένων για την καταχώρηση και την αναζήτηση πληροφοριών μιας επιχείρησης.

1.4.5 WSDL

Η Web Service Definition Language (WSDL) αποτελεί μία επέκταση της γλώσσας XML, η οποία χρησιμοποιείται για να περιγράψουμε την λειτουργικότητα που παρέχεται από μία υπηρεσία διαδικτύου αλλά και για να περιγράψει τη διεπαφή ανάμεσα στο διακομιστή και στο χρήστη. Μια WSDL περιγραφή μίας υπηρεσίας διαδικτύου παράγει μία περιγραφή, η οποία μπορεί να διαβαστεί από τον υπολογιστή και περιέχει λεπτομέρειες όπως το πώς ονομάζεται η υπηρεσία διαδικτύου, το πώς μπορείς να καλέσεις την συγκεκριμένη υπηρεσία διαδικτύου, το τι παραμέτρους χρειάζεται για να κληθεί καθώς επίσης και το τι παραμέτρους επιστρέφει. Η WSDL περιγράφει τις υπηρεσίες διαδικτύου ως σύνολα από τελικά σημεία του δικτύου (end points) και πύλες (ports) [8].

Επιπρόσθετα, ο ορισμός που δίνεται από το W3C είναι ότι η WSDL είναι ένα σχήμα XML για την περιγραφή δικτυακών υπηρεσιών σαν ένα σύνολο από τελικά σημεία που λειτουργούν σε μηνύματα τα οποία περιέχουν πληροφορία είτε προσανατολισμένη στα έγγραφα είτε προσανατολισμένη στις διαδικασίες.

Επίσης, η WSDL χρησιμοποιείται για την περιγραφή των υπηρεσιών διαδικτύου. Χρησιμοποιείται δηλαδή, για να καθορίσει τις λειτουργίες καθώς και τις μεθόδους τις οποίες μπορεί να παρέχει μια υπηρεσία διαδικτύου. Ακόμα, χρησιμοποιείται για τον καθορισμό της τοποθεσίας τις κάθε υπηρεσία διαδικτύου καθώς και του τρόπου με τον οποίο μπορεί να αποκτηθεί πρόσβαση σε αυτή [9].

Τα αρχεία της WSDL για να περιγράψουν μία υπηρεσία διαδικτύου χρησιμοποιούν αρχικά τους τύπους, έπειτα το μήνυμα, τη λειτουργία, τον τύπο της θύρας, την ένωση, τη θύρα και την υπηρεσία.

1.5 Πλαίσιο Χρήσης

1.5.1 Ορισμός Πλαισίου Χρήσης

Ένας ορισμός του πλαισίου χρήσης όπως αυτός δόθηκε από τον Dey και Abowd είναι ότι «Κάθε πληροφορία μπορεί να χρησιμοποιηθεί για να χαρακτηρίσει την κατάσταση μιας οντότητας. Μια οντότητα είναι ένα πρόσωπο, ένα τοπίο ή ένα αντικείμενο το οποίο συσχετίζεται με την αλληλεπίδραση που υπάρχει ανάμεσα σε ένα χρήστη και σε μια εφαρμογή, συμπεριλαμβανομένων του ίδιου του χρήστη και της ίδιας της εφαρμογής " [10].
Η έννοια του πλαισίου χρήσης είναι άρρηκτα δεμένη με τις έννοιες της διαλειτουργικότητας και του user profiling. Επιπρόσθετα, περιλαμβάνει θέματα για το πώς «εξωτερικοί» παράγοντες έχουν τη δυνατότητα να επηρεάσουν το προφίλ του χρήστη, έτσι ώστε να αλλοιώσουν τις προτιμήσεις του χρήστη. Για να καθοριστεί ποιοι είναι αυτοί οι «εξωτερικοί» παράγοντες θα πρέπει αρχικά να διαχωριστεί το πού τελειώνει το πλαίσιο χρήσης και το πού ξεκινούν οι πληροφορίες για το προφίλ του χρήστη. Για παράδειγμα, το πλαίσιο χρήσης μπορεί να περιλαμβάνει την "κατάσταση" του χρήστη, τη θέση του, το χρόνο του, το ρόλο του, κ.λπ. Έτσι μπορούμε να πούμε ότι το πλαίσιο χρήσης επηρεάζει όλους τους τομείς του Μοντέλου Αναφοράς. Επιπλέον, αξίζει να σημειωθεί ότι χωρίς την διαλειτουργικότητα του πλαισίου χρήσης, τα μοντέλα χρήστης δεν μπορούν να είναι διαλειτουργικά [11].

1.5.2 Πλαίσιο Χρήσης και Υπηρεσίες Διαδικτύου

Το πλαίσιο χρήσης μίας Υπηρεσίας Διαδικτύου αναφέρεται στα χαρακτηριστικά τα οποία είτε αφορούν είτε έχουν σχέση με το χρήστη. Για παράδειγμα, μπορεί οι πληροφορίες αυτές να έχουν σχέση με την υπηρεσία που τρέχει, τη συσκευή την οποία χρησιμοποιεί κλπ.
Με βάση τους Dey και Abowd το πλαίσιο χρήσης ορίζεται ως οποιαδήποτε πληροφορία μπορεί να χρησιμοποιηθεί για να χαρακτηρίσει την κατάσταση μιας οντότητας. Μία οντότητα μπορεί να είναι ένα άτομο, ένα μέρος ή ένα αντικείμενο που θεωρείται σχετικό με την αλληλεπίδραση μεταξύ ενός χρήστη και μίας εφαρμογής. Έτσι, μπορούμε να πούμε ότι το πλαίσιο χρήσης ασχολείται με οποιαδήποτε πληροφορία αφορά το χρήστη και την εφαρμογή που προσπαθεί να τρέξει [12].

1.6 Ιδιωτικότητα

1.6.1 Ορισμός Ιδιωτικότητας

Σύμφωνα με τους Warren και Brandeis, ιδιωτικότητα είναι το δικαίωμα του να είναι κανείς μόνος του. Επιπρόσθετα όπως διατύπωσε ο Allen Westin το 1967 ιδιωτικότητα είναι το δικαίωμα κάθε ανθρώπου ή το δικαίωμα μίας ομάδας ανθρώπων να καθορίζουν από μόνοι του, πότε, πώς και σε ποιο βαθμό οι προσωπικές τους πληροφορίες θα γίνονται γνωστές σε τρίτους [14].

Με βάση τον Rosenberg η ιδιωτικότητα χωρίζεται σε τρεις βασικές έννοιες. Αρχικά έχουμε τη χωρική ιδιωτικότητα, η οποία είναι η προστασία του φυσικού χώρου ο οποίος περιβάλει ένα άτομο. Στην συνέχεια, έχουμε την ιδιωτικότητα του ατόμου, η οποία είναι ουσιαστικά η προστασία του ίδιου του ατόμου από τρίτους. Τέλος, έχουμε την ιδιωτικότητα της πληροφορίας που είναι ουσιαστικά το δικαίωμα το οποίο έχει κάθε άτομο να ελέγχει τον τρόπο με τον οποίο θα συλλέγονται, θα αποθηκεύονται, θα επεξεργάζονται τα προσωπικά του στοιχεία από τρίτους. Επίσης, κάθε άτομο μπορεί να ελέγχει κατά πόσο θα συλλέγονται, θα αποθηκεύονται, θα επεξεργάζονται τα προσωπικά του στοιχεία ή όχι [14].

Στη σημερινή κοινωνία όπου ο καθένας μπορεί να βρει ότι χρειάζεται από το διαδίκτυο, δεν μπορούμε να πούμε ότι οι νόμοι της κάθε χώρας είναι ικανοί να καλύψουν την έννοια της ιδιωτικότητας. Έτσι, η ιδιωτικότητα θεωρείται πλέον βασική απαίτηση για κάθε νέο σύστημα που δημιουργείται.

1.6.2 Απαιτήσεις Ιδιωτικότητας

Για να είναι δυνατόν να ορισθεί η ιδιωτικότητα ως βασική απαίτηση για τη δημιουργία κάθε νέου συστήματος χρειάστηκε να ορισθούν κάποιες επιπρόσθετες απαιτήσεις [14].

Αρχικά, έχουμε την αυθεντικοποίηση (authentication) μέσω της οποίας μπορεί ένα σύστημα να επιβεβαιώσει την ταυτότητα κάποιου χρήστη που προσπαθεί να έχει πρόσβαση σε αυτό. [15] Στο διαδίκτυο συνήθως η αυθεντικοποίηση γίνεται με την εισαγωγή κάποιου συνθηματικού (username) και κάποιου κωδικού πρόσβασης(password) [16].

Στη συνέχεια έχουμε την εξουσιοδότηση (authorization), η οποία είναι η διαδικασία μέσω της οποίας μπορείς να δώσεις σε κάποιο χρήστη δικαίωμα πρόσβασής σε συγκεκριμένα δεδομένα ή δικαίωμα εκτέλεσης συγκεκριμένων ενεργειών [17]. Για παράδειγμα, σε μία εταιρεία με υπαλλήλους, οι οποίοι έχουν διαφορετικές θέσεις και συνεπώς διαφορετικά

καθήκοντα, ο πρόεδρος της εταιρείας αναλαμβάνει να εξουσιοδοτήσει κάθε υπάλληλο με τα καθήκοντα που είναι ανάλογα με το ρόλο του και τις αρμοδιότητές του.

Έπειτα έχουμε την αναγνώριση (identification), που αποτελεί την διαδικασία με βάση την οποία δεν επιτρέπεται σε μη εξουσιοδοτημένους χρήστες να έχουν πρόσβαση σε δεδομένα τα οποία είναι φυλαγμένα στο σύστημα. Έτσι είναι δυνατόν να διαφυλάσσεται η ιδιωτικότητα των ατόμων που έχουν τα προσωπικά τους δεδομένα στο σύστημα [14].

Επιπρόσθετα, έχουμε την προστασία των δεδομένων (data protection), που αφορά την επεξεργασία των προσωπικών δεδομένων και την ελεύθερη διακίνησή τους αφού παρθεί η σχετική άδεια από τον κάτοχό τους [14].

Τέλος, έχουμε την ανωνυμία (anonymity) με βάση την οποία ένας χρήστης μπορεί να χρησιμοποιήσει μία υπηρεσία ή ένα σύνολο υπηρεσιών χωρίς να είναι υποχρεωμένος να αποκαλύψει την ταυτότητά του [14].

1.6.3 Προσωπικά Δεδομένα

Τα προσωπικά δεδομένα χωρίζονται σε Δεδομένα Προσωπικού Χαρακτήρα και σε Ευαίσθητα Προσωπικά Δεδομένα.

Με βάση ορισμό ο οποίος καθορίστηκε από την EU Data Protection Directive (95/46/EC), τα Δεδομένα Προσωπικού Χαρακτήρα ορίζονται ως τα δεδομένα τα οποία αναφέρονται σε ένα υπαρκτό άτομο. Δεδομένα τα οποία αποτελούν προσωπικά δεδομένα είναι το όνομα και το επίθετο, η ηλικία, η οικογενειακή κατάσταση κλπ [18].

Τα Ευαίσθητα Προσωπικά Δεδομένα ορίζονται ως τα δεδομένα τα οποία αναφέρονται στη μόρφωση, στην οικονομική κατάσταση, στο ιατρικό ιστορικό, στο ποινικό μητρώο και στο αρχείο εργοδότησης κάθε ατόμου [19]. Επίσης Ευαίσθητα Προσωπικά Δεδομένα θεωρούνται και δεδομένα τα οποία αφορούν πολιτικά και θρησκευτικά πιστεύω, προσωπικές πεποιθήσεις, συμμετοχή σε πολιτικές ενώσεις και πληροφορίες για την ερωτική ζωή του ατόμου [14].

Κεφάλαιο 2

Υφιστάμενη Αρχιτεκτονική Υπηρεσιών Διαδικτύου

2.1 Σύστημα Διαχείρισης Πλαισίου Χρήσης 12
2.1.1 Apache Axis 2 12
2.1.2 Apache CXF 13
2.2 XML APIs 13
2.2.1 SAX 13
2.2.2 DOM 14

2.1 Σύστημα Διαχείρισης Πλαισίου Χρήσης

2.1.1 Apache Axis 2

Το Apache Axis2 είναι ένα από τα πιο διαδεδομένα συστήματα διαχείρισης πλαισίου χρήσης. Το Apache Axis2 παρέχει υπηρεσίες με δύο υλοποιήσεις ApacheAxis2/Java και Apache Axis2/C. [13] Η επικοινωνία μεταξύ των υπηρεσιών διαδικτύου είναι δυνατή με τη βοήθεια της γλώσσας XML και του πρωτοκόλλου SOAP. Το SOAP καθορίζει τη δομή των μηνυμάτων που θα ανταλλάσσονται κατά τη χρήση υπηρεσιών διαδικτύου. Το Apache Axis2 αποτελεί μια SOAP engine, η οποία είναι υπεύθυνη τόσο για τη δημιουργία όσο και για την ερμηνεία των SOAP μηνυμάτων [25].

Το Apache Axis2 υποστηρίζει SOAP 1.1 καθώς και SOAP 1.2. Το ApacheAxis2 είναι πιο αποτελεσματικό αλλά και πιο επεκτάσιμο από την παλαιότερη έκδοση Axis1.x. Το Apache Axis2 προσφέρει πολλά νέα χαρακτηριστικά. Μερικά από αυτά είναι η ταχύτητα, η χαμηλή κατανάλωση μνήμης, AXIOM, ασύγχρονες υπηρεσίες διαδικτύου, υποστήριξη MEP, ευελιξία, σταθερότητα, component-oriented ανάπτυξη, επεκτασιμότητα κλπ [25].

2.1.2 Apache CXF

Το Apache CXF είναι ένα σύστημα διαχείρισης πλαισίου χρήσης με μόνη διαφορά ότι είναι ανοικτού κώδικα. Ξεκίνησε ως ο συνδυασμός των Celtix που αναπτύχθηκε από IONA Technologies και XFire το οποίο αναπτύχθηκε από μια ομάδα στο Codehaus. Τα δύο αυτά projects ενώθηκαν από τους ανθρώπους που εργάζονταν στο Apache Software Foundation. Το όνομα CXF προέρχεται από το συνδυασμό "Celtix" και "XFire" [26].
Τα βασικά στοιχεία σχεδιασμού του CXF περιλαμβάνουν τον ξεκάθαρο διαχωρισμό των front-end, απλότητα στη δημιουργία πελατών και end – points, υψηλή απόδοση με την ελάχιστη υπολογιστική επιβάρυνση, και τη δυνατότητα ενσωμάτωσης στοιχείων υπηρεσιών διαδικτύου [26].

2.2 XML APIs

Το SAX (Simple API για XML) και το DOM (Document Object Model) σχεδιάστηκαν για να επιτρέπουν στους προγραμματιστές να έχουν πρόσβαση στις πληροφορίες τους χωρίς να χρειάζεται να γράψουν ένα parser σε κάποια συγκεκριμένη γλώσσα προγραμματισμού. Με τη διατήρηση των πληροφοριών σε μορφή XML και με τη χρήση είτε του SAX είτε του DOM API, ο προγραμματιστής είναι ελεύθερος να χρησιμοποιήσει ότι parser επιθυμεί.
Επομένως, τόσο το SAX όσο και το DOM, δημιουργήθηκαν για να εξυπηρετούν τον ίδιο σκοπό, ο οποίος είναι να παρέχει στους προγραμματιστές πρόσβαση σε πληροφορίες που είναι αποθηκευμένες σε έγγραφα XML χρησιμοποιώντας οποιαδήποτε γλώσσα προγραμματισμού καθώς και ένα parser για τη συγκεκριμένη γλώσσα. Ωστόσο, οι δύο αυτές τεχνικές χρησιμοποιούν πολύ διαφορετικές προσεγγίσεις για να μπορούν να δίνουν πρόσβαση στις πληροφορίες των XML αρχείων [31].

2.2.1 SAX

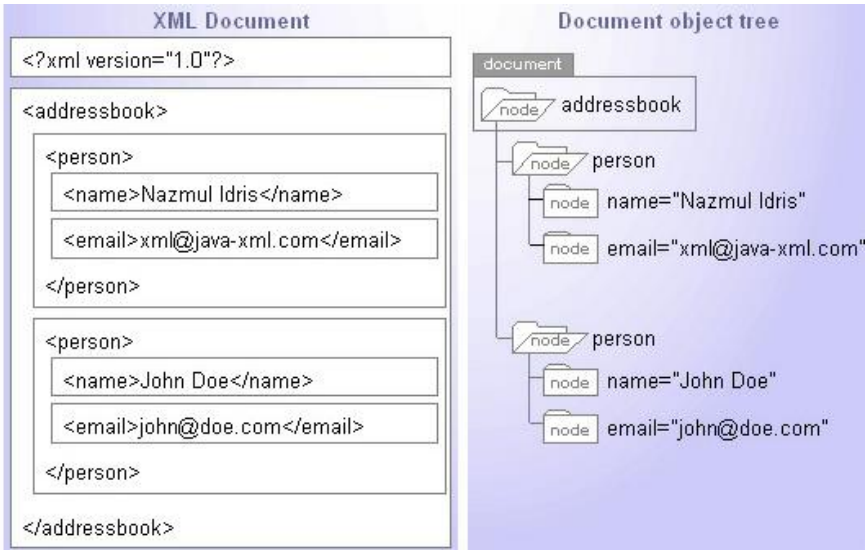
Το SAX (Simple API for XML) παρέχει πρόσβαση στις πληροφορίες που υπάρχουν στα XML έγγραφα ως μια αλληλουχία γεγονότων. Το SAX επιλέγει να μην δημιουργήσει ένα πρότυπο αντικειμένου της Java πάνω από το έγγραφο XML, με αποτέλεσμα το SAX να εργάζεται πολύ πιο γρήγορα [31].

Το SAX έχει όμως συγκεκριμένες απαιτήσεις. Συγκεκριμένα απαιτεί την δημιουργία ενός μοντέλου αντικειμένων το οποίο θα βασίζεται στα γεγονότα του SAX και θα δημιουργεί το μοντέλο αντικειμένου [31].
Το SAX απαιτεί μόνο από τον parser να διαβάσει το έγγραφο XML, και να ενεργοποιεί τα κατάλληλα events ανάλογα με την ετικέτα που υπάρχει στο συγκεκριμένο σημείο του εγγράφου XML [31].

Το SAX χρησιμοποιείται εάν οι πληροφορίες που είναι αποθηκευμένες στο XML έγγραφο είναι σε γλώσσα μηχανής αφού παρέχει στα προγράμματα πρόσβαση στις πληροφορίες αυτές [31].

2.2.2 DOM

Όπως προανέφερα το DOM έχει τη δυνατότητα να δίνει πρόσβαση σε πληροφορίες που είναι αποθηκευμένες σε έγγραφο XML ως ένα ιεραρχικό μοντέλο αντικειμένου. Το DOM δημιουργεί ένα δέντρο από κόμβους και ο προγραμματιστής έχει τη δυνατότητα να έχει πρόσβαση στις πληροφορίες του αρχείου μέσα από αλληλεπίδραση με το δέντρο αυτό [31].



Σχήμα 2.1

Το Σχήμα 2.1 είναι ένα παράδειγμα του πώς η πληροφορία που υπάρχει στο XML ενός εγγράφου μετατρέπεται σε δέντρο από κόμβους χρησιμοποιώντας το DOM. Το DOM δημιουργεί ένα δέντρο κόμβων ανεξάρτητα από το είδος των πληροφοριών που περιέχονται στο XML έγγραφο. Με τον τρόπο αυτό το DOM αναγκάζει τον προγραμματιστή να χρησιμοποιεί το δέντρο για να έχει πρόσβαση στις πληροφορίες [31].

Το DOM χρησιμοποιείται κυρίως εάν τα έγγραφα XML περιέχουν δεδομένα. Επίσης, χρησιμοποιείται σε συστήματα που ασχολούνται με διαχείριση πληροφοριών σε έγγραφα [31].

Κεφάλαιο 3

Σχεδίαση Συστήματος και Υλοποίηση

3.1 Ανάλυση Τεχνολογιών Υλοποίησης Σεναρίου 17
3.1.1 Apache Axis2 17
3.1.2 Eclipse Java EE IDE for Web Developers 17
3.1.3 Apache Tomcat 7 17
3.1.4 DOM 18
3.2 Ανάλυση web services 18
3.2.1 Welcoming Service 18
3.2.2 User Profile Service 18
3.2.3 Current Location Service 19
3.2.4 Buy Service 20
3.2.5 Books Store Service 20
3.2.6 Books List Service 21
3.3 Ανάλυση Parser 22
3.3.1 Επεξήγηση XML Schema (upp.xsd) 22
3.3.2 Επεξήγηση Privacy Preferences 23
3.4 Επεξήγηση Υφιστάμενης Αρχιτεκτονικής Axis 2 Handler 26
3.5 Ανάλυση Axis 2 Handler 27
3.5.1 Ανάλυση Plugins Σεναρίου 27
3.5.1.1 Books List Plugin 27
3.5.1.2 Buying Plugin 28
3.5.1.3 Welcoming Plugin 29
3.5.2 Ανάλυση Providers Σεναρίου 30
3.5.2.1 Books List Provider 30
3.5.2.2 Books Store Provider 30
3.5.2.3 Buy Provider 30
3.5.2.4 Welcome Provider 31

3.1 Ανάλυση Τεχνολογιών Υλοποίησης Σεναρίου

3.1.1 Apache Axis2

Για την υλοποίηση του συνόλου των σεναρίων, ώστε να καταστεί εφικτή η εκπόνηση της ατομικής διπλωματικής μου εργασίας επέλεξα ως σύστημα διαχείρισης πλαισίου χρήσης το Apache Axis2. Αρχικά, επέλεξα το συγκεκριμένο σύστημα διαχείρισης πλαισίου χρήσης επειδή προσφέρεται στην γλώσσα με την οποία υλοποίησα την ατομική διπλωματική μου εργασία που είναι η Java. Επίσης επέλεξα το Apache Axis2 διότι όπως προανέφερα προσφέρει πολλά νέα χαρακτηριστικά σε σχέση με άλλες εκδόσεις. Μερικά από αυτά είναι η ταχύτητα, η χαμηλή κατανάλωση μνήμης, AXIOM, ασύγχρονες υπηρεσίες διαδικτύου, υποστήριξη MEP, ευελιξία, σταθερότητα, component-oriented ανάπτυξη, επεκτασιμότητα κλπ [25].

3.1.2 Eclipse Java EE IDE for Web Developers

Αφού επέλεξα την γλώσσα στην οποία υλοποίησα την διπλωματική μου αποφάσισα ότι θα χρησιμοποιούσα το Eclipse Java EE IDE for Web Developers για την δημιουργία των υπηρεσιών διαδικτύου. Το Eclipse Java EE IDE for Web Developers χρησιμοποιείται για την δημιουργία Java applications. Θεωρείται ως το καλύτερο εργαλείο ανάπτυξης εφαρμογών, αφού παρέχει επεξεργασία της γλώσσας Java με την στοιχειώδη σύνταξη, υποστήριξη Java EE 5, εργαλείο για σύνταξη HTML / JSP / JSF, εργαλεία διαχείρισης βάσεων δεδομένων καθώς και υποστήριξη για τους πιο δημοφιλής application servers [28].

3.1.3 Apache Tomcat 7

Για την υλοποίηση των υπηρεσιών διαδικτύου χρειάστηκα και έναν εξυπηρετητή στον οποίο θα δημοσιεύονταν. Ο Apache Tomcat είναι ένας εξυπηρετητής δικτύου ανοικτού λογισμικού και ο servlet container που αναπτύχθηκε από τον οργανισμό Apache Software Foundation (ASF). Ο Tomcat υλοποιεί τις προδιαγραφές του Java Servlet και των JavaServer Pages (JSP) από την Oracle Corporation και παρέχει ένα ξεκάθαρο Java HTTP web server περιβάλλον στο οποίο μπορεί να τρέξει κώδικας Java [29].

3.1.4 DOM

Το Document Object Model (DOM), είναι μια προδιαγραφή διεπαφής προγραμματισμού η οποία αναπτύχθηκε από το World Wide Web Consortium (W3C) και επιτρέπει σε έναν προγραμματιστή να δημιουργήσει και να τροποποιήσει σελίδες HTML αλλά και XML έγγραφα ως αντικείμενα του προγράμματος. Μέχρι τώρα η HTML (Hypertext Markup Language) και XML (Extensible Markup Language) είναι τρόποι για να εκφραστεί ένα έγγραφο σε όρους των δομών δεδομένων. Ως αντικείμενα του προγράμματος, τα έγγραφα αυτά θα είναι σε θέση να έχουν τόσο το περιεχόμενο τους όσο και τα δεδομένα τους κρυμμένα μέσα στο αντικείμενο, εξασφαλίζοντας έτσι τον έλεγχο για το ποιος έχει δικαίωμα να χειριστεί το συγκεκριμένο έγγραφο. Ως αντικείμενα, τα έγγραφα έχουν την δυνατότητα να μεταφέρουν μαζί τους αντικειμενοστραφή διαδικασίες, οι οποίες ονομάζονται μέθοδοι [30].
Για να διαβάζω τις προτιμήσεις των χρηστών μου από τα έγγραφα XML αποφάσισα να χρησιμοποιήσω το DOM διότι μου παρέχει πιο εύκολη πρόσβαση στα αντικείμενα που περιέχονται μέσα στο έγγραφό μου από ότι το SAX.

3.2 Ανάλυση web services

3.2.1 Welcoming Service

Η υπηρεσία διαδικτύου Welcoming Service είναι υπεύθυνη να καλωσορίζει τον χρήστη στο διαδικτυακό βιβλιοπωλείο. Η συγκεκριμένη υπηρεσία παίρνει σαν παράμετρο το όνομα, το επίθετο και την γλώσσα του χρήστη και του εμφανίζει ένα μήνυμα καλωσορίσματος.

3.2.2 User Profile Service

Η υπηρεσία διαδικτύου User Profile Service έχει πρόσβαση στα προσωπικά στοιχεία για κάθε ένα από τους χρήστες του διαδικτυακού βιβλιοπωλείου.
Η συγκεκριμένη υπηρεσία περιέχει οκτώ μεθόδους. Αρχικά, η μέθοδος getUserName η οποία παίρνει σαν παράμετρο ένα userId. Η μέθοδος αυτή είναι υπεύθυνη να επιστρέψει το όνομα του χρήστη με το συγκεκριμένο userId.
Έπειτα, η μέθοδος getUserSurname που έχει σαν παράμετρο ένα userId. Η συγκεκριμένη μέθοδος είναι υπεύθυνη να επιστρέψει το επίθετο του χρήστη με το συγκεκριμένο userId.

Επίσης, η μέθοδος getUserAge δέχεται σαν παράμετρο ένα userId. Η συγκεκριμένη μέθοδος επιστρέφει την ηλικία του χρήστη με το συγκεκριμένο userId.
Στην συνέχεια ακολουθεί η μέθοδος getUserPaymentMethod που δέχεται σαν παράμετρο ένα userId. Η μέθοδος αυτή, είναι υπεύθυνη να επιστρέψει τον προτεινόμενο τρόπο πληρωμής του χρήστη με το συγκεκριμένο userId.
Ακολούθως η μέθοδος getUserCardType η οποία παίρνει σαν παράμετρο ένα userId. Η μέθοδος αυτή είναι υπεύθυνη να επιστρέψει τον τύπο της πιστωτικής κάρτας του χρήστη με το συγκεκριμένο userId.
Επιπρόσθετα η μέθοδος getUserCardNumber, η οποία έχει για παράμετρο ένα userId. Η μέθοδος getUserCardNumber είναι υπεύθυνη να επιστρέψει τον μοναδικό αριθμό της πιστωτικής κάρτας του χρήστη με το συγκεκριμένο userId.
Στην συνέχεια η μέθοδος getUserCardExpMonth η οποία δέχεται σαν παράμετρο ένα userId, είναι υπεύθυνη να επιστρέψει τον μήνα λήξης της πιστωτικής κάρτας του χρήστη με το συγκεκριμένο userId.
Τέλος, υπάρχει η μέθοδος getUserCardExpYear η οποία παίρνει σαν παράμετρο ένα userId και επιστρέφει την χρονιά την οποία λήγει η πιστωτική κάρτα του χρήστη με το συγκεκριμένο userId.

3.2.3 Current Location Service

Η υπηρεσία διαδικτύου Current Location Service η οποία είναι υπεύθυνη για την χώρα ή την πόλη στην οποία βρίσκεται ο συγκεκριμένος χρήστης καθώς και για την τοπική γλώσσα. Η υπηρεσία αυτή περιέχει τρεις μεθόδους.
Αρχικά, η μέθοδος getCurrentCountry που δέχεται σαν παράμετρο το userId ενός χρήστη και επιστρέφει τυχαία μία χώρα, από τον κατάλογο με τις χώρες, η οποία αντιστοιχεί στην χώρα στην οποία βρίσκεται την δεδομένη στιγμή ο συγκεκριμένος χρήστης.
Στην συνέχεια, η μέθοδο getCurrentCity η οποία δέχεται σαν παράμετρο το userId ενός χρήστη και επιστρέφει τυχαία μία πόλη, από ένα κατάλογο με διάφορες πόλεις, η οποία αντιστοιχεί στην πόλη στην οποία βρίσκεται την δεδομένη στιγμή ο συγκεκριμένος χρήστης.
Τέλος, η μέθοδο getNativeLanguage έχει σαν παράμετρο το userId ενός χρήστη και επιστρέφει τυχαία μία γλώσσα, από ένα κατάλογο που περιέχει διαφορετικές γλώσσες, η οποία αποτελεί την τοπική γλώσσα στην περιοχή όπου βρίσκεται την δεδομένη στιγμή ο συγκεκριμένος χρήστης.

3.2.4 Buy Service

Η υπηρεσία διαδικτύου Buy Service, ασχολείται με την πληρωμή και τον τρόπο πληρωμής για κάθε βιβλίο. Η υπηρεσία αυτή απαρτίζεται από δύο μεθόδους, οι οποίες αντιστοιχούν στις δύο διαφορετικές μεθόδους πληρωμής.

Αρχικά, υπάρχει η μέθοδος cashPayment, η οποία αντιστοιχεί στην πληρωμή με μετρητά. Η μέθοδος αυτή δέχεται σαν παραμέτρους το bookId, το ποσό της πληρωμής καθώς και ένα string με τις πληροφορίες των βιβλιοπωλείων από όπου ο χρήστης μπορεί να προμηθευτεί το βιβλίο του. Η μέθοδος αυτή εμφανίζει ένα μήνυμα / απόδειξη στον χρήστη και το/τα βιβλιοπωλείο/α από όπου μπορεί να παραλάβει το βιβλίο του.

Επιπρόσθετα, η μέθοδος creditCardPayment αντιπροσωπεύει την πληρωμή με πιστωτική κάρτα. Η συγκεκριμένη μέθοδος δέχεται επτά παραμέτρους, οι οποίες είναι το bookId, ο αριθμός της πιστωτικής κάρτας, ο μήνας λήξης της πιστωτικής κάρτας, ο χρόνος λήξης της πιστωτικής κάρτας, το ποσό της πληρωμής καθώς και ένα string με τις πληροφορίες των βιβλιοπωλείων από όπου ο χρήστης μπορεί να προμηθευτεί το βιβλίο του. Η μέθοδος creditCardPayment, ελέγχει τον μήνα και την χρονιά λήξης της κάρτας με την σημερινή ημερομηνία. Εάν η κάρτα είναι έγκυρη, τότε εμφανίζει στον χρήστη το σχετικό μήνυμα / απόδειξη και ακολούθως από πού μπορεί να παραλάβει το βιβλίο του. Εάν όμως η πιστωτική κάρτα είναι άκυρη, τότε εμφανίζει στον χρήστη μήνυμα λάθους με την ένδειξη ότι έχει λήξει η πιστωτική κάρτα την οποί έχει καταχωρίσει και γι’ αυτό δεν μπορεί να ολοκληρωθεί η πληρωμή.

3.2.5 Books Store Service

Η υπηρεσία διαδικτύου Books Store Service περιέχει πληροφορίες σχετικά με όλα τα βιβλιοπωλεία τα οποία συνεργάζεται το Online Book Store και έχει την δυνατότητα να ενημερώνει τον χρήστη από πού μπορεί να παραλάβει την παραγγελία του. Η συγκεκριμένη υπηρεσία απαρτίζεται τρεις μεθόδους.

Αρχικά, η μέθοδος getBookStoreById είναι υπεύθυνη να επιστρέψει τα στοιχεία ενός συγκεκριμένου βιβλιοπωλείου. Η μέθοδος αυτή λαμβάνει σαν παράμετρο ένα bookStoreId και επιστρέφει την εγγραφή για το βιβλιοπωλείο που έχει τον συγκεκριμένο κωδικό.

Έπειτα, η μέθοδος getBookStoreByCountry επιστρέφει μία λίστα με τα βιβλιοπωλεία τα οποία υπάρχουν σε κάποια συγκεκριμένη χώρα. Η συγκεκριμένη μέθοδος παίρνει σαν

παράμετρο το όνομα της χώρας που θέλουμε να ελέγξουμε και επιστρέφει την λίστα με τα βιβλιοπωλεία. Εάν δεν υπάρχουν βιβλιοπωλεία στην συγκεκριμένη πόλη, τότε η μέθοδος επιστρέφει μία λίστα με όλα τα βιβλιοπωλεία. Σε αντίθετη περίπτωση επιστρέφει μία λίστα με τα βιβλιοπωλεία τα οποία βρίσκονται στην συγκεκριμένη χώρα.

Στην συνέχεια, η μέθοδος getBookStoreByCity δημιουργεί μία λίστα με τα βιβλιοπωλεία τα οποία υπάρχουν στην συγκεκριμένη πόλη. Η μέθοδος getBookStoreByCity δέχεται σαν παράμετρο το όνομα της πόλης. Εάν δεν υπάρχουν βιβλιοπωλεία στην πόλη αυτή, τότε η μέθοδος επιστρέφει μία λίστα με όλα τα διαθέσιμα βιβλιοπωλεία. Αλλιώς επιστρέφει μία λίστα με τα βιβλιοπωλεία τα οποία βρίσκονται στην πόλη που δόθηκε σαν παράμετρος.

3.2.6 Books List Service

Τέλος, η υπηρεσία διαδικτύου Books List Service, διαχειρίζεται τα στοιχεία όλων των βιβλίων που είναι διαθέσιμα στο διαδικτυακό μας βιβλιοπωλείο. Μέσω της συγκεκριμένης υπηρεσίας μπορεί κάθε χρήστης να διαχωρίσει τα βιβλία ανάλογα με την ηλικία του και έπειτα ανάλογα με κάποιο κωδικό του βιβλίου που του αρέσει. Επιπρόσθετα μπορεί κάθε χρήστης να δει μία λίστα με όσα βιβλία ανήκουν σε συγκεκριμένες κατηγορίες οι οποίες του αρέσουν όπως πχ επιστημονική φαντασία, δράμα, μυστήριο κλπ. Ο χρήστης έχει επίσης την δυνατότητα να δει μια λίστα με βιβλία με βάση την γλώσσα του. Επιπρόσθετα ο χρήστης έχει την δυνατότητα να δει μία λίστα με βιβλία, το κόστος των οποίων θα κυμαίνεται στα πλαίσια που θα καθορίσει ο ίδιος. Ακόμη, ο χρήστης μπορεί να δει όλα τα βιβλία τα οποία έχουν εκδοθεί μία συγκεκριμένη χρονιά. Τέλος, μπορεί ο χρήστης να δει όσα βιβλία έχουν γραφτεί από ένα συγκεκριμένο συγγραφέα.

Η υπηρεσία διαδικτύου Books List Service απαρτίζεται από 8 μεθόδους.

Αρχικά, η μέθοδος getBookById η οποία είναι υπεύθυνη να επιστρέφει τα στοιχεία του βιβλίου το οποίο έχει ένα συγκεκριμένο bookId. Η μέθοδος δέχεται σαν παράμετρο το Id του βιβλίου που ψάχνουμε και επιστρέφει μία εγγραφή με τα στοιχεία του βιβλίου.

Ακολούθως, η μέθοδο getBookPrice επιστρέφει την τιμή ενός συγκεκριμένου βιβλίου. Η μέθοδος αυτή παίρνει σαν παράμετρο ένα μοναδικό bookId και επιστρέφει την τιμή του βιβλίου με τον εν λόγω κωδικό.

Έπειτα, η μέθοδο getBookByAllowedAge που επιστρέφει μία λίστα με όσα βιβλία αντιστοιχούν στην ηλικία του χρήστη. Η συγκεκριμένη μέθοδος δέχεται σαν παράμετρο την ηλικία του χρήστη και επιστρέφει μία λίστα με τα βιβλία τα οποία επιλέχθηκαν ως κατάλληλα για την συγκεκριμένη ηλικία.

Ακολούθως, υπάρχει η μέθοδος getBookByAuthor, η οποία είναι υπεύθυνη να επιστρέφει μία λίστα με όσα βιβλία έχουν γραφτεί από ένα συγκεκριμένο συγγραφέα τον οποία έχει επιλέξει ο χρήστης. Η μέθοδος getBookByAuthor δέχεται σαν παράμετρο το όνομα του συγγραφέα που έχει επιλέξει ο χρήστης και επιστρέφει μία λίστα με τα στοιχεία των βιβλίων τα οποία έχουν συγγραφεί από τον συγκεκριμένο συγγραφέα / δημιουργό.
Στην συνέχεια, ακολουθεί η μέθοδος getBookByYear, η οποία επιστρέφει μία λίστα με όσα βιβλία έχουν εκδοθεί την χρονιά την οποία επέλεξε ο χρήστης. Η μέθοδος αυτή δέχεται σαν παράμετρο την χρονιά που επιθυμεί ο χρήστης και επιστρέφει μία λίστα με τα στοιχεία των βιβλίων τα οποία εκδόθηκαν την εν λόγω χρονιά.
Επιπρόσθετα, η μέθοδος getBookByCategory επιστρέφει μία λίστα με όσα βιβλία ανήκουν σε μία συγκεκριμένη θεματική κατηγορία την οποία επέλεξε ο χρήστης. Η μέθοδος αυτή δέχεται σαν παράμετρο την κατηγορία που επιθυμεί ο χρήστης και επιστρέφει μία λίστα με τα στοιχεία των βιβλίων τα οποία ανήκουν στην κατηγορία η οποία έχει επιλεχθεί από τον χρήστη.
Επίσης, η getBookByLanguage μέθοδος χρησιμοποιείται για να δημιουργεί μία λίστα με όσα βιβλία είναι γραμμένα σε κάποια γλώσσα την οποία έχει επιλέξει ο χρήστης. Η συγκεκριμένη μέθοδος λαμβάνει σαν παράμετρο κάποια γλώσσα και επιστρέφει μία λίστα με τα στοιχεία των βιβλίων που είναι γραμμένα στην γλώσσα αυτή.
Τέλος, η μέθοδος getBookByPrice, η οποία αναπτύσσει μία λίστα με όσα βιβλία βρίσκονται μεταξύ δύο συγκεκριμένων χρηματικών ποσών. Η μέθοδος αυτή δέχεται σαν παράμετρο μία minPrice και μία maxPrice. Στην συνέχεια, ελέγχει για κάθε βιβλίο εαν η τιμή του ανήκει μεταξύ των 2 τιμών που δόθηκαν σαν παράμετροι και επιστρέφει την λίστα που δημιουργήθηκε με τα στοιχεία των βιβλίων πληρούν της συγκεκριμένες προϋποθέσεις.

3.3 Ανάλυση Parser

3.3.1 Επεξήγηση XML Schema (upp.xsd)

Το αρχείο upp.xsd είναι ένα XML Schema, το οποίο αποτελεί μια περιγραφή του τύπου του XML εγγράφου που περιέχει τις προτιμήσεις ιδιωτικότητας των χρηστών. Το XML Schema περιέχει τους περιορισμούς σχετικά με τη δομή και το περιεχόμενο των XML εγγράφων.
Στο Παράρτημα Α βρίσκεται ο κώδικας του αρχείο upp.xsd. Είναι εμφανές ότι τα δύο κυρίαρχα στοιχεία του XML Schema είναι το allow και το block. Δηλαδή σε ποιες πληροφορίες μπορείς να έχεις πρόσβαση και σε ποιες κάτι τέτοιο δεν είναι δυνατόν.

Επίσης, υπάρχει το dataEnd, το οποίο περιέχει το πεδίο DataCategory και μπορεί να μας διαχωρίσει εάν ένα στοιχείο ανήκει στα Personal Info ή στα Personal Preferences.
Υπάρχει ακόμα και η επιλογή Location στο πεδίο DataCategory, η οποία ασχολείται με περισσότερους περιορισμούς.
Αρχικά, υπάρχει το timePeriodIn με τα πεδία startWeekDay και endWeekDay, όπου εάν για παράδειγμα βρίσκεται στο allow για να σου επιτρέψει πρόσβαση στις υπηρεσίες διαδικτύου του συγκεκριμένου σεναρίου θα πρέπει να βρίσκεσαι σε κάποια μέρα μεταξύ startWeekDay και endWeekDay. Ενώ αντίστοιχα, εάν βρίσκεσαι στο block για να σου επιτραπεί η πρόσβαση στις υπηρεσίες διαδικτύου που απαρτίζουν το συγκεκριμένο σενάριο θα πρέπει να βρίσκεσαι σε κάποια μέρα της εβδομάδας η οποία να μην βρίσκεται μεταξύ startWeekDay και endWeekDay. Επιπρόσθετα, υπάρχει και ο περιορισμός της ώρας με τα πεδία startTime και endTime. Η χρήση τους είναι αντίστοιχη του startWeekDay και endWeekDay.
Επίσης, περιέχει το πεδίο spaceLimitIn το οποίο απαρτίζεται από τα πεδία Country και City. Εάν τα συγκεκριμένα πεδία βρίσκονται στο allow για να σου επιτραπεί η πρόσβαση στις υπηρεσίες διαδικτύου του συγκεκριμένου σεναρίου θα πρέπει να βρίσκεσαι σε στην συγκεκριμένη χώρα και πόλη. Αντίστοιχα εάν βρίσκεται στο block για να σου επιτραπεί η πρόσβαση στις υπηρεσίες διαδικτύου που απαρτίζουν το συγκεκριμένο σενάριο θα πρέπει να μην βρίσκεσαι στην συγκεκριμένη χώρα και πόλη.

3.3.2 Επεξήγηση Privacy Preferences

Το αρχείο Privacy Preferences ουσιαστικά αποτελεί τον parser μας. Με τις μεθόδους του συγκεκριμένου αρχείου μπορούμε να διαβάσουμε τα xml αρχεία τα οποία περιέχουν μέσα τις προτιμήσεις των χρηστών σε σχέση με την ιδιωτικότητα των στοιχείων τους.
Στο συγκεκριμένο αρχείο ορίζονται 10 μεθόδους. Αρχικά, η μέθοδο PrivacyPreferences, που αποτελεί τον κατασκευαστή της κλάσης, παίρνει σαν παράμετρο το userId. Στην μέθοδο αυτή καθορίζεται το path του αρχείου cfg.properties, το οποίο περιέχει το path του φακέλου στον tomcat στον οποίο βρίσκονται τα αρχεία με τις προτιμήσεις των χρηστών. Αφού μεταφερθεί στον συγκεκριμένο φάκελο, η μέθοδος ψάχνει να βρει εάν υπάρχει ένα αρχείο με όνομα userId.xml. Εάν υπάρχει τότε γεμίζει τον πίνακα allowArray[] και blockArray[] με τις προτιμήσεις του χρήστη.
Ακολούθως, η μέθοδος isEnabled είναι υπεύθυνη να επιστρέψει την τιμή της μεταβλητής isEnabled. Η συγκεκριμένη μέθοδος χρησιμοποιείται για να δηλωθεί εάν υπάρχει αρχείο προτιμήσεων ή όχι.

Έπειτα, υπάρχει η μέθοδος isPersonalInfoFieldAllowed, που παίρνει σαν παράμετρο ένα string. Η τιμή που εκλαμβάνεται σαν παράμετρος είναι μία από τις προσωπικές πληροφορίες του χρήστη, που θέλουμε να μάθουμε εάν είναι allowed. Η συγκεκριμένη μέθοδος διατρέχει τον πίνακα allowArray και συγκρίνει της τιμές με αυτή της παραμέτρου. Εάν βρεθεί η παράμετρος στον πίνακα τότε η μέθοδος τυπώνει μήνυμα που ενημερώνει ότι η συγκεκριμένη παράμετρος είναι Allowed αλλιώς τυπώνει μήνυμα που να ενημερώνει ότι δεν είναι Allowed. Επίσης επιστρέφει την αντίστοιχη απάντηση.

Στην συνέχεια, έχει δημιουργηθεί μία παρόμοια μέθοδος με όνομα isPersonalInfoBlocked. Η μέθοδος αυτή είναι υπεύθυνη να διερευνήσει τον πίνακα blockArray για να αποφανθεί εάν η τιμή της παραμέτρου, που ανήκει στα Personal Info, είναι στον πίνακα ή όχι. Η συγκεκριμένη μέθοδος ερευνά ολόκληρο τον πίνακα blockArray συγκρίνοντας τις τιμές του με αυτή της παραμέτρου. Στην περίπτωση που βρίσκεται, η μέθοδος εμφανίζει ένα μήνυμα που ενημερώνει ότι η συγκεκριμένη παράμετρος είναι Blocked, αλλιώς εκτυπώνει μήνυμα που να ενημερώνει ότι δεν είναι Blocked. Η μέθοδος επιστρέφει το αποτέλεσμα της έρευνάς της.

Οι δύο πιο πάνω μέθοδοι κάνουν σχετικά παρόμοια αλλά αντίθετα πράγματα αφού η μία ψάχνει εάν ένα χαρακτηριστικό από τις προσωπικές πληροφορίες ενός χρήστη είναι allowed και η άλλη εάν είναι blocked. Η ύπαρξη και τον δύο είναι απαραίτητη για να μπορούν να προσφέρουν την βεβαίωση ότι κάθε xml αρχείο είναι έγκυρο, δηλαδή ότι περιέχει ένα χαρακτηριστικό από τις προσωπικές πληροφορίες του χρήστη και σαν allowed αλλά και σαν blocked.

Επιπρόσθετα υπάρχει η μέθοδος isPersonalPreferencesFieldAllowed, η οποία δέχεται σαν παράμετρο ένα string. Η τιμή της παραμέτρου είναι μία από τις προσωπικές προτιμήσεις του χρήστη, και επιθυμούμε να μάθουμε εάν είναι allowed. Η μέθοδος αυτή ψάχνει στον πίνακα allowArray. Εάν βρεθεί η τιμή της παραμέτρου στον πίνακα τότε η μέθοδος τυπώνει μήνυμα που ενημερώνει ότι η τιμή αυτή είναι Allowed, αλλιώς τυπώνει μήνυμα που να ενημερώνει ότι δεν είναι Allowed και επιστρέφει την απάντηση.

Η αντίστοιχη μέθοδος της είναι η isPersonalPreferencesBlocked. Η μέθοδος isPersonalPreferencesBlocked προσπαθεί να δει εάν η τιμή, η οποία ανήκει στα Personal Preferences και έχει περαστεί σαν παράμετρος, βρίσκεται στον πίνακα blockArray. Η μέθοδος ψάχνει στον πίνακα blockArray συγκρίνει της τιμές με αυτή της παραμέτρου εμφανίζοντας το κατάλληλο μήνυμα. Επίσης η μέθοδος επιστρέφει την απάντηση στην αναζήτηση της παραμέτρου στον πίνακα.

Όπως έχω επισημάνει και για τις μεθόδους isPersonalInfoBlocked και isPersonalInfoAllowed, έτσι και οι δύο πιο πάνω μέθοδοι εκτελούν αναζητήσεις σε διαφορετικούς πίνακες. Η isPersonalPreferencesFieldAllowed ψάχνει εάν ένα χαρακτηριστικό από τις προσωπικές προτιμήσεις ενός χρήστη είναι allowed και η isPersonalPreferencesBlocked εάν είναι blocked. Η ύπαρξη και τον δύο είναι απαραίτητη για να είναι βέβαιο ότι κάθε xml αρχείο είναι έγκυρο, δηλαδή ότι περιέχει ένα χαρακτηριστικό από τις προσωπικές προτιμήσεις του χρήστη και σαν allowed αλλά και σαν blocked.

Στην συνέχεια, ακολουθεί η μέθοδος isLocationAllowed η οποία δεν παίρνει καμία παράμετρο. Η μέθοδος αυτή διατρέχει τον πίνακα allowArray και βλέπει εάν υπάρχει το χαρακτηριστικό Location μέσα στον πίνακα. Σαν έξοδο τυπώνει το αντίστοιχο μήνυμα, δηλαδή εάν το χαρακτηριστικό Location βρεθεί μέσα στον πίνακα εκτυπώνει ότι Location is allowed, ενώ εάν δεν βρεθεί στον πίνακα τότε εκτυπώνεται το μήνυμα Location is not allowed.

Επιπρόσθετα έχει δημιουργηθεί μία παρόμοια μέθοδος με όνομα isLocationBlocked. Η μέθοδος αυτή επίσης δεν δέχεται παραμέτρους και ψάχνει εάν το χαρακτηριστικό Location βρίσκεται στον πίνακα blockArray. Εάν βρεθεί το χαρακτηριστικό Location στον πίνακα τότε η μέθοδος τυπώνει μήνυμα που ενημερώνει ότι Location is Blocked αλλιώς τυπώνει μήνυμα που να ενημερώνει ότι Location is not Blocked.

Έπειτα, έχει δημιουργηθεί η μέθοδος validaterAllowedLocation, η οποία παίρνει 4 παραμέτρους, weekday, time, country και city. Η μέθοδος είναι υπεύθυνη να βεβαιωθεί ότι εάν το χαρακτηριστικό Location υπάρχει στο αρχείο xml, τότε είναι έγκυρο. Η συγκεκριμένη μέθοδος ελέγχει εάν ο χρήστης, που προσπαθεί να τρέξει κάποιο από τα σενάρια, βρίσκεται στην χώρα και πόλη που πρέπει αλλά και τις ώρες και μέρες που πρέπει. Ακόμη, η συγκεκριμένη μέθοδος εκτυπώνει μηνύματα για το ποιες από τις location conditions είναι valid και ποιες όχι. Τέλος, εμφανίζει το γενικό μήνυμα είτε All Location conditions are valid είτε Could not validate all Location Conditions.

Εν καταλείδι, υπάρχει η μέθοδος getWeekDayNumber η οποία είναι μία βοηθητική συνάρτηση που παίρνει σαν παράμετρο ένα string με την μέρα και σου επιστρέφει τον αριθμό που αντιστοιχεί στην μέρα αυτή.

Οι μέθοδοι αυτοί ενσωματώνονται σε κάθε plugin το οποίο έχει δημιουργηθεί μέσω ενός αντικειμένου τύπου privacypreferences, και έτσι μπορούν να χρησιμοποιηθούν για να γίνονται οι απαραίτητοι έλεγχοι για την ιδιωτικότητα την οποία επιθυμούν οι χρήστες στα δεδομένα τους.

3.4 Επεξήγηση Υφιστάμενης Αρχιτεκτονικής Axis 2 Handler

Η αρχιτεκτονική στην οποία στηρίχθηκε η ανάπτυξη των σεναρίων καθώς και η επέκταση για την ενσωμάτωση της επιλογής για την ιδιωτικότητα των χρηστών στηρίζεται στο σύστημα διαχείρισης πλαισίου χρήσης Apache Axis 2. Η μεθοδολογία η οποία υποστηρίζεται βασίζεται στο SOAP πρωτόκολλο [24].

Ο Axis 2 Handler ο οποίος χρησιμοποιήθηκε για την παρούσα ατομική διπλωματική υποστηρίζει την ενσωμάτωση παραμέτρων (parameter injection), την επιλογή καταλληλότερης μεθόδου (operation selection) και τον κατάλληλο χειρισμό της απάντησης (response manipulation). Στην παρούσα ατομική διπλωματική εργασία χρησιμοποιήθηκαν η ενσωμάτωση παραμέτρων (parameter injection) και η επιλογή καταλληλότερης μεθόδου (operation selection) [24].

Η ενσωμάτωση παραμέτρων (parameter injection) χρησιμοποιείται για να αλλαχθούν μια ή περισσότερες παράμετροι στο request μήνυμα. Οι παράμετροι που αλλάζουν έχουν να κάνουν με το πλαίσιο χρήσης. Για παράδειγμα πληροφορίες οι οποίες αφορούν τον ίδιο το χρήστη αλλάζονται με τις ορθές πληροφορίες οι οποίες δόθηκαν από τον χρήστη [24].

Η επιλογή καταλληλότερης μεθόδου (operation selection) γίνεται από το αρχείο handler.xml το οποίο καθοδηγεί τον Axis 2 Handler για το πιο plugin χρησιμοποιείται σε ποια υπηρεσία διαδικτύου και από ποια μέθοδο [24].

Η εφαρμογή των πιο πάνω λειτουργιών γίνεται κατά την διάρκεια ενός αιτήματος από έναν πελάτη (client). Αρχικά, πραγματοποιείται η «υποκλοπή» του SOAP αιτήματος (SOAP request interception). Συγκεκριμένα, ο Axis 2 Handler «κλέβει» το SOAP αίτημα και με βάση τα περιεχόμενα του αρχείου handler.xml μπορεί να προβεί σε τροποποιήσεις [24].

Ακολουθεί, η φόρτωση των plugins τα οποία σχετίζονται με την υπηρεσία (loading of appropriate plugins), από τον Axis 2 Handler. Ο Axis 2 Handler φροντίζει να αποστείλει το SOAP φάκελο (SOAP envelope), για επεξεργασία, σε κάθε plugin που καλεί [24].

Στην συνέχεια, πραγματοποιείται η τροποποίηση του μηνύματος (Message Modification), κατά την οποία το plugin τροποποιεί το περιεχόμενο του SOAP φακέλου (SOAP envelope) βασισμένο στις πληροφορίες του πλαισίου και τέλος το επιστρέφει πίσω στον Axis 2 Handler. Για να διαφοροποιηθούν οι πληροφορίες οι οποίες αφορούν τον ίδιο τον χρήστη χρειάζεται πρόσβαση στις πληροφορίες του. Η πρόσβαση αυτή επιτυγχάνεται με την ταυτοποίηση ενός μοναδικού πεδίου από τις πληροφορίες του χρήστη [24].

Επόμενο βήμα αποτελεί η «υποκλοπή» της SOAP απάντησης (SOAP response interception). Για το σκοπό αυτό, επαναλαμβάνεται η διαδικασία που ανέλυσα πιο πάνω με μόνη

διαφοροποίηση πως η «υποκλοπή» αυτή τη φορά αφορά την απάντηση και όχι το αίτημα. Αναλυτικότερα, πραγματοποιείται «υποκλοπή» του μηνύματος της απάντησης (message interception), κλήση και φόρτωση των κατάλληλων plugins (loading of appropriate plugins) και τέλος τροποποίηση του μηνύματος SOAP (SOAP message modification) [24].

3.5 Ανάλυση Axis 2 Handler

3.5.1 Ανάλυση Plugins Σεναρίου

Για την ενσωμάτωση των plugins που χρησιμοποιήθηκαν στο σενάριο της παρούσας ατομικής διπλωματικής χρειάστηκε να αλλαχθεί το αρχείο handler.xml έτσι ώστε να εισαχθούν τα plugins και να συσχετιστούν με τα ανάλογα web services.

3.5.1.1 Books List Plugin

Το αρχείο Books List Plugin είναι το plugin για την αντίστοιχη υπηρεσία διαδικτύου. Με την χρήση του συγκεκριμένου plugin αρχικά, διαχωρίζεται το userId το οποίο προέρχεται από τον Soap Envelope. Ακολούθως χρησιμοποιείται το συγκεκριμένο userId για να αποκτηθεί πρόσβαση στα στοιχεία του συγκεκριμένου χρήστη. Προτού μπορέσει να δοθεί πρόσβαση στα στοιχεία του χρήστη με το userId όμως πρέπει να ελεγχθεί ότι υπάρχει η δυνατότητα πρόσβασης στην ηλικία του χρήστη.
Σε περίπτωση που η πρόσβαση είναι εφικτή, χρησιμοποιώντας το userId μπορεί να βρεθεί και να επιστραφεί η ηλικία του χρήστη, έτσι ώστε να χρησιμοποιηθεί για την εμφάνιση όλων των βιβλίων τα οποία αντιστοιχούν στην ηλικία του χρήστη.
Εάν η πρόσβαση στην ηλικία του χρήστη δεν είναι δυνατή διότι το απαγορεύει ο ίδιος ο χρήστης μέσω του xml αρχείου προτιμήσεών του, τότε το plugin δεν χρειάζεται να προχωρήσει σε οποιαδήποτε ενέργεια.

3.5.1.2 Buy Plugin

Το αρχείο Buy Plugin αποτελεί το plugin για την υπηρεσία διαδικτύου BuyService. Με την χρήση του συγκεκριμένου plugin ουσιαστικά αρχικά διαχωρίζεται το userId το οποίο προέρχεται από τον Soap Envelope. Ακολούθως χρησιμοποιείται το συγκεκριμένο userId για να αποκτηθεί πρόσβαση στα στοιχεία του συγκεκριμένου χρήστη. Προτού μπορέσει να δοθεί πρόσβαση στα προσωπικά στοιχεία του χρήστη μέσω του userId πρέπει να ελεγχθεί ότι είναι δυνατή η πρόσβαση στα στοιχεία του χρήστη τα οποία είναι απαραίτητα για να ολοκληρωθεί μία πληρωμή.

Έτσι χρησιμοποιώντας το userId βλέπω εάν υπάρχει πρόσβαση στο στοιχείο του χρήστη που με ενημερώνει για την επιλεγόμενη μέθοδο πληρωμής, δηλαδή το preferredPaymentMethod. Εάν η πρόσβαση δεν είναι εφικτή, τότε το plugin δεν προβαίνει σε οποιαδήποτε ενέργεια καθώς θα χρησιμοποιηθούν οι παράμετροι οι οποίοι αποστέλλονται από τον provider.

Ανάλογα με την μέθοδο πληρωμής εκτελούνται και οι αντίστοιχες ενέργειες. Αρχικά, υπάρχει η Cash Payment η οποία δέχεται το bookId σαν παράμετρο και στην περίπτωση που το ποσό δεν δοθεί, τότε μέσω του bookId ανακτάται η τιμή του συγκεκριμένου βιβλίου. Εάν ο χρήστης επιτρέπει την πρόσβαση στις πληροφορίες της πόλης και της χώρας στην οποία βρίσκεται την δεδομένη στιγμή, τότε το πρόγραμμα χρησιμοποιεί αρχικά την πόλη για να βρει βιβλιοπωλεία που τυχόν υπάρχουν. Στην συνέχεια, εάν δεν βρεθούν βιβλιοπωλεία στην πόλη αυτή τότε προσπαθεί να βρει στη χώρα στην οποία βρίσκεται την εν λόγω στιγμή ο χρήστης. Εάν δεν βρεθεί κάποιο βιβλιοπωλείο ούτε στην χώρα αλλά ούτε και στην πόλη στην οποία βρίσκεται ο χρήστης, τότε εμφανίζεται το ανάλογο μήνυμα για να ενημερωθεί ο χρήστης.

Σε περίπτωση που ο χρήστης δεν επιτρέπει πρόσβαση στις πληροφορίες της πόλης και της χώρας στις οποίες βρίσκεται την δεδομένη στιγμή, τότε το plugin δεν χρειάζεται να κάνει κάποια ενέργεια, αφού στον χρήστη θα παρουσιαστούν τα βιβλιοπωλεία τα οποία δόθηκαν σαν παράμετροι στον provider.

Ακόμα, υπάρχει η μέθοδος πληρωμής Credit Card Payment η οποία παίρνει και πάλι το bookId και το χρησιμοποιεί για να βρει την τιμή του βιβλίου στην περίπτωση που αυτή δεν δίνεται. Επίσης, εάν ο χρήστης δίνει πρόσβαση στις προσωπικές του πληροφορίες και συγκεκριμένα σε όλες τις πληροφορίες που αφορούν την πιστωτική του κάρτα, τότε χρησιμοποιώντας το userId ανακτώνται οι πληροφορίες αυτές. Ακολούθως, εάν ο χρήστης επιτρέπει την πρόσβαση στις πληροφορίες της πόλης και της χώρας όπου βρίσκεται τη δεδομένη στιγμή, τότε το πρόγραμμα χρησιμοποιεί αρχικά την πόλη για να βρει

βιβλιοπωλεία που τυχόν υπάρχουν. Στην συνέχεια, εάν δεν βρεθούν βιβλιοπωλεία στην πόλη αυτή, το πρόγραμμα προσπαθεί να βρει στην χώρα στην οποία βρίσκεται την δεδομένη στιγμή ο χρήστης. Σε περίπτωση που δεν βρεθεί κάποιο βιβλιοπωλείο, ούτε στην χώρα ούτε στην πόλη στην οποία βρίσκεται ο χρήστης, τότε εμφανίζεται το ανάλογο μήνυμα για να ενημερώσει το χρήστη.

Εάν ο χρήστης δεν επιτρέπει πρόσβαση στις πληροφορίες της πόλης και της χώρας στις οποίες βρίσκεται την δεδομένη στιγμή, τότε το plugin δεν χρειάζεται να προβεί σε κάποια περεταίρω ενέργεια αφού στον χρήστη θα παρουσιαστούν τα βιβλιοπωλεία τα οποία δόθηκαν στον provider.

3.5.1.3 Welcome Plugin

Το Welcome Plugin είναι το plugin για την WelcomingService υπηρεσία διαδικτύου. Στο συγκεκριμένο plugin αρχικά, διαχωρίζεται το userId που προέρχεται από τον Soap Envelope. Ακολούθως, γίνεται χρήση του userId για να αποκτηθεί πρόσβαση στα στοιχεία του συγκεκριμένου χρήστη. Προτού όμως μπορέσει να δοθεί πρόσβαση στα στοιχεία του χρήστη με το userId, πρέπει να ελεγχθεί εάν υπάρχει η δυνατότητα πρόσβασης στα στοιχεία του χρήστη τα οποία είναι απαραίτητα.

Αρχικά το σύστημα ελέγχει εάν δίνεται πρόσβαση στο όνομα του χρήστη και εάν η πρόσβαση είναι εφικτή τότε χρησιμοποιείται το userId για να την ανάκτησή του. Εάν ο χρήστης δεν επιτρέπει πρόσβαση στο όνομά του, τότε το plugin δεν εκτελεί κάποια περεταίρω ενέργεια, διότι χρησιμοποιείται το όνομα το οποίο δόθηκε στον provider.

Στην συνέχεια, ελέγχεται εάν υπάρχει πρόσβαση στο επίθετο του χρήστη και εάν αυτή είναι δυνατή, τότε το σύστημα χρησιμοποιεί το userId για να το ανακτήσει. Εάν ο χρήστης δεν επιτρέπει πρόσβαση στο επίθετο του τότε το plugin δεν κάνει κάτι, διότι χρησιμοποιείται το επίθετο το οποίο δόθηκε στον provider.

Τέλος, ελέγχεται από το σύστημα εάν υπάρχει πρόσβαση στις πληροφορίες της πόλης και της χώρας στην οποία βρίσκεται τώρα ο χρήστης. Στην περίπτωση αυτή, το σύστημα χρησιμοποιεί το userId για να επιστρέψει στο χρήστη κάποια γλώσσα από τον πίνακα γλωσσών.

Εάν ο χρήστης δεν επιτρέπει πρόσβαση στις πληροφορίες της πόλης και της χώρας στις οποίες βρίσκεται την δεδομένη στιγμή το plugin δεν χρειάζεται να πραγματοποιήσει κάποια ενέργεια αφού στον χρήστη θα παρουσιαστεί το μήνυμα στην γλώσσα η οποία δόθηκε στον provider.

3.5.2 Ανάλυση Providers Σεναρίου

3.5.2.1 Books List Provider

Το αρχείο Books List Provider είναι αρχείο που ανήκει στον Axis 2 Handler και χρησιμοποιείται για την εκτέλεση παραδειγμάτων τα οποία καλούν την υπηρεσία Books List Service.
Το αρχείο αυτό όπως προανέφερα εκτελεί ένα συγκεκριμένο σενάριο το οποίο του δόθηκε, όπου δίνονται συγκεκριμένες τιμές στις παραμέτρους των μεθόδων που επιθυμώ να καλέσω. Προτού όμως κληθούν οι μέθοδοι αυτοί ενεργοποιείται το Books List Plugin, το οποίο είναι υπεύθυνο να ελέγξει την πρόσβαση στις προσωπικές πληροφορίες του χρήστη και να εκτελέσει τις απαραίτητες τροποποιήσεις.
Αφού κληθεί η μέθοδος η οποία περιέχεται στο συγκεκριμένο σενάριο εμφανίζεται το μήνυμα με το αποτέλεσμα από την κλήση της μεθόδου αυτής.

3.5.2.2 Books Store Provider

Το αρχείο Books Store Provider είναι αρχείο το οποίο ανήκει στον Axis 2 Handler και χρησιμοποιείται για να εκτελεστούν από το σύστημα τα παραδείγματα τα οποία καλούν την υπηρεσία Books Store Service.
Το αρχείο αυτό, όπως προανέφερα, εκτελεί ένα συγκεκριμένο σενάριο το οποίο του δόθηκε. Σε κάθε σενάριο δίνονται τιμές στις παραμέτρους των μεθόδων οι οποίες θα κληθούν. Αφού κληθεί η μέθοδος του σεναρίου εμφανίζεται το μήνυμα, το οποίο αποτελεί το αποτέλεσμα από την κλήση της μεθόδου, η οποία εξετάζεται στο σενάριο αυτό.

3.5.2.3 Buying Provider

Το αρχείο Buying Provider αποτελεί αρχείο του Axis 2 Handler και χρησιμοποιείται για να την εκτέλεση συγκεκριμένων παραδειγμάτων, που καλούν την υπηρεσία Buy Service.
Το αρχείο εκτελεί ένα συγκεκριμένο σενάριο κάθε φορά το οποίο του δίνεται. Αρχικά,. Δίνονται κάποιες σταθερές τιμές στις παραμέτρους των μεθόδων που εξετάζονται. Προτού καλεστούν οι συγκεκριμένες μέθοδοι όμως ενεργοποιείται το Buy Plugin το οποίο είναι υπεύθυνο να ανακτήσει και να αντικαταστήσει τα πραγματικά χαρακτηριστικά του χρήστη τα οποία αφορούν την πληρωμή, δεδομένου ότι ο χρήστη έχει δώσει πρόσβαση σε αυτά.

Αφού κληθεί η μέθοδος του σεναρίου, η οποία εξετάζεται την δεδομένη στιγμή, εμφανίζεται το μήνυμα με το αποτέλεσμα από την κλήση της μεθόδου του σεναρίου.

3.5.2.4 Welcoming Provider

Το αρχείο Welcoming Provider ανήκει στον Axis 2 Handler και χρησιμοποιείται για την εκτέλεση συγκεκριμένων παραδειγμάτων, τα οποία με την σειρά τους καλούν την υπηρεσία Welcoming Service.

Το συγκεκριμένο αρχείο εκτελεί ένα σενάριο, κάθε φορά, το οποίο του δίνεται. Για κάθε μέθοδο η οποία καλείται από το σενάριο αυτό, δίνονται κάποιες σταθερές τιμές σαν παράμετροι. Το Welcome Plugin είναι υπεύθυνο, πριν την εκτέλεση κάθε μεθόδου, να ελέγξει εάν υπάρχει η δυνατότητα πρόσβασης στα στοιχεία του χρήστη τα οποία είναι απαραίτητα για την συγκεκριμένη μέθοδο και να τα αντικαταστήσει.

Αφού κληθεί η μέθοδος του σεναρίου, με τις κατάλληλες παραμέτρους, εμφανίζεται ένα μήνυμα με το αποτέλεσμα από την κλήση της μεθόδου του σεναρίου.

Κεφάλαιο 4

Σχεδίαση Σεναρίου χρήσης επέκτασης του συστήματος

4.1 Ανάλυση Σεναρίου 32
4.2 Απαιτήσεις Συστήματος 34
4.2.1 Λειτουργίες Συστήματος 34
4.2.2 Περιορισμοί Συστήματος 34
4.3 Data Flow Diagrams 35
4.3.1 WelcomingService 35
4.3.2 UserProfileService 36
4.3.3 CurrentLocationService 42
4.3.4 BuyService 45
4.3.5 BooksStoreService 47
4.3.6 BooksListService 50
4.4 Διαγράμματα Κλάσεων 56
4.4.1 WelcomingService 56
4.4.2 UserProfileService 57
4.4.3 CurrentLocationService 59
4.4.4 BuyService 60
4.4.5 BooksStoreService 61
4.4.6 BooksListService 63

4.1 Ανάλυση Σεναρίου

Το σενάριο το οποίο δημιούργησα ασχολείται με ένα διαδικτυακό βιβλιοπωλείο. Στο συγκεκριμένο βιβλιοπωλείο κάθε νόμιμος χρήστης του, έχει την δυνατότητα να βλέπει τα βιβλία τα οποία παρέχονται, να εκτελεί αναζητήσεις πάνω στα βιβλία με βάση κάποια κριτήρια και τέλος αφού επιλέξει κάποιο από αυτά να πληρώνει με όποια μέθοδο πληρωμής επιθυμεί. Για την υλοποίηση του πιο πάνω σεναρίου χρειάστηκε να υλοποιήσω 5 υπηρεσίες διαδικτύου (web services).

Οι υπηρεσίες διαδικτύου (web services) που δημιουργήθηκαν είναι η Current Location Service, η Books List Service, η User Profile Service, η Buy Service, η Books Store Service και τέλος η Welcoming Service.

Η Current Location Service, περιέχει ένα κατάλογο από χώρες, πόλεις και γλώσσες. Όταν κληθεί επιστρέφει τυχαία μια χώρα, μια πόλη και μια γλώσσα ως η χώρα και η πόλη στην οποία βρίσκεται κάποιος χρήστης τη δεδομένη στιγμή, καθώς επίσης και μία γλώσσα ως η τοπική γλώσσα.

Η υπηρεσία διαδικτύου Books List Service, απαρτίζεται από τα βιβλία τα οποία είναι διαθέσιμα στο διαδικτυακό μας βιβλιοπωλείο. Μέσω της συγκεκριμένης υπηρεσίας κάθε χρήστης μπορεί να διαχωρίσει τα βιβλία ανάλογα με την ηλικία του και έπειτα ανάλογα με κάποιο κωδικό του βιβλίου που του αρέσει. Επιπρόσθετα, κάθε χρήστης μπορεί να δει μία λίστα με όσα βιβλία ανήκουν σε συγκεκριμένες κατηγορίες οι οποίες του αρέσουν, όπως πχ. επιστημονική φαντασία, δράμα, μυστήριο κλπ. Επιπρόσθετα, ο χρήστης έχει τη δυνατότητα να δει όλα τα βιβλία τα οποία εκδόθηκαν κάποια συγκεκριμένη χρονιά. Ακόμα, ο χρήστης μπορεί να δει μια λίστα με βιβλία με βάση τη γλώσσα. Επίσης, προσφέρεται η δυνατότητα στον χρήστη να δει μία λίστα με βιβλία τα οποία το κόστος του κάθε βιβλίου δεν ξεπερνά ένα ποσό το οποίο θα καθορίσει ο ίδιος. Τέλος, μπορεί ο χρήστης να δει όσα βιβλία έχουν συγκεκριμένο συγγραφέα.

Έπειτα, η υπηρεσία διαδικτύου User Profile Service, παρέχει προσωπικά στοιχεία για κάθε ένα από τους χρήστες του βιβλιοπωλείου.

Στη συνέχεια, η υπηρεσία διαδικτύου Buy Service, ασχολείται με τον τρόπο πληρωμής για κάθε βιβλίο.

Επιπρόσθετα, η υπηρεσία διαδικτύου Books Store Service, περιέχει πληροφορίες για όλα τα βιβλιοπωλεία με τα οποία συνεργάζεται το Online Book Store.

Τέλος έχουμε την υπηρεσία διαδικτύου Welcoming Service, η οποία είναι υπεύθυνη να καλωσορίζει τον χρήστη στο διαδικτυακό βιβλιοπωλείο, στην γλώσσα του κάθε χρήστη.

4.2 Απαιτήσεις Συστήματος

4.2.1 Λειτουργίες Συστήματος

Οι λειτουργίες που που μπορούν να πραγματοποιηθούν, εκτελούνται από την υπηρεσία διαδικτύου που είναι υπεύθυνη για την διεκπεραίωση της συγκεκριμένης λειτουργίας.
Αρχικά, πραγματοποιείται η είσοδος του χρήστη στο σύστημα κατά την οποία του παρουσιάζεται το μήνυμα καλωσορίσματος στην τοπική γλώσσα ή στην γλώσσα που δόθηκε σαν παράμετρος κατά την κλήση της συγκεκριμένης μεθόδου. Η συγκεκριμένη λειτουργία εκτελείται από την υπηρεσία διαδικτύου Welcoming Service.
Έπειτα, θα παρουσιάζεται στον χρήστη ένας κατάλογος με όλα τα βιβλία τα οποία είναι διαθέσιμα, λειτουργία η οποία εκτελείται από την υπηρεσία διαδικτύου Books List Service.
Έχοντας πρόσβαση στον κατάλογο με τα βιβλία, ο χρήστης έχει την δυνατότητα να εκτελεί αναζητήσεις σε αυτά με βάση την γλώσσα, τον συγγραφέα, το είδος, το κόστος, τον κωδικό και την χρονιά έκδοσης του βιβλίου.
Στην συνέχεια, ο χρήστης μπορεί να επιλέξει το βιβλίο το οποίο επιθυμεί να αγοράσει. Η διεκπεραίωση της αγοράς του βιβλίου γίνεται μέσω της υπηρεσίας διαδικτύου Buy Service.
Όταν η πληρωμή ολοκληρωθεί, ο χρήστης μπορεί να δει ένα κατάλογο με τα βιβλιοπωλεία από τα οποία μπορεί να παραλάβει την παραγγελία του. Η αναζήτηση για τα βιβλιοπωλεία γίνεται με βάση την πόλη ή την χώρα στην οποία βρίσκεται ο χρήστης την δεδομένη στιγμή. Η αναζήτηση για τα βιβλιοπωλεία πραγματοποιείται μέσω της υπηρεσίας διαδικτύου Books Store Service. Το αποτέλεσμα της αναζήτησης επιστρέφεται στην υπηρεσία Buy Service όπου και εμφανίζεται.

4.2.2 Περιορισμοί Συστήματος

Όπως προανέφερα σκοπός της ατομικής διπλωματικής εργασίας μου είναι η διαχείριση των προτιμήσεων του χρήστη για την ιδιωτικότητα. Έτσι χρειάστηκε στις υπηρεσίες διαδικτύου που δημιούργησα να προστεθούν κάποιοι περιορισμοί για να γίνονται σεβαστές οι προτιμήσεις του χρήστη.

Οι περιορισμοί οι οποίοι εφαρμόζονται στις υπηρεσίες διαδικτύου του συγκεκριμένου σεναρίου δίνονται στην μορφή xml αρχείων και αποτελούνται από τις προτιμήσεις των ίδιων των χρηστών για την διαχείριση των πληροφορίες τους.

4.3 Data Flow Diagrams

4.3.1 Welcoming Service

Η υπηρεσία διαδικτύου Welcoming Service εμφανίζει στον χρήστη το μήνυμα καλωσορίσματος στο Online Book Store.

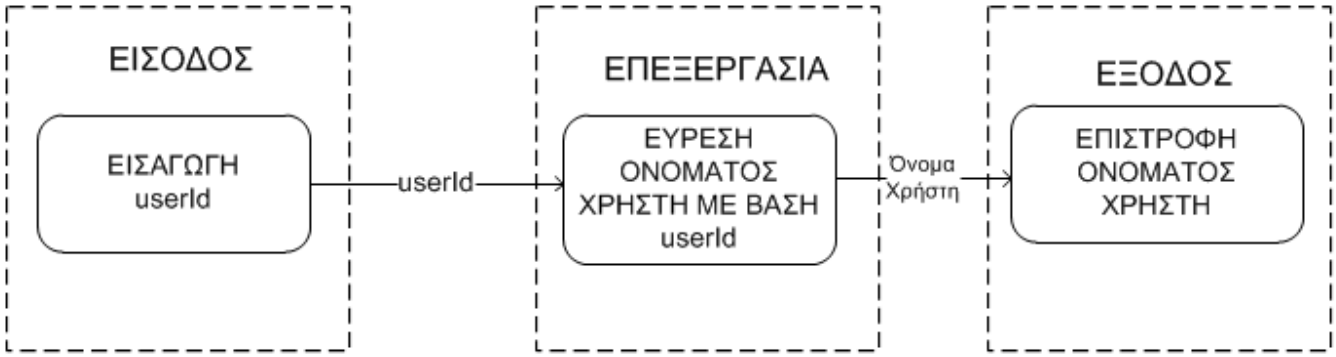
Σχήμα 4.1

Το Σχήμα 4.1 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου Welcoming Service. Η υπηρεσία διαδικτύου εμφανίζει μήνυμα καλωσορίσματος στο χρήστη, όταν αυτός εισέλθει στο σύστημα. Η λειτουργία της εμφάνισης του μηνύματος καλωσορίσματος στο χρήστη απαρτίζεται από την εισαγωγή του ονόματος, του επιθέτου καθώς και της γλώσσας του χρήστη. Ακολούθως, δημιουργείται το μηνύματος καλωσορίσματος του χρήστη στο σύστημα στην γλώσσα που επιλέχθηκε. Τέλος, το σύστημα εμφανίζει το μήνυμα καλωσορίσματος στη γλώσσα επιλογής του χρήστη.

4.3.2 User Profile Service

Η υπηρεσία διαδικτύου User Profile Service είναι υπεύθυνη να επιστρέφει / εμφανίζει τα στοιχεία ενός χρήστη. Πιο κάτω βρίσκονται τα Data Flow Datagrams της επιστροφής του ονόματος του χρήστης, του επιθέτου, της ηλικίας, του προτεινόμενου τρόπου πληρωμής, του τύπου της πιστωτικής του κάρτας, του αριθμού της κάρτας του, του μήνα λήξης της πιστωτικής του κάρτας καθώς και της χρονιάς λήξης της πιστωτικής κάρτας του χρήστη.

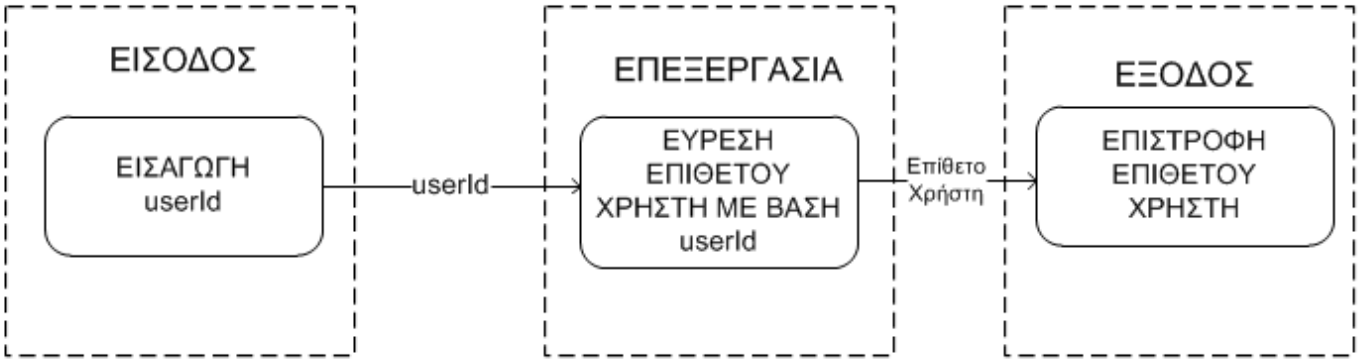
Αρχικά θα αναλύσουμε την περίπτωση κατά την οποία γίνεται η ανάκτηση του ονόματος ενός χρήστη με βάση το μοναδικό πεδίο UserId.



Σχήμα 4.2

Το Σχήμα 4.2 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου User Profile Service. Η υπηρεσία διαδικτύου επιστρέφει το όνομα ενός χρήστη ο οποίος έχει ως χαρακτηριστικό ένα συγκεκριμένο UserId. Η λειτουργία της ανάκτησης του ονόματος του χρήστη απαρτίζεται από την εισαγωγή του μοναδικού πεδίου UserId του χρήστη σαν είσοδο. Έπειτα σαν επεξεργασία γίνεται η εύρεση του ονόματος του χρήστη με το συγκεκριμένο UserId και στην συνέχεια η επιστροφή του ονόματος του χρήστη σαν έξοδος.

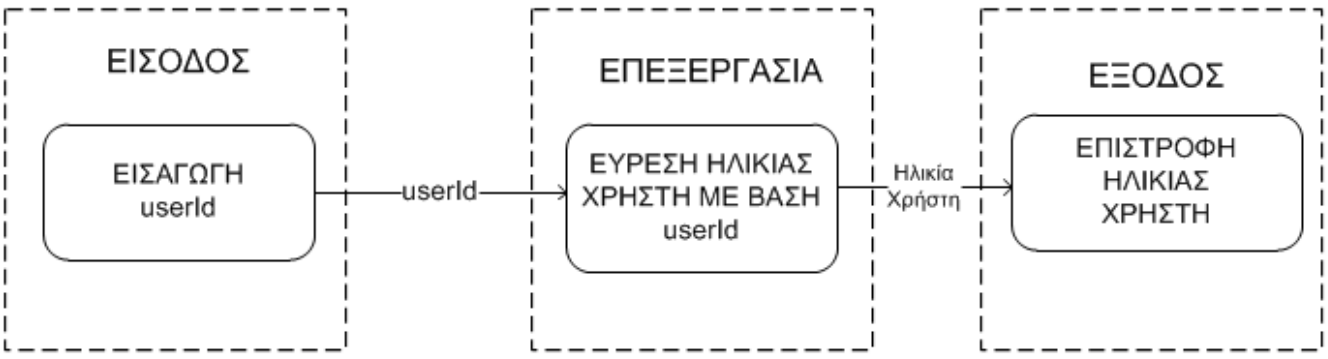
Ακολούθως, θα επιχειρήσω να αναλύσω την περίπτωση στην οποία γίνεται η ανάκτηση του επιθέτου ενός χρήστη με βάση το μοναδικό πεδίο UserId.



Σχήμα 4.3

Το Σχήμα 4.3 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου User Profile Service. Η υπηρεσία διαδικτύου επιστρέφει το επίθετο ενός χρήστη ο οποίος έχει ως χαρακτηριστικό ένα συγκεκριμένο UserId. Η λειτουργία της ανάκτησης του επιθέτου του χρήστη αποτελείται από την εισαγωγή του μοναδικού πεδίου UserId του χρήστη σαν είσοδο. Έπειτα, σαν επεξεργασία γίνεται η εύρεση του επιθέτου του χρήστη με το συγκεκριμένο UserId και στην συνέχεια το σύστημα επιστρέφει το επίθετο του χρήστη σαν έξοδος.

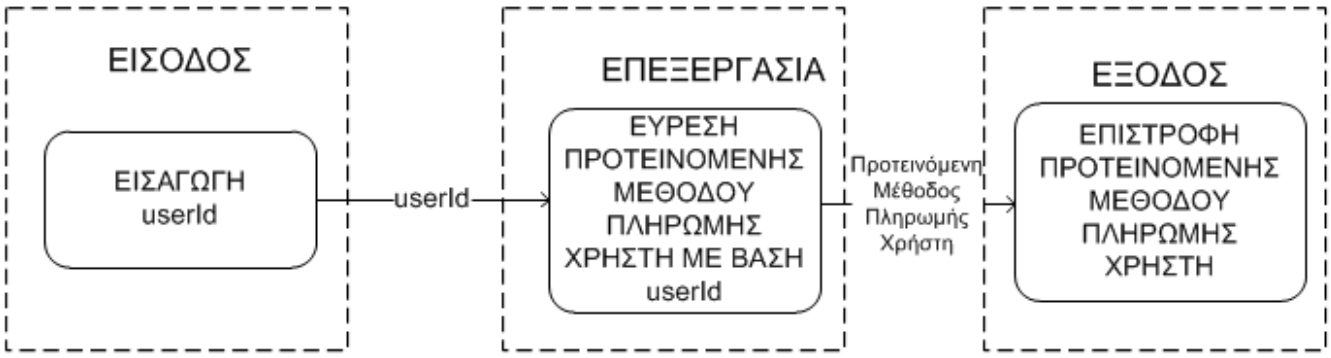
Στο σημείο αυτό, θα αναλυθεί η περίπτωση στην οποία γίνεται η ανάκτηση της ηλικίας ενός χρήστη με βάση το μοναδικό πεδίο UserId.



Σχήμα 4.4

Το Σχήμα 4.4 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου User Profile Service. Η υπηρεσία διαδικτύου επιστρέφει την ηλικία ενός χρήστη ο οποίος έχει ως χαρακτηριστικό ένα συγκεκριμένο UserId. Η λειτουργία της ανάκτησης της ηλικίας του χρήστη εμπεριέχει την εισαγωγή του μοναδικού πεδίου UserId του χρήστη σαν είσοδο. Η επεξεργασία περιλαμβάνει την εύρεση της ηλικίας του χρήστη με το συγκεκριμένο UserId και στην συνέχεια η ηλικία του χρήστη επιστρέφεται σαν έξοδος.

Επιπρόσθετα, περιγράφεται η περίπτωση στην οποία γίνεται η ανάκτηση της προτεινόμενης μεθόδου πληρωμής ενός χρήστη με βάση το μοναδικό πεδίο UserId.



Σχήμα 4.5

Το Σχήμα 4.5 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου User Profile Service. Η υπηρεσία διαδικτύου επιστρέφει την προτεινόμενη μέθοδο πληρωμής ενός χρήστη ο οποίος έχει ως χαρακτηριστικό ένα συγκεκριμένο UserId. Η λειτουργία της ανάκτησης της προτεινόμενης μεθόδου πληρωμής του χρήστη απαρτίζεται από την εισαγωγή του μοναδικού πεδίου UserId του χρήστη σαν είσοδο. Ακολούθως, σαν επεξεργασία γίνεται η εύρεση της προτεινόμενης μεθόδου πληρωμής του χρήστη με το συγκεκριμένο UserId και στην συνέχεια πραγματοποιείται η επιστροφή της προτεινόμενης μεθόδου πληρωμής του χρήστη σαν έξοδος.

Στην συνέχεια, με την βοήθεια του Σχήματος 4.6 θα επιχειρήσω να αναλύσω την περίπτωση κατά την οποία πραγματοποιείται η ανάκτηση του τύπου της πιστωτικής κάρτας ενός χρήστη με βάση το μοναδικό πεδίο UserId.

Σχήμα 4.6

Το Σχήμα 4.6 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου User Profile Service. Η υπηρεσία διαδικτύου επιστρέφει τον τύπο της πιστωτικής κάρτας ενός χρήστη ο οποίος έχει ως χαρακτηριστικό ένα συγκεκριμένο UserId. Η λειτουργία της ανάκτησης του τύπου της πιστωτικής κάρτας του χρήστη αποτελείται από την εισαγωγή του μοναδικού πεδίου UserId του χρήστη σαν είσοδο. Σαν επεξεργασία γίνεται η εύρεση της ανάκτησης του τύπου της πιστωτικής κάρτας του χρήστη με το συγκεκριμένο UserId και στην συνέχεια γίνεται η επιστροφή της ανάκτησης του τύπου της πιστωτικής κάρτας του χρήστη σαν έξοδος.

Επίσης, θα παρουσιαστεί η περίπτωση όπου γίνεται η ανάκτηση του αριθμού της πιστωτικής κάρτας ενός χρήστη, με βάση το μοναδικό πεδίο UserId.

Σχήμα 4.7

Το Σχήμα 4.7, απεικονίζει το Data Flow Datagram για την υπηρεσία διαδικτύου User Profile Service. Η υπηρεσία διαδικτύου επιστρέφει τον αριθμό της πιστωτικής κάρτας ενός χρήστη ο οποίος έχει ως χαρακτηριστικό ένα συγκεκριμένο UserId. Η λειτουργία της ανάκτησης του αριθμού της πιστωτικής κάρτας του χρήστη περιέχει την εισαγωγή του μοναδικού πεδίου UserId του χρήστη σαν είσοδο. Έπειτα, σαν επεξεργασία γίνεται η εύρεση της ανάκτησης του αριθμού της πιστωτικής κάρτας του χρήστη με το συγκεκριμένο UserId και στην συνέχεια πραγματοποιείται η επιστροφή της ανάκτησης του αριθμού της πιστωτικής κάρτας του χρήστη σαν έξοδος.

Ακόμη, θα αναλύσουμε την περίπτωση όπου γίνεται η ανάκτηση του μήνα λήξης της πιστωτικής κάρτας ενός χρήστη με βάση το μοναδικό πεδίο UserId.

Σχήμα 4.8

Το Σχήμα 4.8 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου User Profile Service. Η υπηρεσία διαδικτύου επιστρέφει τον μήνα λήξης της πιστωτικής κάρτας του χρήστη, ο οποίος έχει ως χαρακτηριστικό ένα συγκεκριμένο UserId. Η λειτουργία της ανάκτησης του μήνα λήξης της πιστωτικής κάρτας του χρήστη απαρτίζεται από την εισαγωγή του μοναδικού πεδίου UserId του χρήστη σαν είσοδο. Έπειτα, σαν επεξεργασία γίνεται η εύρεση της ανάκτησης του μήνα λήξης της πιστωτικής κάρτας του χρήστη με το συγκεκριμένο UserId. Τέλος, σαν έξοδος πραγματοποιείται η επιστροφή της ανάκτησης του μήνα λήξης της πιστωτικής κάρτας του χρήστη σαν έξοδο.

Εν κατακλείδι, θα παρουσιαστεί η περίπτωση κατά την οποία γίνεται η ανάκτηση της χρονιάς λήξης της πιστωτικής κάρτας ενός χρήστη με βάση το μοναδικό πεδίο UserId.

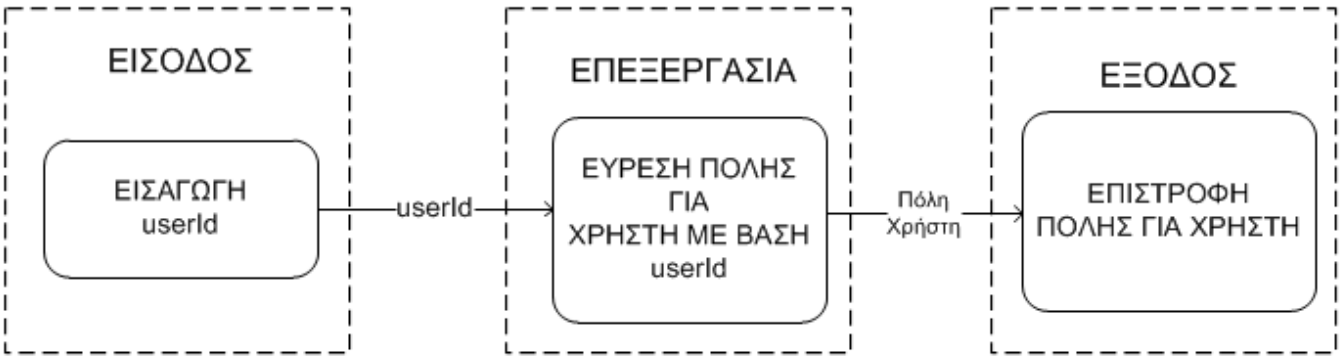
Σχήμα 4.9

Το Σχήμα 4.9 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου User Profile Service. Η υπηρεσία διαδικτύου επιστρέφει τη χρονιά λήξης της πιστωτικής κάρτας ενός χρήστη, ο οποίος έχει ως χαρακτηριστικό ένα συγκεκριμένο UserId. Η λειτουργία της ανάκτησης της χρονιάς λήξης της πιστωτικής κάρτας του χρήστη εμπεριέχει την εισαγωγή του μοναδικού πεδίου UserId του χρήστη σαν είσοδο. Σαν επεξεργασία, γίνεται η εύρεση της ανάκτησης της χρονιάς λήξης της πιστωτικής κάρτας του χρήστη με το συγκεκριμένο UserId. Τέλος, γίνεται η επιστροφή της ανάκτησης της χρονιάς λήξης της πιστωτικής κάρτας του χρήστη σαν έξοδος.

4.3.3 Current Location Service

Η υπηρεσία διαδικτύου Current Location Service είναι υπεύθυνη να επιστρέφει / εμφανίζει την πόλη ή την χώρα στην οποία βρίσκεται ένας χρήστης καθώς επίσης και την τοπική γλώσσα για την περιοχή στην οποία βρίσκεται ο χρήστης.

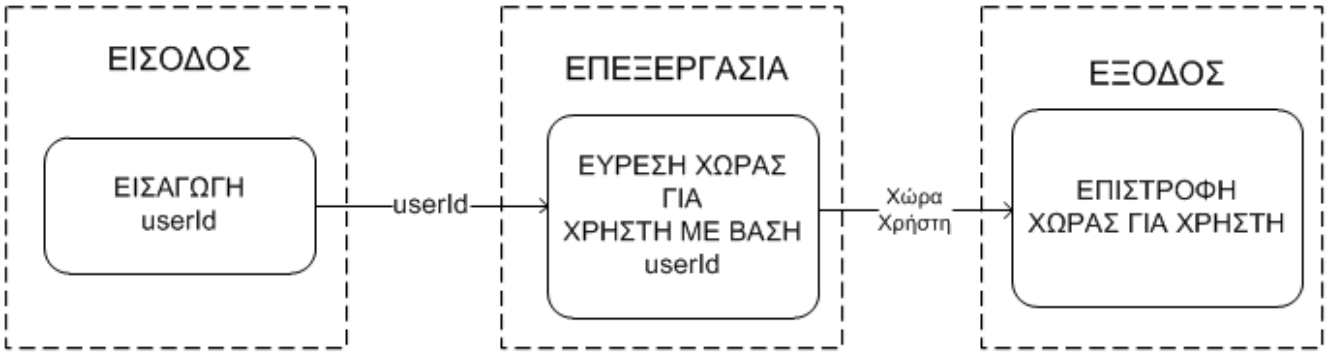
Αρχικά, θα αναλύσουμε την περίπτωση κατά την οποία ζητείται η πόλη στην οποία βρίσκεται ο χρήστης.



Σχήμα 4.10

Το Σχήμα 4.10 παρουσιάζει το Data Flow Datagram για την υπηρεσία διαδικτύου Current Location Service. Η υπηρεσία διαδικτύου είναι υπεύθυνη να εμφανίζει την πόλη στην οποία βρίσκεται ένας χρήστης με ένα συγκεκριμένο UserId. Η λειτουργία της εύρεσης της πόλης στην οποία βρίσκεται ο χρήστης απαρτίζεται αρχικά από την είσοδο, όπου γίνεται η εισαγωγή του UserId του χρήστη. Έπειτα, ως επεξεργασία πραγματοποιείται από το σύστημα η εύρεση μιας τυχαίας πόλης ως η πόλη στην οποία βρίσκεται ο χρήστης με το συγκεκριμένο UserId και ακολούθως σαν έξοδο γίνεται η επιστροφή/εμφάνιση της πόλης στην οποία βρίσκεται ο συγκεκριμένος χρήστης.

Στην συνέχεια θα αναλύσουμε την περίπτωση όπου ζητείται η χώρα στην οποία βρίσκεται ο χρήστης.



Σχήμα 4.11

Το Σχήμα 4.11 απεικονίζει το Data Flow Datagram για την υπηρεσία διαδικτύου Current Location Service. Η υπηρεσία διαδικτύου είναι υπεύθυνη να επιστρέφει την χώρα στην οποία βρίσκεται ένας χρήστης με ένα συγκεκριμένο UserId. Η λειτουργία της εύρεσης της χώρας στην οποία βρίσκεται ο χρήστης περιέχει αρχικά σαν είσοδο την εισαγωγή του UserId του χρήστη. Έπειτα, ως επεξεργασία πραγματοποιείται η εύρεση μιας τυχαίας χώρας ως η χώρα στην οποία βρίσκεται ο χρήστης με το συγκεκριμένο UserId και ακολούθως σαν έξοδο γίνεται η επιστροφή/εμφάνιση της χώρας στην οποία βρίσκεται ο χρήστης με το συγκεκριμένο UserId.

Τέλος, θα αναλύσουμε την περίπτωση στην οποία ζητείται η γλώσσα της τοποθεσίας στην οποία βρίσκεται την δεδομένη στιγμή ο χρήστης.

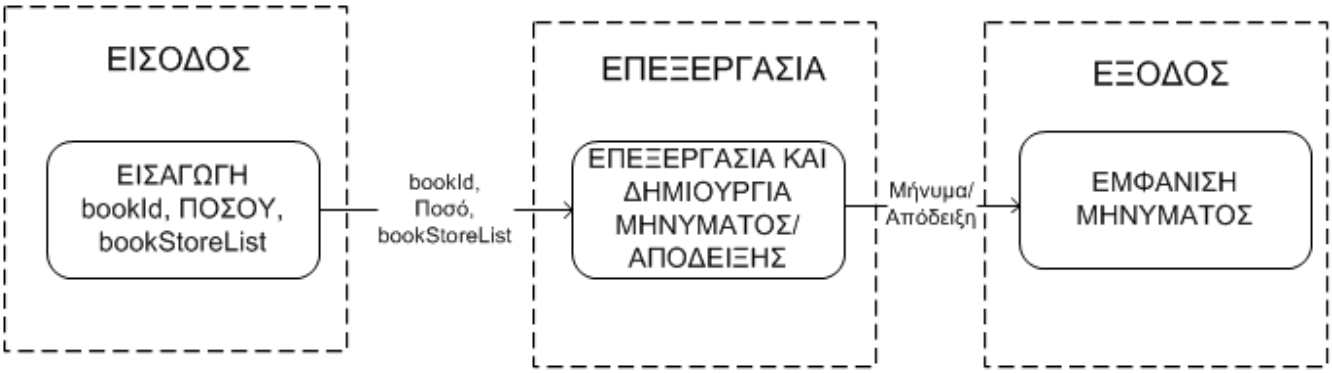
Σχήμα 4.12

Το Σχήμα 4.12 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου Current Location Service. Η υπηρεσία διαδικτύου είναι υπεύθυνη να επιστρέφει την τοπική γλώσσα για το χρήστη με ένα συγκεκριμένο UserId. Η λειτουργία της εύρεσης της τοπικής γλώσσας στην περιοχή στην οποία βρίσκεται ο χρήστης περιέχει αρχικά σαν είσοδο την εισαγωγή του UserId του χρήστη. Έπειτα, ως επεξεργασία πραγματοποιείται η εύρεση μιας τυχαίας γλώσσας ως η τοπική γλώσσα για το χρήστης με το συγκεκριμένο UserId και ακολούθως σαν έξοδο έχω την επιστροφή/εμφάνιση της τοπικής γλώσσας για το χρήστης με το συγκεκριμένο UserId.

4.3.4 Buy Service

Η υπηρεσία διαδικτύου Buy Service είναι υπεύθυνη να εκτελεί την πληρωμή για την αγορά του χρήστη. Η πληρωμή του χρήστη μπορεί να γίνει είτε με μετρητά είτε με επιταγή.

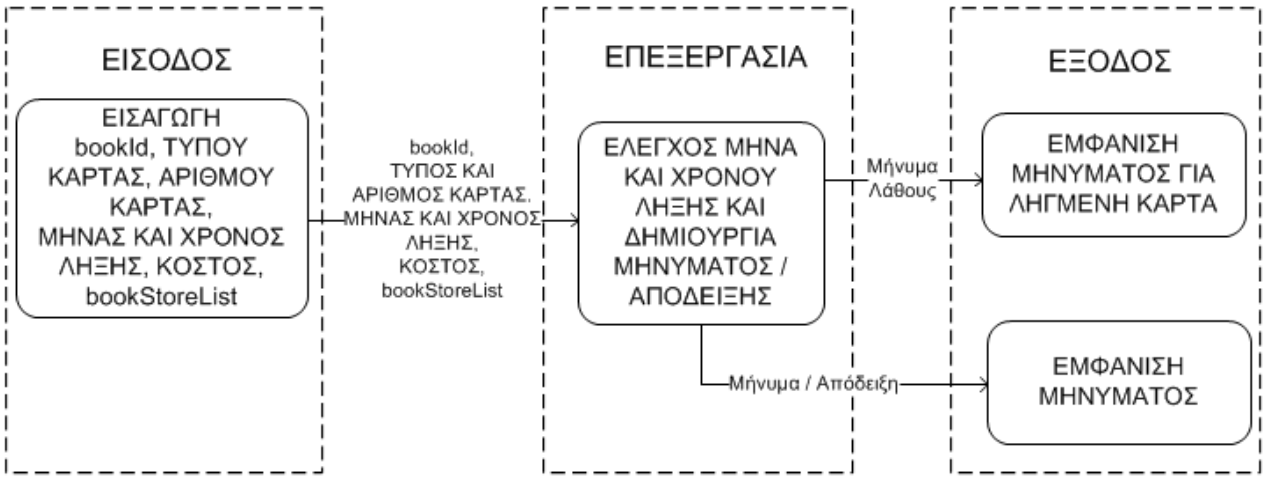
Αρχικά θα αναλύσουμε την περίπτωση όπου η πληρωμή γίνεται με μετρητά.



Σχήμα 4.13

Το Σχήμα 4.13 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου Buy Service. Η υπηρεσία διαδικτύου είναι υπεύθυνη να εκτελεί την πληρωμή για την αγορά του χρήστη με βάση το ποσό της αγοράς. Η λειτουργία της πληρωμής την οποία εκτελεί ο χρήστης περιέχει αρχικά σαν είσοδο την εισαγωγή του bookId ή / και του ποσού της αγοράς που έκανε ο χρήστης. Έπειτα, ως επεξεργασία πραγματοποιείται η επεξεργασία της πληρωμής και η δημιουργία μηνύματος/απόδειξης προς τον χρήστη και ακολούθως σαν έξοδο γίνεται η εμφάνιση του μηνύματος.

Στην συνέχεια θα δούμε την περίπτωση όπου η πληρωμή γίνεται με πιστωτική κάρτα.



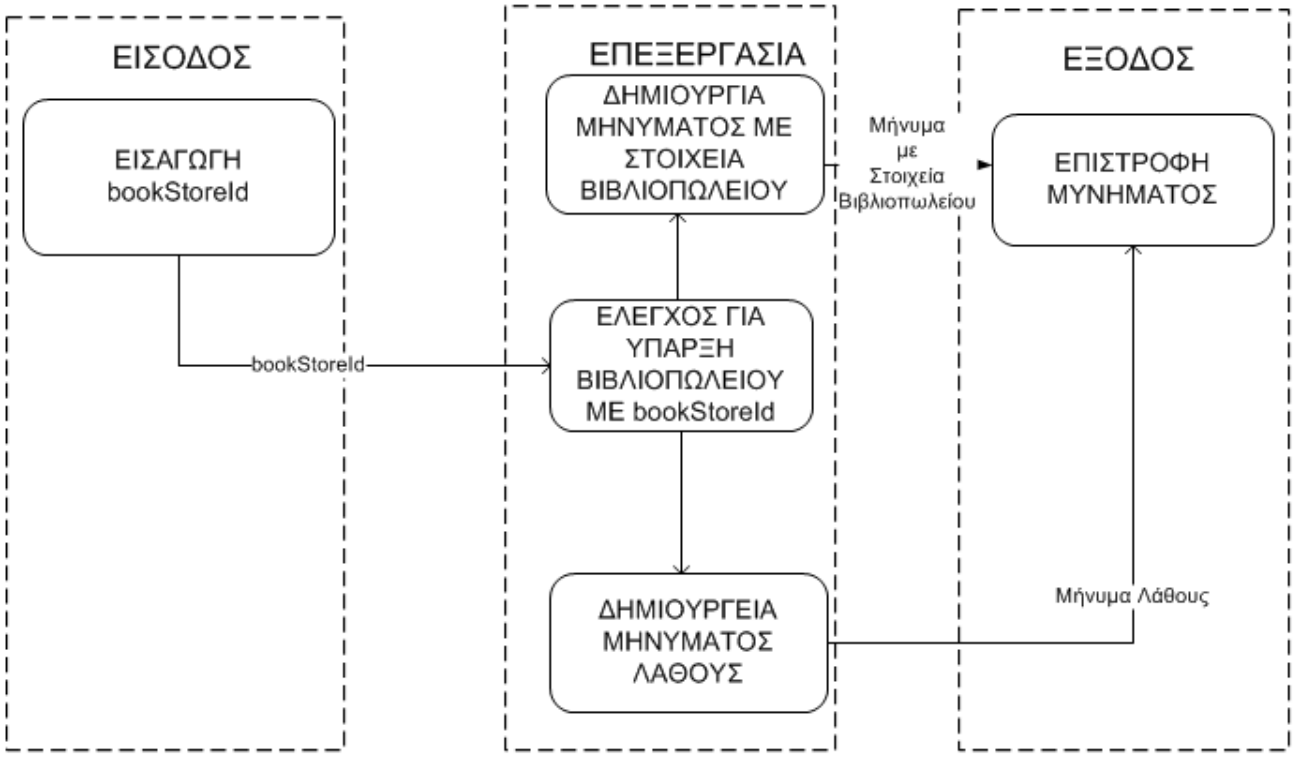
Σχήμα 4.14

Το Σχήμα 4.14 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου Buy Service. Η υπηρεσία διαδικτύου είναι υπεύθυνη να εκτελεί την πληρωμή για την αγορά του χρήστη με βάση το ποσό της αγοράς. Η λειτουργία της πληρωμής την οποία εκτελεί ο χρήστης απαρτίζεται από την εισαγωγή του bookId, του ποσού της αγοράς που έκανε ο χρήστης, του αριθμού της κάρτας καθώς και το μήνα και το χρόνο λήξης της κάρτας σαν είσοδο του συστήματος. Έπειτα, ως επεξεργασία πραγματοποιείται ο έλεγχος του μήνα και του χρόνου για την λήξη της κάρτας. Εάν η κάρτα έχει λήξει τότε εμφανίζεται μήνυμα λάθους. Σε αντίθετη περίπτωση, γίνεται επεξεργασία της πληρωμής και δημιουργείται ένα μήνυμα/απόδειξη το οποίο θα εμφανιστεί στον χρήστη. Τέλος, σαν έξοδο έχουμε την εμφάνιση του μηνύματος/απόδειξης προς τον χρήστη.

4.3.5 Books Store Service

Η υπηρεσία διαδικτύου Books Store Service είναι υπεύθυνη να εμφανίζει λίστα με όλα τα βιβλιοπωλεία τα οποία βρίσκονται σε μια συγκεκριμένη χώρα ή μια συγκεκριμένη πόλη.

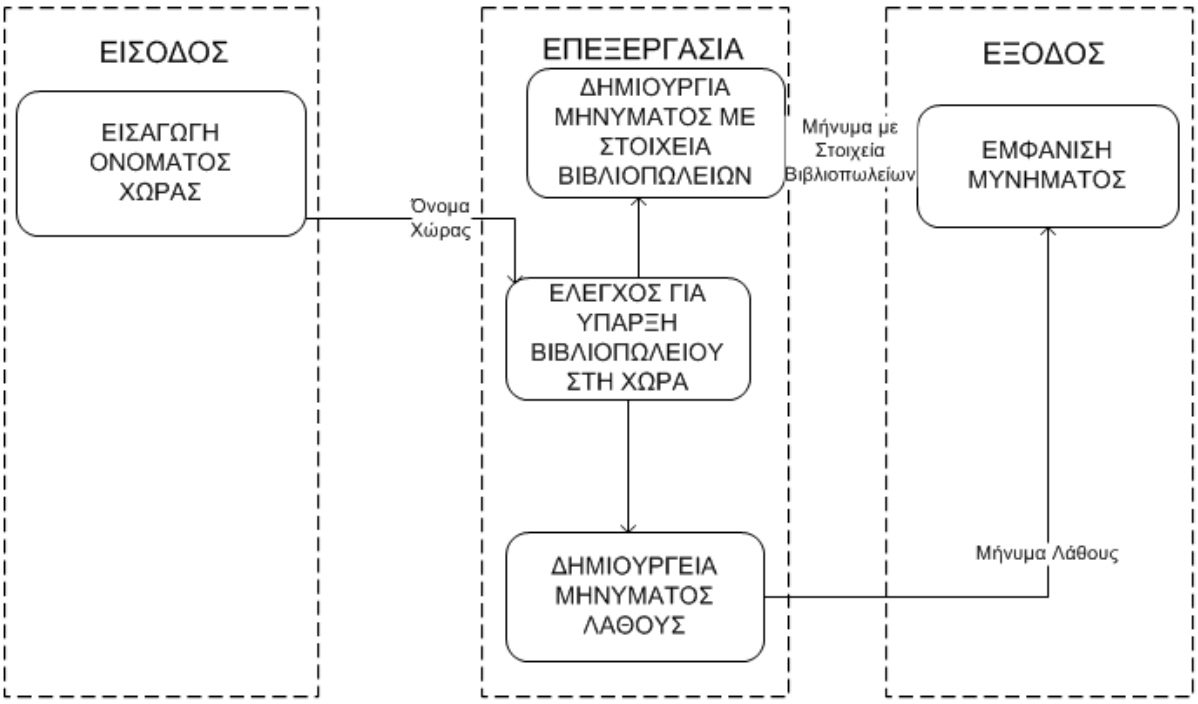
Αρχικά, θα αναλύσουμε την περίπτωση όπου η εύρεση του βιβλιοπωλείου γίνεται με βάση ένα συγκεκριμένο BookStoreId..



Σχήμα 4.15

Το Σχήμα 4.15 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου Books Store Service. Η υπηρεσία διαδικτύου είναι υπεύθυνη να επιστρέφει τα στοιχεία ενός συγκεκριμένου βιβλιοπωλείου. Η λειτουργία της εύρεσης του βιβλιοπωλείου περιέχει σαν είσοδο την εισαγωγή του μοναδικού bookStoreId. Έπειτα, στην επεξεργασία γίνεται ο έλεγχος για την ύπαρξη βιβλιοπωλείου με το συγκεκριμένο bookStoreId. Εάν δεν υπάρχουν το συγκεκριμένο βιβλιοπωλείο δημιουργείται ένα μήνυμα λάθους. Εάν υπάρχει, τότε δημιουργείται ένα μήνυμα με τα στοιχεία του βιβλιοπωλείου με το συγκεκριμένο bookStoreId. Στην έξοδο έχουμε την επιστροφή του μηνύματος που δημιουργήθηκε μετά τον έλεγχο για την εύρεση του βιβλιοπωλείου με το bookStoreId.

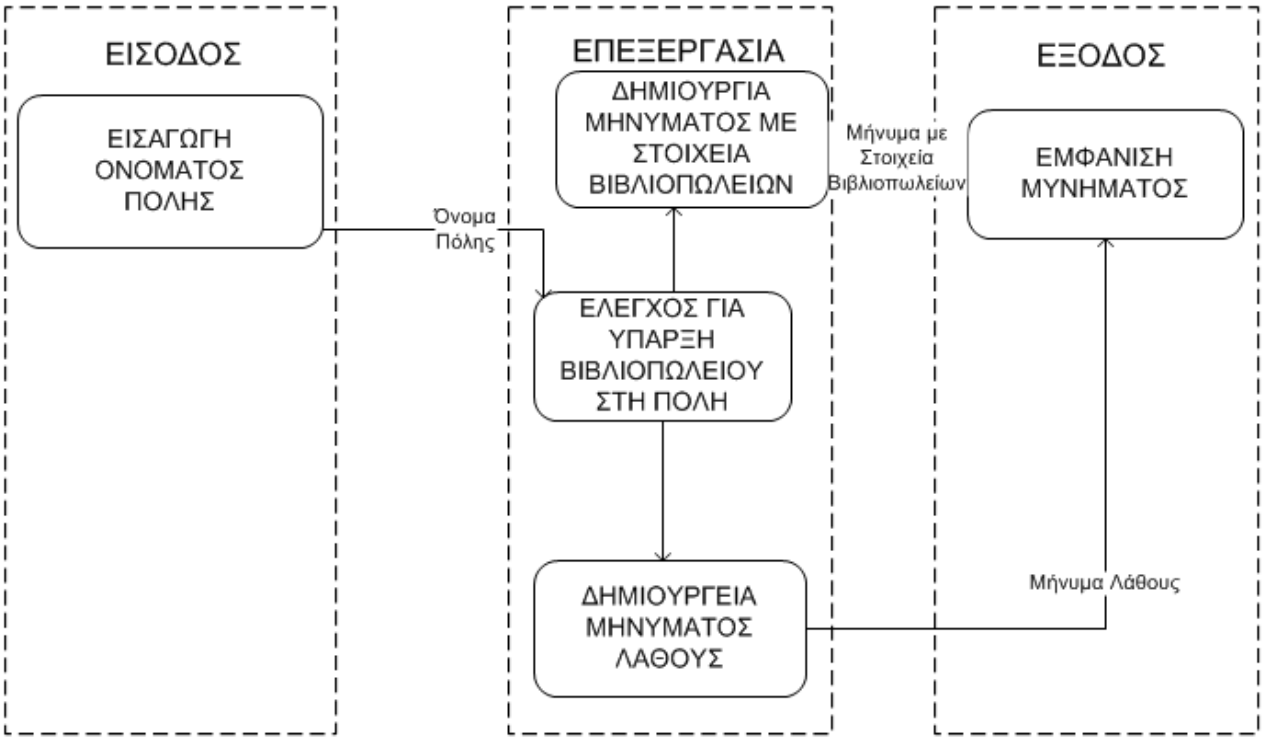
Επιπρόσθετα, θα αναλύσουμε την περίπτωση όπου η εύρεση των βιβλιοπωλείων γίνεται με βάση την χώρα στην οποία βρίσκονται.



Σχήμα 4.16

Το Σχήμα 4.16 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου Books Store Service. Η υπηρεσία διαδικτύου είναι υπεύθυνη να εμφανίζει μία λίστα με όλα τα βιβλιοπωλεία τα οποία πληρούν συγκεκριμένα χαρακτηριστικά και στην συγκεκριμένη περίπτωση χρειάζεται να εμφανίσει μία λίστα με όλα τα βιβλιοπωλεία τα οποία βρίσκονται στην συγκεκριμένη χώρα. Η λειτουργία της εύρεσης όλων των βιβλιοπωλείων τα οποία βρίσκονται σε μία συγκεκριμένη χώρα περιέχει αρχικά σαν είσοδο την εισαγωγή του ονόματος της χώρας. Έπειτα, ως επεξεργασία τον έλεγχο για την ύπαρξη βιβλιοπωλείων στην συγκεκριμένη χώρα. Εάν δεν υπάρχουν βιβλιοπωλεία, δημιουργείται ένα μήνυμα το οποίο ενημερώνει τον χρήστη για την μη ύπαρξη βιβλιοπωλείων στην συγκεκριμένη χώρα. Εάν υπάρχουν, δημιουργείται ένα μήνυμα με όλα τα βιβλιοπωλεία τα οποία υπάρχουν στην συγκεκριμένη πόλη. Σαν έξοδο έχουμε την εμφάνιση του μηνύματος και την επιστροφή της λίστας.

Ακολούθως, θα εξετάσουμε την περίπτωση όπου η εύρεση των βιβλιοπωλείων γίνεται με βάση την πόλη.



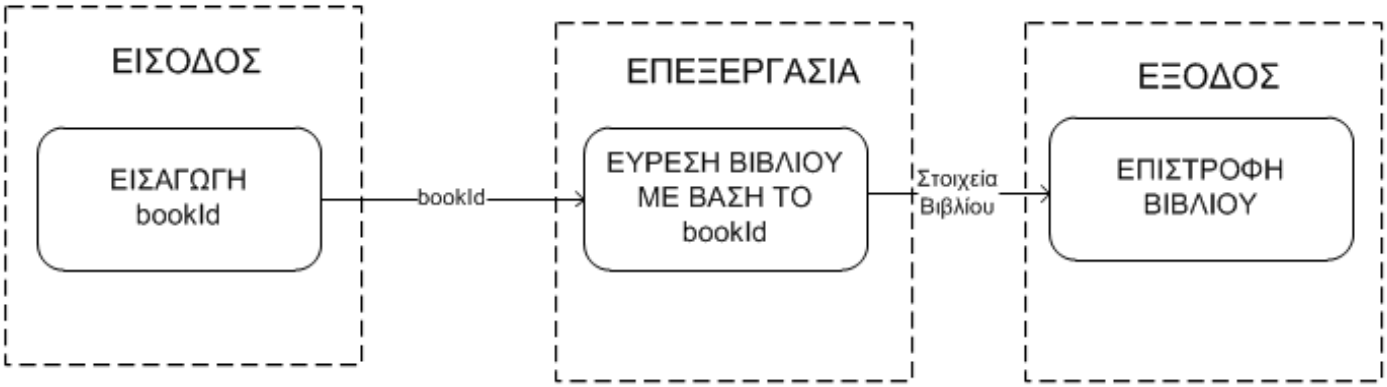
Σχήμα 4.17

Το Σχήμα 4.17 αποτελεί το Data Flow Datagram για την υπηρεσία διαδικτύου Books Store Service. Η συγκεκριμένη υπηρεσία διαδικτύου είναι υπεύθυνη να εμφανίζει ένα κατάλογο με όλα τα βιβλιοπωλεία τα οποία βρίσκονται στην συγκεκριμένη πόλη. Η λειτουργία αυτή περιέχει αρχικά σαν είσοδο την εισαγωγή του ονόματος της πόλης. Στην συνέχεια, ως επεξεργασία έχουμε την τον έλεγχο για την ύπαρξη βιβλιοπωλείων στην συγκεκριμένη πόλη. Εάν δεν υπάρχουν βιβλιοπωλεία στην υπό εξέταση πόλη, δημιουργείται ένα μήνυμα λάθους για να ενημερώσει τον χρήστη. Στην αντίθετη περίπτωση, δημιουργείται ένα μήνυμα με όλα τα βιβλιοπωλεία τα οποία υπάρχουν στην συγκεκριμένη πόλη. Σαν έξοδο, παρέχεται το μήνυμα και η επιστροφή της λίστας με τα στοιχεία των βιβλιοπωλείων.

4.3.6 Books List Service

Η υπηρεσία διαδικτύου Book List Service είναι υπεύθυνη να εμφανίζει λίστα με όλα τα βιβλία τα οποία πληρούν κάποια συγκεκριμένα χαρακτηριστικά όπως το συγγραφέα, την τιμή, το είδος κλπ.

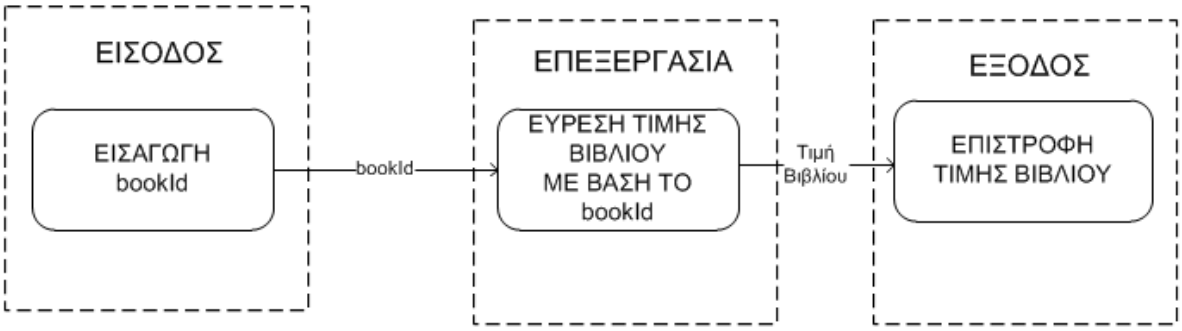
Αρχικά, θα αναλύσουμε την περίπτωση όπου γίνεται εύρεση ενός βιβλίου με βάση το μοναδικό πεδίο BookId.



Σχήμα 4.18

Το Σχήμα 4.18 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου Book List Service. Η λειτουργία της εύρεσης του βιβλίου με βάση το μοναδικό πεδίο BookId περιέχει αρχικά σαν είσοδο την εισαγωγή της τιμής του πεδίου BookId. Έπειτα, ως επεξεργασία έχουμε την εύρεση του συγκεκριμένου βιβλίου. Σαν έξοδο, επιστρέφονται τα στοιχεία του βιβλίου το οποία έχει το συγκεκριμένο BookId.

Επιπρόσθετα, θα αναλύσουμε την περίπτωση όπου γίνεται εύρεση της τιμής ενός συγκεκριμένου βιβλίου με βάση το bookId του.



Σχήμα 4.19

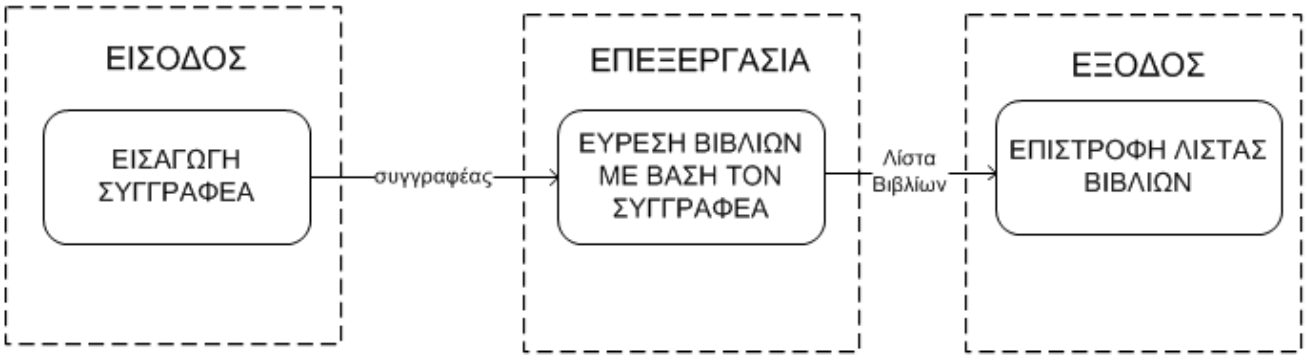
Το Σχήμα 4.19 αποτελεί το Data Flow Datagram για την υπηρεσία διαδικτύου Book List Service. Η λειτουργία της εύρεσης της τιμής ενός συγκεκριμένου βιβλίου το οποίο έχει το μοναδικό bookId περιέχει αρχικά σαν είσοδο την εισαγωγή του bookId το οποίο χρησιμοποιείται για τον έλεγχο. Έπειτα, ως επεξεργασία εκτελείται η εύρεση του βιβλίου το οποίο έχει το μοναδικό αυτό bookId. Σαν έξοδο, γίνεται η εμφάνιση της τιμής του βιβλίου, του οποίο το bookId του αντιστοιχεί στο bookId το οποίο δόθηκε σαν είσοδος, και επιστρέφεται.

Ακολούθως, θα με την βοήθεια του Σχήματος 4.20 περιγράφεται η περίπτωση όπου η εύρεση των βιβλίων γίνεται με βάση την ηλικία του χρήστη.

Σχήμα 4.20

Το Data Flow Datagram περιγράφεται στο Σχήμα 4.20 για την υπηρεσία διαδικτύου Book List Service. Η υπηρεσία διαδικτύου εμφανίζει όλα τα βιβλία τα οποία πληρούν συγκεκριμένα χαρακτηριστικά και στην συγκεκριμένη περίπτωση χρειάζεται να εμφανίσει μία λίστα με όλα τα βιβλία τα οποία αντιστοιχούν στην ηλικία του χρήστη. Η λειτουργία της εύρεσης όλων των βιβλίων με βάση την ηλικία του χρήστη περιέχει αρχικά σαν είσοδο την εισαγωγή της ηλικίας του χρήστη. Έπειτα, ως επεξεργασία έχουμε την εύρεση όλων των βιβλίων τα οποία αντιστοιχούν στην ηλικία του χρήστη και την δημιουργία μιας λίστας με αυτά. Σαν έξοδο έχουμε την επιστροφή της λίστας αυτής με τα στοιχεία των βιβλίων τα οποία αντιστοιχούν στην ηλικία του χρήστη.

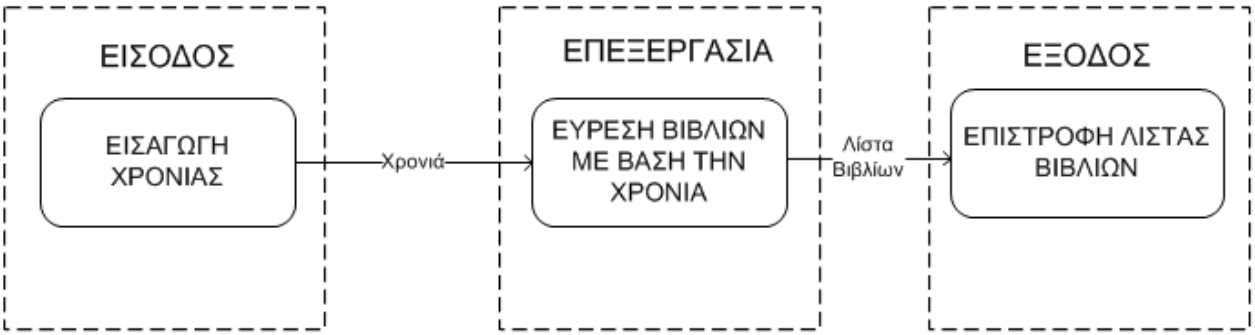
Επιπρόσθετα, θα αναλύσουμε την περίπτωση όπου η εύρεση των βιβλίων γίνεται με βάση ένα συγκεκριμένο συγγραφέα.



Σχήμα 4.21

Το Σχήμα 4.21 αποτελεί το Data Flow Datagram για την υπηρεσία διαδικτύου Book List Service. Η μέθοδος της υπηρεσία διαδικτύου είναι υπεύθυνη να εμφανίζει ένα κατάλογο με όλα τα βιβλία τα οποία αντιστοιχούν σε ένα συγκεκριμένο συγγραφέα. Είσοδο, της λειτουργίας της εύρεσης όλων των βιβλίων με βάση ένα συγκεκριμένο συγγραφέα, αποτελεί η εισαγωγή του ονόματος του συγγραφέα. Έπειτα, ως επεξεργασία πραγματοποιείται η εύρεση όλων των βιβλίων τα οποία έχουν εκδοθεί από το συγκεκριμένο συγγραφέα και η δημιουργία μιας λίστας με τα βιβλία αυτά. Τέλος, σαν έξοδο έχουμε την επιστροφή της λίστας αυτής με τα στοιχεία των βιβλίων τα οποία έχουν γραφτεί από το συγκεκριμένο συγγραφέα.

Επίσης, θα αναλύσουμε την περίπτωση όπου η εύρεση των βιβλίων γίνεται με βάση μία συγκεκριμένη χρονιά.



Σχήμα 4.22

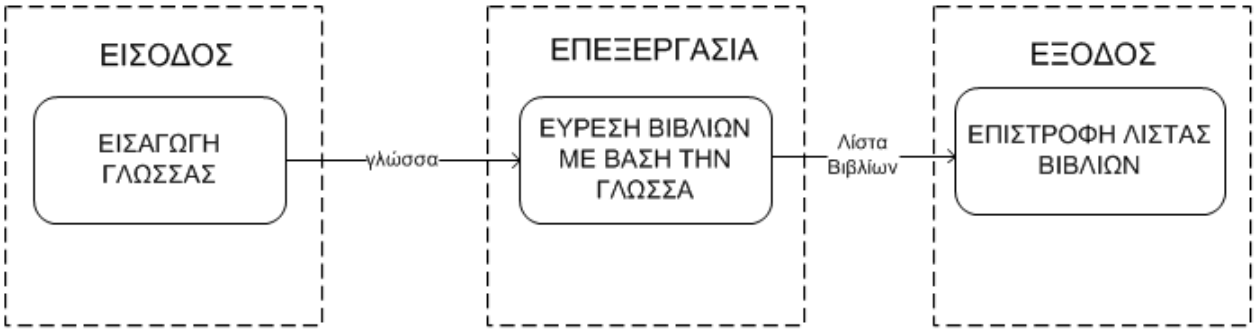
Το Σχήμα 4.22 αποτελεί το Data Flow Datagram για την υπηρεσία διαδικτύου Book List Service, η οποία είναι υπεύθυνη να εμφανίζει ένα κατάλογο με όλα τα βιβλία τα οποία πληρούν συγκεκριμένα χαρακτηριστικά. Στη περίπτωση αυτή, χρειάζεται να εμφανίσει μία λίστα με όλα τα βιβλία τα οποία έχουν εκδοθεί μία συγκεκριμένη χρονιά. Αρχικά, σαν είσοδος της λειτουργίας αυτής, γίνεται η εισαγωγή της υπο εξέταση χρονιάς. Έπειτα, ως επεξεργασία πραγματοποιείται η εύρεση όλων των βιβλίων τα οποία έχουν εκδοθεί την δεδομένη χρονιά και την δημιουργία μιας λίστας με τα βιβλία αυτά. Τέλος, σαν έξοδος μας παρέχεται η λίστα με τα στοιχεία των βιβλίων τα οποία αποφάνθηκε ότι έχουν εκδοθεί την χρονιά που δόθηκε σαν είσοδος.

Έπειτα θα αναλύσουμε την περίπτωση όπου η εύρεση των βιβλίων γίνεται με βάση μία κατηγορία βιβλίων.

Σχήμα 4.23

Το Σχήμα 4.23 απεικονίζει το Data Flow Datagram για την υπηρεσία διαδικτύου Book List Service. Η συγκεκριμένη υπηρεσία διαδικτύου δημιουργεί λίστες με βιβλία τα οποία πληρούν συγκεκριμένα χαρακτηριστικά. Στο Σχήμα 4.23 είναι απαραίτητη η εμφάνιση μιας λίστα με όλα τα βιβλία τα οποία ανήκουν σε μία συγκεκριμένη κατηγορία (πχ μυστηρίου, φαντασίας κλπ). Η κατηγορία η οποία εξετάζεται δίνεται σαν παράμετρος στην λειτουργία αυτή. Ακολούθως, επεξεργασία της λειτουργίας αποτελεί η εύρεση όλων των βιβλίων τα οποία αντιστοιχούν στην συγκεκριμένη κατηγορία και η δημιουργία μιας λίστας με τα βιβλία αυτά. Εν κατακλείδι, έξοδο της λειτουργίας αποτελεί η επιστροφή της λίστας αυτής με τα στοιχεία των βιβλίων τα οποία αντιστοιχούν στην συγκεκριμένη κατηγορία.

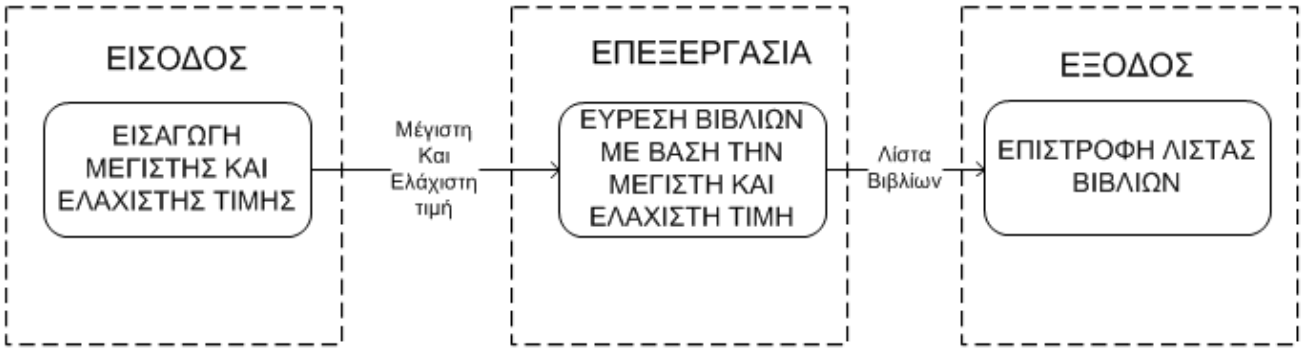
Στην συνέχεια θα αναλύσουμε την περίπτωση όπου η εύρεση των βιβλίων γίνεται με βάση την γλώσσα.



Σχήμα 4.24

Το Σχήμα 4.24 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου Book List Service. Η υπηρεσία διαδικτύου είναι επίσης υπεύθυνη να εμφανίζει ένα κατάλογο με όλα τα βιβλία τα οποία είναι γραμμένα σε συγκεκριμένη γλώσσα. Η λειτουργία της εύρεσης όλων των βιβλίων με βάση την γλώσσα περιέχει αρχικά σαν είσοδο την εισαγωγή της γλώσσας. Στην συνέχεια, σαν επεξεργασία, έχουμε την εύρεση όλων των βιβλίων τα οποία είναι γραμμένα στην συγκεκριμένη γλώσσα και την δημιουργία μιας λίστας με τα βιβλία αυτά. Έπειτα, σαν έξοδο έχουμε την εμφάνιση της λίστας αυτής με τα στοιχεία των βιβλίων τα οποία είναι γραμμένα στην συγκεκριμένη γλώσσα.

Εν κατακλείδι, θα διερευνηθεί η περίπτωση όπου η εύρεση των βιβλίων γίνεται με βάση μία συγκεκριμένη τιμή.

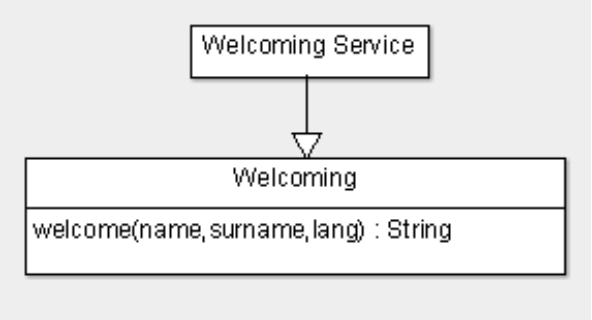


Σχήμα 4.25

Το Σχήμα 4.25 είναι το Data Flow Datagram για την υπηρεσία διαδικτύου Book List Service. Στην συγκεκριμένη περίπτωση, η υπηρεσία διαδικτύου καλείται να εμφανίσει μία λίστα με όλα τα βιβλία τα οποία βρίσκονται ανάμεσα σε κάποιο συγκεκριμένο εύρος τιμών. Η λειτουργία της εύρεσης όλων των βιβλίων με βάση την τιμή τους περιέχει σαν είσοδο την εισαγωγή της μέγιστης και ελάχιστης τιμής που θέλουμε να χρησιμοποιήσουμε σαν έλεγχο. Έπειτα, ως επεξεργασία, γίνεται η εύρεση όλων των βιβλίων τα οποία αντιστοιχούν στο εύρος τιμών ανάμεσα στην μέγιστη και τη ελάχιστη τιμή και την δημιουργία μιας λίστας με τα βιβλία αυτά. Σαν έξοδος, εκτελείται η επιστροφή της λίστας αυτής με τα στοιχεία των βιβλίων τα οποία αντιστοιχούν στο συγκεκριμένο εύρος τιμών.

4.4 Διαγράμματα Κλάσεων

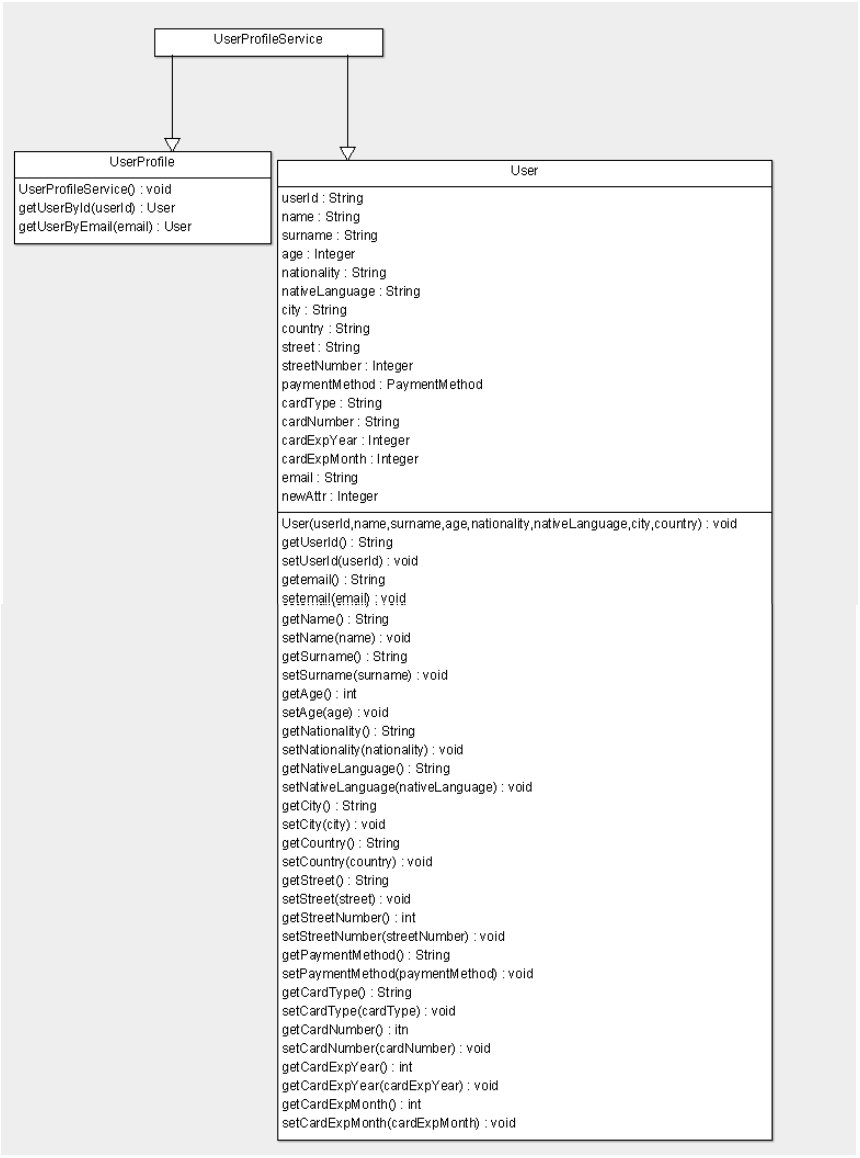
4.4.1 Welcoming Service



Σχήμα 4.26

Το Σχήμα 4.26 παρουσιάζει το Διάγραμμα Κλάσης της υπηρεσίας διαδικτύου Welcoming Service. Η υπηρεσία διαδικτύου Welcoming Service περιέχει την κλάση Welcoming η οποία έχει μόνο μία μέθοδο την welcome. Η συγκεκριμένη μέθοδος παίρνει τρεις παραμέτρους το όνομα του χρήστη, το επίθετο του χρήστη και την γλώσσα του, δηλαδή name, surname και language. Επιπρόσθετα η μέθοδος αυτή επιστρέφει το μήνυμα καλωσορίσματος με το όνομα του χρήστη στην τοπική γλώσσα.

4.4.2 User Profile Service



Σχήμα 4.27

Το Σχήμα 4.27 αποτελεί το Διάγραμμα Κλάσης της υπηρεσίας διαδικτύου User Profile Service. Η υπηρεσία διαδικτύου User Profile Service περιέχει δύο κλάσεις.

Η πρώτη κλάση της υπηρεσίας διαδικτύου είναι η User Profile, η οποία απαρτίζεται από τρεις μεθόδους. Η UserProfileService, είναι η πρώτη μέθοδος της συγκεκριμένης κλάσης και αποτελεί ένα κατασκευαστή για αυτή. Μέσα στον constructor αρχικοποιούμε την λίστα με τους χρήστες μας. Έπειτα, η κλάση περιέχει τη μέθοδο getUserById, η οποία παίρνει σαν παράμετρο ένα userId. Η συγκεκριμένη μέθοδος είναι υπεύθυνη για να επιστρέφει τον χρήστη με το συγκεκριμένο id. Τέλος, η μέθοδος getUserByEmail η οποία παίρνει σαν

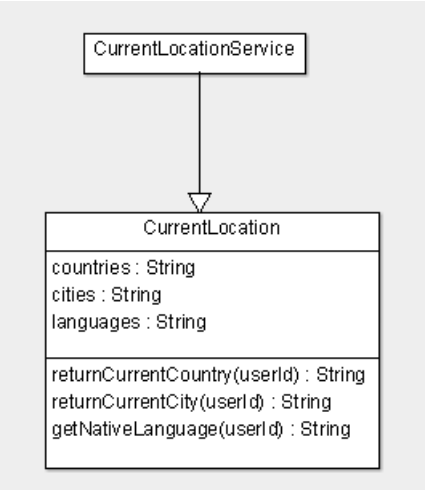
παράμετρο ένα email. Η μέθοδος αυτή είναι υπεύθυνη για να επιστρέφει το χρήστη με το συγκεκριμένο email.

Ακολουθεί η κλάση Users, η οποία περιέχει όλα τα χαρακτηριστικά των χρηστών του συστήματος. Η κλάση αυτή περιέχει 16 μεταβλητές και 33 μεθόδους. Οι μεταβλητές είναι userId, name, surname, age, nationality, nativeLanguage, city, country, street, streetNumber, paymentMethod, cardType, cardNumber, cardExpYear, cardExpMonth και email.

Κάθε μία από αυτές τις μεταβλητές έχει τον δικό της τύπο αλλά είναι δηλωμένη σαν private. Οι 32 από τις 33 μεθόδους είναι set και get μεθόδοι για τις μεταβλητές μας. Για παράδειγμα, για το email έχουμε 2 μεθόδους με ονόματα String getemail και Setemail. Αξίζει να σημειωθεί ότι κάθε μέθοδος τύπου set παίρνει σαν παράμετρο την τιμή της μεταβλητής που θέλει να ορίσει. Για παράδειγμα, η Setemail παίρνει σαν παράμετρο μία μεταβλητή email.

Τέλος, η τελευταία μέθοδος της κλάσης αυτής είναι ο constructor της κλάσης Users ο οποίος χρησιμοποιείται για να αρχικοποιήσει με τιμές όλες τις μεταβλητές της κλάσης Users την στιγμή που δημιουργείται ένα νέο αντικείμενο αυτής της κλάσης. Η μέθοδος η οποία αποτελεί τον constructor της κλάσης Users παίρνει σαν παραμέτρους μεταβλητές των οποίων οι τιμές θα ανατεθούν στις μεταβλητές τις κλάσης.

4.4.3 Current Location Service



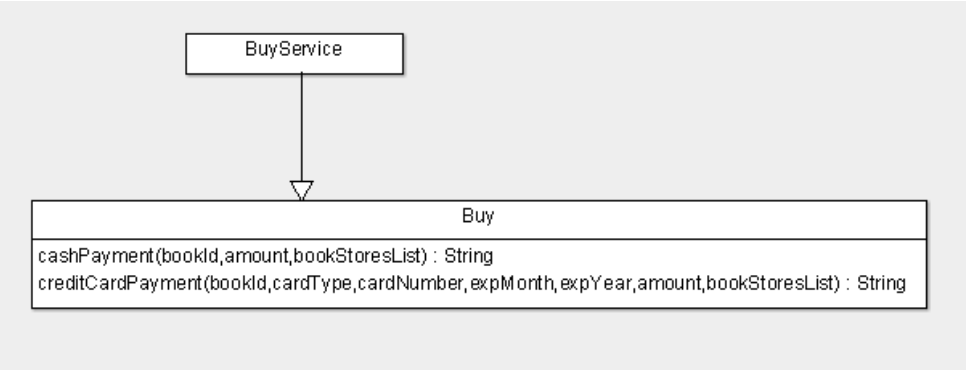
Σχήμα 4.28

Το Σχήμα 4.28 απεικονίζει το Διάγραμμα Κλάσης της υπηρεσίας διαδικτύου Current Location Service. Η υπηρεσία διαδικτύου Current Location Service περιέχει μία κλάση, η οποία ονομάζεται Current Location.

Η κλάση αυτή περιέχει 3 μεταβλητές. Αρχικά, υπάρχει η μεταβλητή countries η οποία είναι ένας πίνακα τύπου String στον οποίο αποθηκεύονται οι χώρες. Ακολούθως, υπάρχει η μεταβλητή cities η οποία είναι και πάλι ένας πίνακα τύπου String στον οποίο αποθηκεύονται οι πόλεις. Τέλος, υπάρχει η μεταβλητή languages η οποία είναι πίνακας τύπου String στον οποίο αποθηκεύονται οι γλώσσες.

Επιπρόσθετα, η κλάση αυτή περιέχει τρεις μεθόδους. Η πρώτη μέθοδος είναι η returnCurrentCountry, η οποία παίρνει σαν παράμετρο το userId. Η μέθοδος αυτή είναι υπεύθυνη να επιστρέψει μία τυχαία χώρα από τον πίνακα countries σαν τη χώρα στην οποία βρίσκεται την δεδομένη στιγμή ο χρήστης με το συγκεκριμένο userId. Η δεύτερη μέθοδος που υποστηρίζεται από την συγκεκριμένη κλαση είναι η μέθοδος returnCurrentCity, η οποία παίρνει σαν παράμετρο το userId. Η μέθοδος αυτή είναι υπεύθυνη να επιστρέψει μία τυχαία πόλη από τον πίνακα cities σαν την πόλη στην οποία βρίσκεται την δεδομένη στιγμή ο χρήστης με το συγκεκριμένο userId. Τέλος, η μέθοδος returnNativeLanguage παίρνει σαν παράμετρο το userId. Η μέθοδος αυτή είναι υπεύθυνη να επιστρέψει μία τυχαία γλώσσα από τον πίνακα languages σαν η τοπική γλώσσα για την περιοχή στην οποία βρίσκεται την δεδομένη στιγμή ο χρήστης με το συγκεκριμένο userId.

4.4.4 Buy Service



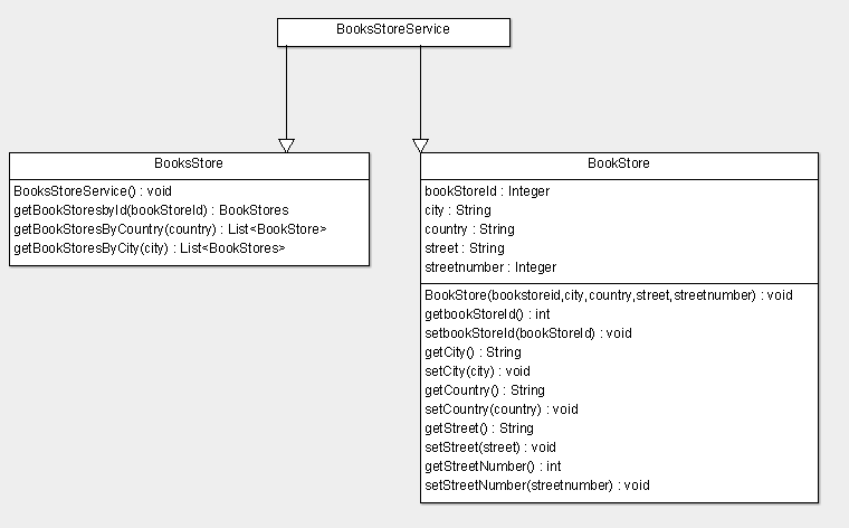
Σχήμα 4.29

Το Σχήμα 4.29 απεικονίζει το Διάγραμμα Κλάσης της υπηρεσίας διαδικτύου Buy Service. Η υπηρεσία διαδικτύου Buy Service περιέχει μία κλάση, η οποία ονομάζεται Buy.

Η κλάση αυτή αποτελείται από δύο μεθόδους. Η πρώτη μέθοδος είναι η cashPayment, η οποία παίρνει σαν παράμετρο το bookId, το amount και το bookStoresList. Η μέθοδος αυτή είναι υπεύθυνη να επιστρέψει ένα μήνυμα το οποίο αποτελεί απόδειξη για την πληρωμή του χρήστη με μετρητά για το συγκεκριμένο ποσό, το οποίο δίνεται σαν παράμετρος. Επίσης εμφανίζει το μήνυμα μαζί με την λίστα από τα βιβλιοπωλεία από όπου μπορεί να προμηθευτεί ο χρήστης την αγορά του.

Η δεύτερη μέθοδος είναι η creditCardPayment και παίρνει σαν παραμέτρους το bookId, το cardType, το cardNumber, το expMonth, το expYear, το amount και το bookStoresList. Η μέθοδος αυτή αρχικά ελέγχει την χρονιά λήξης της πιστωτικής κάρτας. Στην συνέχεια, ελέγχει το μήνα λήξης για να αποφανθεί εάν η πιστωτική κάρτα είναι έγκυρη. Εάν η κάρτα κριθεί έγκυρη εκδίδεται ένα μήνυμα το οποίο αποτελεί απόδειξη της πληρωμής του χρήστη για το ποσό, το οποίο δίνεται σαν παράμετρος.

4.4.5 Books Store Service



Σχήμα 4.30

Το Σχήμα 4.30 απεικονίζει το Διάγραμμα Κλάσης της υπηρεσίας διαδικτύου Books Store Service. Η υπηρεσία διαδικτύου Books Store Service περιέχει δύο κλάσεις, οι οποίες ονομάζονται BooksStore και BookStore.

Αρχικά έχουμε την κλάση BooksStore, η οποία περιέχει 4 μεθόδους. Πρώτη μέθοδος είναι η κλάση BooksStoreService που είναι ο constructor της συγκεκριμένης κλάσης. Στην μέθοδο αυτή αρχικοποιείται ένας πίνακας με όλα τα βιβλιοπωλεία τα οποία περιέχονται στο σενάριο του Online Book Store.

Η δεύτερη μέθοδος είναι η getBookStoreById, η οποία πάιρνει σαν παράμετρο ένα BookStoreId. Η μέθοδος αυτή επιστρέφει μια μεταβλητή η οποία αντιστοιχεί στην εγγραφή του βιβλιοπωλείου με το συγκεκριμένο BookStoreId.

Στην συνέχεια έχουμε την μέθοδο getBookStoreByCountry η οποία παίρνει σαν παράμετρο το όνομα μίας χώρας. Η συγκεκριμένη μέθοδος είναι υπεύθυνη να εμφανίζει μία λίστα με όλα τα διαθέσιμα βιβλιοπωλεία στην συγκεκριμένη χώρα.

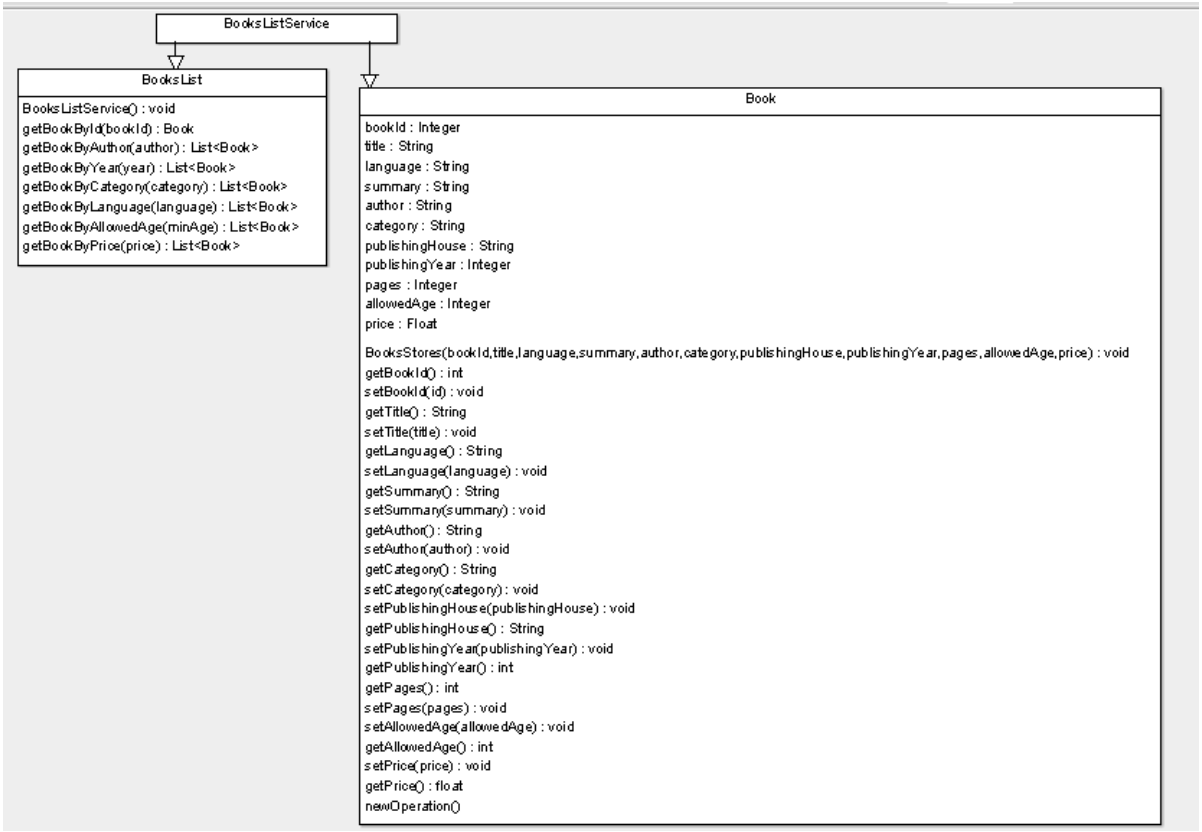
Τέλος έχουμε την μέθοδο getBookStoreByCity η οποία παίρνει σαν παράμετρο το όνομα μίας πόλης. Η μέθοδος αυτή είναι υπεύθυνη να εμφανίζει μία λίστα με όλα τα διαθέσιμα βιβλιοπωλεία στην συγκεκριμένη πόλη.

Η δεύτερη κλάση της υπηρεσίας διαδικτύου Books Store Service είναι η BookStore η οποία περιέχει 5 μεταβλητές και 11 μεθόδους. Οι μεταβλητές οι οποίες περιέχονται στην κλάση αυτή είναι οι bookStoreId, city, country, street, streetnumber. Η μεταβλητή bookStoreId αποτελεί ένα μοναδικό χαρακτηριστικό κάθε βιβλιοπωλείου. Επίσης, η μεταβλητή city, αναφέρεται στην πόλη στην οποία βρίσκεται κάθε βιβλιοπωλείο, ενώ η μεταβλητή country,

περιγράφει την χώρα στην οποία βρίσκεται το κάθε βιβλιοπωλείο. Ακόμη, υπάρχει η μεταβλητή street, η οποία αντιστοιχεί στην οδό στην οποία βρίσκεται το κάθε βιβλιοπωλείο και η μεταβλητή streetnumber, η οποία αναφέρεται στον αριθμό της οδού στην οποία βρίσκεται το συγκεκριμένο βιβλιοπωλείο.

Όπως προαναφέρθηκε η κλάση BookStore περιέχει έντεκα μεθόδους. Αρχικά, υπάρχει η μέθοδος BooksStores η οποία παίρνει σαν παράμετρο τις μεταβλητές bookstoreid, city, country, street και streetnumber. Η μέθοδος αυτή είναι υπεύθυνη να αρχικοποιήσει την τιμή κάθε μεταβλητής της συγκεκριμένης κλάσης με την αντίστοιχη παράμετρο. Οι υπόλοιπες έντεκα μέθοδοι, αποτελούν μεθόδους τύπου get και set για κάθε μεταβλητή της συγκεκριμένη κλάσης. Για παράδειγμα για την μεταβλητή city έχουμε τις μεθόδους getCity και setCity. Η μέθοδος getCity είναι υπεύθυνη να επιστρέφει την τιμή της μεταβλητής City για το συγκεκριμένο αντικείμενο της κλάσης. Επίσης η μέθοδος setCity παίρνει σαν παράμετρο το όνομα μίας πόλης και είναι υπεύθυνη να θέτει την τιμή της συγκεκριμένης μεταβλητής για το αντικείμενο από το οποίο κλήθηκε.

4.4.6 Books List Service



Σχήμα 4.31

Το Σχήμα 4.31 απεικονίζει το Διάγραμμα Κλάσης της υπηρεσίας διαδικτύου Book List Service. Η υπηρεσία διαδικτύου Book List Service περιέχει δύο κλάσεις, οι οποίες ονομάζονται BooksList και Book.

Αρχικά, η κλάση BooksList, περιέχει 8 μεθόδους. Η μέθοδος BooksListService είναι ο constructor της συγκεκριμένης κλάσης και με αυτή αρχικοποιείται ένας πίνακας με όλα τα βιβλία τα οποία είναι διαθέσιμα στα βιβλιοπωλεία τα οποία περιέχονται στην BookStoreService.

Στην συνέχεια, υπάρχει η μέθοδος getBookById, η οποία παίρνει σαν παράμετρο ένα bookId. Η συγκεκριμένη μέθοδος είναι υπεύθυνη να εντοπίσει και να επιστρέψει το βιβλίο με το συγκεκριμένο bookId.

Επιπλέον, η μέθοδο getBookByAuthor παίρνει σαν παράμετρο το όνομα ενός συγγραφέα. Η συγκεκριμένη μέθοδος είναι υπεύθυνη να εμφανίζει μία λίστα με όλα τα βιβλία τα οποία είναι διαθέσιμα στα βιβλιοπωλεία και έχουν γραφτεί από τον συγγραφέα του οποίου το όνομα δόθηκε σαν παράμετρος.

Ακολουθεί η μέθοδος getBookByYear, η οποία παίρνει σαν παράμετρο μία χρονιά. Η συγκεκριμένη μέθοδος είναι υπεύθυνη να εμφανίζει μία λίστα με όλα τα βιβλία τα οποία εκδόθηκαν την συγκεκριμένη χρονιά.

Έπειτα, η μέθοδος getBookByCategory με παράμετρο το όνομα μίας κατηγορίας. Η συγκεκριμένη μέθοδος είναι υπεύθυνη να εμφανίζει μία λίστα με όλα τα βιβλία τα οποία είναι διαθέσιμα στα βιβλιοπωλεία και ανήκουν στην συγκεκριμένη κατηγορία, η οποία δόθηκε σαν παράμετρος.

Επιπρόσθετα, η μέθοδος getBookByLanguage παίρνει σαν παράμετρο το όνομα μίας γλώσσας. Η συγκεκριμένη μέθοδος είναι υπεύθυνη να εμφανίζει μία λίστα με όλα τα βιβλία τα οποία είναι διαθέσιμα στα βιβλιοπωλεία και προσφέρονται στην γλώσσα η οποία δηλώθηκε ως παράμετρος.

Ακόμα, η μέθοδος getBookByAllowedAge παίρνει σαν παράμετρο μια ηλικία. Η συγκεκριμένη μέθοδος είναι υπεύθυνη να βρει και να επιστρέψει όλα τα βιβλία τα οποία αντιστοιχούν στην ηλικία η οποία δόθηκε σαν παράμετρος.

Τέλος η μέθοδος getBookByPrice, η οποία παίρνει σαν παράμετρο μία συγκεκριμένη τιμή. Η συγκεκριμένη μέθοδος είναι υπεύθυνη να εμφανίζει μία λίστα με όλα τα βιβλία τα οποία είναι διαθέσιμα στα βιβλιοπωλεία των οποίων η τιμή δεν ξεπερνά την τιμή που δόθηκε σαν παράμετρος.

Ακολουθεί ανάλυση της κλάσης Books, η οποία περιέχει 11 μεταβλητές και 23 μεθόδους. Οι μεταβλητές οι οποίες περιέχονται στην κλάση αυτή είναι οι bookId, title, language, summary, author, category, publishingHouse, publishingYear, pages, allowedAge και price. Η μεταβλητή bookId αποτελεί ένα μοναδικό χαρακτηριστικό για κάθε βιβλίο, ενώ η μεταβλητή title περιγράφει τον τίτλο του βιβλίου. Έπειτα, υπάρχει η μεταβλητή language, η οποία αναφέρεται στην γλώσσα στην οποία είναι γραμμένο το βιβλίο καθώς και η μεταβλητή summary που αναφέρεται σε μία συνοπτική περίληψη του βιβλίου. Η μεταβλητή author αναφέρεται στον συγγραφέα του βιβλίου. Ακόμα η μεταβλητή category, αναφέρεται στην κατηγορία στην οποία ανήκει το βιβλίο και η μεταβλητή publishingHouse, που αναφέρεται στον εκδοτικό οίκο του βιβλίου. Επιπρόσθετα υπάρχει η μεταβλητή publishingYear, η οποία αναφέρεται στην χρονιά που εκδόθηκε το βιβλίο, καθώς και η μεταβλητή pages, η οποία αναφέρεται στον αριθμό των σελίδων που απαρτίζουν το βιβλίο. Ακολούθως, υπάρχει η μεταβλητή allowedAge, που αναφέρεται στην επιτρεπόμενη ηλικία που πρέπει να έχει ένας χρήστης για να διαβάσει το συγκεκριμένο βιβλίο. Τέλος, υπάρχει η μεταβλητή price που είναι η τιμή του βιβλίου.

Η κλάση Books περιέχει είκοσι τρείς μεθόδους. Αρχικά, υπάρχει η μέθοδος Books η οποία παίρνει σαν παράμετρο τις bookId, title, language, summary, author, category, publishingHouse, publishingYear, pages, allowedAge και price. Η μέθοδος αυτή είναι υπεύθυνη να αρχικοποιεί την τιμή κάθε μεταβλητής της συγκεκριμένης κλάσης με την αντίστοιχη παράμετρο. Οι υπόλοιπες είκοσι δύο μέθοδοι αποτελούν μεθόδους τύπου get και set για κάθε μεταβλητή της συγκεκριμένη κλάσης. Για παράδειγμα, για την μεταβλητή city έχουμε τις μεθόδους getTitle και setTitle. Η μέθοδος getTitle είναι υπεύθυνη να επιστρέφει την τιμή της μεταβλητής Title για το συγκεκριμένο αντικείμενο της κλάσης. Επίσης, η μέθοδος setTitle παίρνει σαν παράμετρο τον τίτλο του βιβλίου και είναι υπεύθυνη να θέτει την τιμή της συγκεκριμένης μεταβλητής για το αντικείμενο από το οποίο κλήθηκε.

Κεφάλαιο 5

Συμπεράσματα

5.1 Προστασία προσωπικών δεδομένων στις υπηρεσίες διαδικτύου 66

5.1 Προστασία προσωπικών δεδομένων στις υπηρεσίες διαδικτύου

Στις μέρες μας, η ασφάλεια των πληροφοριών και ειδικότερα η ασφάλεια των προσωπικών μας πληροφοριών, είναι ιδιαίτερα σημαντική και αποτελεί για τον καθένα από εμάς αυτονόητη προϋπόθεση. Σε πολλές περιπτώσεις, παραχωρούμε τα προσωπικά μας δεδομένα και εμπιστευόμαστε τους παραλήπτες να τα αξιοποιήσουν μόνο για συγκεκριμένες εργασίες.
Ωστόσο, με το διαδίκτυο η ασφάλεια των προσωπικών μας δεδομένων έχει γίνει μία αφαιρετική έννοια καθώς οποιοσδήποτε μπορεί να διεισδύσει στα αρχεία κάποιας εταιρίας και να υποκλέψει τις προσωπικές μας πληροφορίες. Σε αρκετές περιπτώσεις μάλιστα η παραβίαση των προσωπικών μας δεδομένων μπορεί γίνει και από τους ίδιους τους ανθρώπους στους οποίους εμπιστευτήκαμε τα προσωπικά μας δεδομένα.
Σήμερα, με τις υπηρεσίες διαδικτύου (web services) το πρόβλημα παραμένει. Στην παρούσα ατομική διπλωματική εργασία δόθηκε ένας τρόπος εξασφάλισης της χρήσης των προσωπικών δεδομένων του χρήστη με τον τρόπο που αυτός επιθυμεί. Η επέκταση της υφιστάμενης αρχιτεκτονικής για την εισαγωγή των προτιμήσεων του χρήστη σε σχέση με την διαχείριση των στοιχείων του έγινε ακριβώς για να δίνει την δυνατότητα σε κάθε άτομο να επιλέγει πως θα χρησιμοποιηθούν και εάν θα χρησιμοποιηθούν τα στοιχεία του.
Η επέκταση της υφιστάμενης αρχιτεκτονικής, όπως φάνηκε και από την εφαρμογή του σεναρίου, διασφαλίζει την ιδιωτικότητα τόσο των στοιχείων των χρηστών όσο και των προτιμήσεών τους για την χρήση των στοιχείων αυτών.

Βιβλιογραφία

[1] XML, [Online]. Available: http://en.wikipedia.org/wiki/XML

[2] XML (Extensible Markup Language), [Online]. Available: http://searchsoa.techtarget.com/definition/XML

[3] XML Schema, [Online]. Available: http://en.wikipedia.org/wiki/XML_schema

[4] Κ. Μαργαρίτης, Θεοχάρης Δημητρίου, Web Services και SOAP, Πανεπιστήμιο Μακεδονίας, Τμήμα Εφαρμοσμένης Πληροφορικής, Θεσσαλονίκη, Μάρτιος 2007

[5] Service Oriented Architecture, [Online]. Available: http://en.wikipedia.org/wiki/Service-oriented_architecture

[6] SOAP, [Online]. Available: http://en.wikipedia.org/wiki/SOAP_(protocol)

[7] Universal Description Discovery and Integration, [Online]. Available: http://en.wikipedia.org/wiki/Universal_Description,_Discovery,_and_Integration

[8] Web Services Description Language, [Online]. Available: http://en.wikipedia.org/wiki/Web_Services_Description_Language

[9] Υπηρεσίες Διαδικτύου 07/15/2012, [Online]. Available: (https://docs.google.com/viewer?a=v&q=cache:mQBp51M2TN8J:150.140.9.29/intech/files/Web%2520Services_08_09_0.ppt+web+services+%CE%BF%CF%81%CE%B9%CF%83%CE%BC%CF%8C%CF%82&hl=el&gl=cy&pid=bl&srcid=ADGEESiWzpH4A-q3AwM2-QoO-ji7IksuOgAsd4jfOrLaoqbBvtKOLMF2VgboBRNYPPBv4o2ruRkZy14MqOKETiVZIhkZEpCyechmGSd3zHH4TYYVyT0pXklhR2yOhbUyIlUV93VqFlRF&sig=AHIEtbQNwNWqA54BTFhctYQ_D5aXFmMcYQ)

[10] Anind K. Dey and Gregory D. Abowd, Towards a Better Understanding of Context and Context-Awareness

[11] User Context, [Online]. Available: https://workinggroups.wiki.dlorg.eu/index.php/User_Context#User_Context_Definition

[12] Dey, K., "Understanding and Using Context", Personal and Ubiquitous Computing

[13] Apache Axis 2, [Online]. Available: http://en.wikipedia.org/wiki/Apache_Axis2

[14] Καλλονιάτης Χρήστος, ΑΣΦΑΛΕΙΑ ΔΕΔΟΜΕΝΩΝ ΣΤΗΝ ΚΟΙΝΩΝΙΑ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ (Ιδιωτικότητα)

[15] [Online]. Available: http://hitachi-id.com/concepts/authentication.html

[16] Authentication, [Online]. Available: http://searchsecurity.techtarget.com/definition/authentication

[17] Authorization, [Online]. Available: http://searchsoftwarequality.techtarget.com/definition/authorization

[18] What is Personal Data? [Online]. Available: http://www.dataprotection.ie/viewdoc.asp?docid=210

[19] Sensitive Personal Information Law & Legal Definition [Online]. Available: http://definitions.uslegal.com/s/sensitive-personal-information/

[20] Web Services Protocols, [Online]. Available: http://booster911.hubpages.com/hub/Web-Services-Protocols

[21] Shi, Xuan. Sharing Service Semantics using SOAP-Based and REST Web Services. March/April 2006

[22] Leon Shklar, Richard Rosen. Web Application Architecture - Principles, Protocols and Practices. s.l. :John Wiley & Sons Ltd, 2003.

[23] Εισαγωγή στα Web Services, [Online]. Available: http://www.it.uom.gr/project/soap/Theory/introduction.html

[24] Georgia M. Kapitsaki, Dimitrios A. Kateros, Iakovos S. Venieris, Architecture for Provision of Context – aware Web Applications based on Web Services

[25] What is Axis2?, [Online]. Available: http://javajazzup.com/issue11/page13.shtml

[26] Apache CXF, [Online]. Available: http://en.wikipedia.org/wiki/Apache_CXF

[27] SOAP – WSDL - UDDI, [Online]. Available: http://www.slideshare.net/PeterREgli/soap-wsdl-uddi

[28] Eclipse IDE for Java EE Developers, [Online]. Available: http://www.eclipse.org/downloads/moreinfo/jee.php

[29] Apache Tomcat, [Online]. Available: http://en.wikipedia.org/wiki/Apache_Tomcat

[30] Document Object Model (DOM), [Online]. Available: http://searchwindevelopment.techtarget.com/definition/Document-Object-Model

[31] Should I use SAX or DOM?, [Online]. Available: http://developerlife.com/tutorials/?p=28#88116

Παράρτημα Α

Handler.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<handler:plugins xmlns:handler="http://www.contextplugins.org/handler"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.contextplugins.org/handler handler.xsd">

	<handler:plugin id="ad1" pckg="plugin.welcoming" classname="WelcomePlugin"
jarfile="WelcomePlugin.jar" />
	<handler:plugin id="ad2" pckg="plugin.bookslist"
classname="BooksListPlugin" jarfile="BooksListPlugin.jar" />
	<handler:plugin id="ad3" pckg="plugin.buying" classname="BuyPlugin"
jarfile="BuyPlugin.jar" />

	<handler:association>
		<handler:serviceEnd name="WelcomingService" operation="welcome" />
		<handler:pluginEnd id="ad1" direction="in" />
	</handler:association>

	<handler:association>
		<handler:serviceEnd name="BooksListService"
operation="getBooksByAllowedAge" />
		<handler:pluginEnd id="ad2" direction="in" />
	</handler:association>

	<handler:association>
		<handler:serviceEnd name="BuyService" operation="cashPayment" />
		<handler:pluginEnd id="ad3" direction="in" />
	</handler:association>

	<handler:association>
		<handler:serviceEnd name="BuyService" operation="creditCardPayment"
/>
		<handler:pluginEnd id="ad3" direction="in" />
	</handler:association>

</handler:plugins>
```

PrivacyPreferences.xml

```
package parser;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Properties;

import org.apache.xmlbeans.XmlObject;

import cy.books.xsdobj.ConditionType;
import cy.books.xsdobj.DataCategoryT;
import cy.books.xsdobj.PrivacyPreferencesDocument;
import cy.books.xsdobj.UserPrivacyPreferences;
import cy.books.xsdobj.UserPrivacyPreferences.Allow;
import cy.books.xsdobj.UserPrivacyPreferences.Block;

public class PrivacyPreferences {

	protected XmlObject messageRoot;
	protected XmlObject messageObject;
	protected String msgCode;

	private String TAG = "[UPP]";
	private boolean isEnabled = false;

	private Allow[] allowArray;
	private Block[] blockArray;

	public PrivacyPreferences(String userId) {
		PrivacyPreferencesDocument root =
PrivacyPreferencesDocument.Factory.newInstance();
		messageRoot = root;
		messageObject = root.addNewPrivacyPreferences();
		msgCode = "privacyPreferences";

		String m_FilePath = "/cfg.properties";
		InputStreamReader is = new
InputStreamReader(this.getClass().getResourceAsStream(m_FilePath));
		BufferedReader in = new BufferedReader(is);
	Properties cfgproperties = new Properties();
	try {
		cfgproperties.load(in);
	} catch (IOException e) {
		e.printStackTrace();
	}
	String uppLocation = cfgproperties.getProperty("pluginslocation") + "upp/";

		try {
			File file = new File(uppLocation + userId + ".xml");
			PrivacyPreferencesDocument ppd =
PrivacyPreferencesDocument.Factory.parse(file);
			messageObject = ppd.getPrivacyPreferences();
			UserPrivacyPreferences p = (UserPrivacyPreferences)
messageObject;
```

```
                        allowArray = p.getAllowArray();
                        blockArray = p.getBlockArray();
                        isEnabled = true;
                } catch (FileNotFoundException fnfe) {
                        System.out.println(TAG + " Cannot find upp file for " + userId
+ ": " + fnfe.getMessage());
                } catch (Exception e) {
                        System.out.println(TAG + " Exception: " + e.getMessage());
                }
        }

        public boolean isEnabled() {
                return isEnabled;
        }

        public boolean isPersonalInfoFieldAllowed(String dataName) {
                for (Allow field : allowArray) {
                        if (field.getDataEnd().getDataCategory() ==
DataCategoryT.PERSONAL_INFO) {
                                if
(field.getDataEnd().getDataName().equalsIgnoreCase(dataName)) {
                                        System.out.println(String.format("%s PersonalInfo
for %s is allowed.", TAG, dataName));
                                        return true;
                                }
                        }
                }
                System.out.println(String.format("%s PersonalInfo for %s is not
allowed.", TAG, dataName));
                return false;
        }

        public boolean isPersonalInfoFieldBlocked(String dataName) {
                for (Block field : blockArray) {
                        if (field.getDataEnd().getDataCategory() ==
DataCategoryT.PERSONAL_INFO) {
                                if
(field.getDataEnd().getDataName().equalsIgnoreCase(dataName)) {
                                        System.out.println(String.format("%s PersonalInfo
for %s is blocked.", TAG, dataName));
                                        return true;
                                }
                        }
                }
                System.out.println(String.format("%s PersonalInfo for %s is not
blocked.", TAG, dataName));
                return false;
        }

        public boolean isPersonalPreferencesFieldAllowed(String dataName) {
                for (Allow field : allowArray) {
                        if (field.getDataEnd().getDataCategory() ==
DataCategoryT.PERSONAL_PREFERENCES) {
                                if
(field.getDataEnd().getDataName().equalsIgnoreCase(dataName)) {
                                        System.out.println(String.format("%s
PersonalPreferences for %s is allowed.", TAG, dataName));
                                        return true;
                                }
                        }
                }
```

```
            System.out.println(String.format("%s PersonalPreferences for %s is
not allowed.", TAG, dataName));
            return false;
      }

      public boolean isPersonalPreferencesFieldBlocked(String dataName) {
            for (Block field : blockArray) {
                  if (field.getDataEnd().getDataCategory() ==
DataCategoryT.PERSONAL_PREFERENCES) {
                        if
(field.getDataEnd().getDataName().equalsIgnoreCase(dataName)) {
                              System.out.println(String.format("%s
PersonalPreferences for %s is blocked.", TAG, dataName));
                              return true;
                        }
                  }
            }
            System.out.println(String.format("%s PersonalPreferences for %s is
not blocked.", TAG, dataName));
            return false;
      }

      public boolean isLocationAllowed() {
            for (Allow field : allowArray) {
                  if (field.getDataEnd().getDataCategory() ==
DataCategoryT.LOCATION) {
                        System.out.println(String.format("%s Location is
allowed.", TAG));
                        return true;
                  }
            }
            System.out.println(String.format("%s Location is not allowed.",
TAG));
            return false;
      }

      public boolean isLocationBlocked() {
            for (Block field : blockArray) {
                  if (field.getDataEnd().getDataCategory() ==
DataCategoryT.LOCATION) {
                        System.out.println(String.format("%s Location is
blocked.", TAG));
                        return true;
                  }
            }
            System.out.println(String.format("%s Location is not blocked.",
TAG));
            return false;
      }

      public boolean validateAllowedLocationCondition(Calendar time, String
country, String city) {
            boolean valid = false;
            for (Allow field : allowArray) {
                  if (field.getDataEnd().getDataCategory() ==
DataCategoryT.LOCATION) {
                        ConditionType condition = field.getCondition();
                        if (condition == null) {
                              return true;
                        }
```

```
                        SimpleDateFormat formatter = new
SimpleDateFormat("EEEE");
                        String weekDay = formatter.format(time.getTime());
                        int currentWeekDay = getWeekDayNumber(weekDay);
                        int startWeekDay =
getWeekDayNumber(condition.getTimePeriodIn().getStartWeekDay());
                        int endWeekDay =
getWeekDayNumber(condition.getTimePeriodIn().getEndWeekDay());
                        // validate weekday
                        if (currentWeekDay >= startWeekDay && currentWeekDay <=
endWeekDay) {
                                System.out.println(String.format("%s Location
condition for weekday is valid.", TAG));
                                // validate time
                                Calendar timeStart =
condition.getTimePeriodIn().getTime().getStartTime();
                                Calendar timeEnd =
condition.getTimePeriodIn().getTime().getEndTime();
                                Calendar timeNow =
condition.getTimePeriodIn().getTime().getStartTime();
                                Calendar now = Calendar.getInstance();

                                timeNow.set(Calendar.HOUR_OF_DAY,
now.get(Calendar.HOUR_OF_DAY));
                                timeNow.set(Calendar.MINUTE,
now.get(Calendar.MINUTE));
                                timeNow.set(Calendar.SECOND,
now.get(Calendar.SECOND));

                                if (timeNow.compareTo(timeStart) >= 0 &&
timeNow.compareTo(timeEnd) <= 0) {
                                        System.out.println(String.format("%s
Location condition for time is valid.", TAG));
                                        // validate city and country
                                        boolean foundCity = false;
                                        for (String c :
condition.getSpaceLimitIn().getCityArray()) {
                                                if (c.equalsIgnoreCase(city)) {
                                                        foundCity = true;

        System.out.println(String.format("%s Location condition for city is
valid.", TAG));
                                                        break;
                                                }
                                        }
                                        boolean foundCountry = false;
                                        for (String c :
condition.getSpaceLimitIn().getCityArray()) {
                                                if (c.equalsIgnoreCase(city)) {
                                                        foundCountry = true;

        System.out.println(String.format("%s Location condition for country is
valid.", TAG));
                                                        break;
                                                }
                                        }
                                        if (foundCity && foundCountry) {
                                                valid = true;
                                        } else {
                                                System.out.println(String.format("%s
Location condition for space (country & city) is not valid.", TAG));
```

```
                                        }
                                }
                        } else {
                                System.out.println(String.format("%s Location
condition for weekday is not valid.", TAG));
                        }
                }
        }
        if (valid) {
                System.out.println(String.format("%s All Location conditions
are valid.", TAG));
        } else {
                System.out.println(String.format("%s Could not validate all
Location conditions.", TAG));
        }
        return valid;
    }

    private int getWeekDayNumber(String weekDay) {
        if (weekDay.equalsIgnoreCase("monday"))
            return 1;
        if (weekDay.equalsIgnoreCase("tuesday"))
            return 2;
        if (weekDay.equalsIgnoreCase("wednesday"))
            return 3;
        if (weekDay.equalsIgnoreCase("thursday"))
            return 4;
        if (weekDay.equalsIgnoreCase("friday"))
            return 5;
        if (weekDay.equalsIgnoreCase("saturday"))
            return 6;
        if (weekDay.equalsIgnoreCase("sunday"))
            return 7;

        return -1;
    }

}
```

BooksListPlugin.java

```
package plugin.bookslist;

import gr.ntua.icbnet.contextModule.plugins.AbstractRPCQueryPlugin;

import java.util.Iterator;

import javax.xml.namespace.QName;

import org.apache.axiom.om.OMAbstractFactory;
import org.apache.axiom.om.OMElement;
import org.apache.axiom.om.OMNode;
import org.apache.axiom.soap.SOAPEnvelope;
import org.apache.axiom.soap.SOAPFactory;
import org.apache.axis2.AxisFault;
import org.apache.axis2.context.MessageContext;

import parser.PrivacyPreferences;

public class BooksListPlugin extends AbstractRPCQueryPlugin {

	private String userId;
	private SOAPEnvelope soapEnv;
	private String servicesNamespace = "http://ws.books_service.cy";
	private String servicesUrl = "http://localhost:8080/axis2/services/";
	private String pluginTag = "[BooksListPlugin]";
	private PrivacyPreferences pp;

	public BooksListPlugin() {
		super();
	}

	public MessageContext modifySOAPEnvelope(MessageContext in) {
		soapEnv = in.getEnvelope();

		// get the userId header element attribute
		OMElement uidOE = soapEnv.getHeader().getFirstChildWithName(new
QName(servicesNamespace, "getBooksByAllowedAge"));
		if (uidOE != null && uidOE.getAttributeValue(new
QName(servicesNamespace, "userId")) != null) {
			userId = uidOE.getAttributeValue(new QName(servicesNamespace,
"userId"));
		}
		System.out.println(pluginTag + " ATTRIBUTE " + userId);

		pp = new PrivacyPreferences(userId);

		try {
			OMElement documentElement = (OMElement)
parseSOAPEnvelope(in.getEnvelope());
```

```
                        super.parseChildren(in.getEnvelope().getBody());
                        SOAPFactory fac = OMAbstractFactory.getSOAP11Factory();
                        SOAPEnvelope envelope = fac.getDefaultEnvelope();
                        Iterator<?> iter = documentElement.getChildren();
                        while (iter.hasNext()) {
                                OMNode child = (OMNode) iter.next();
                                if (envelope != null && envelope.getBody() != null && child
!= null) {
                                        envelope.getBody().addChild(child);
                                }
                        }
                        in.setEnvelope(envelope);
                } catch (Exception e) {
                        e.printStackTrace();
                }

                return in;
        }

        protected void parseChild(OMElement child) {
                try {
                        String localPart = child.getQName().getLocalPart();
                        // arg0: user age argument
                        if (localPart.compareToIgnoreCase("arg0") == 0) {
                                if (pp.isEnabled() && (pp.isPersonalInfoFieldAllowed("age") ||
!pp.isPersonalInfoFieldBlocked("age"))) {
                                        child.setText(String.valueOf(getUserAge(userId)));
                                }
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

        protected int getUserAge(String userId) throws AxisFault {
                setEndpointURL(servicesUrl + "UserProfileService");
                System.out.println(pluginTag + " Retrieving age for user " + userId);
                System.out.println(pluginTag + " Target Endpoint is: " + endpointURL);
                QName op = new QName(servicesNamespace, "getUserAge");
                Object[] opArgs = new Object[] { userId };
                Class[] opReturnTypes = new Class[] { int.class };
                Object[] response = invokeBlocking(op, opArgs, opReturnTypes);
                int result = (int) response[0];
                System.out.println(pluginTag + " age: " + result);
                return result;
        }

}
```

BuyPlugin.java

```
package plugin.buying;

import gr.ntua.icbnet.contextModule.plugins.AbstractRPCQueryPlugin;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import javax.xml.namespace.QName;

import org.apache.axiom.om.OMAbstractFactory;
import org.apache.axiom.om.OMElement;
import org.apache.axiom.om.OMNode;
import org.apache.axiom.soap.SOAPEnvelope;
import org.apache.axiom.soap.SOAPFactory;
import org.apache.axis2.AxisFault;
import org.apache.axis2.context.MessageContext;
import org.apache.axis2.databinding.utils.BeanUtil;
import org.apache.axis2.databinding.utils.MultirefHelper;
import org.apache.axis2.engine.DefaultObjectSupplier;

import parser.PrivacyPreferences;

public class BuyPlugin extends AbstractRPCQueryPlugin {

	private String userId;
	private String bookId;
	private SOAPEnvelope soapEnv;
	private String paymentMethod = "";
	private String preferredPaymentMethod = "";
	private String servicesNamespace = "http://ws.books_service.cy";
	private String servicesUrl = "http://localhost:8080/axis2/services/";
	private String pluginTag = "[BuyPlugin]";
	private PrivacyPreferences pp;

	public BuyPlugin() {
		super();
	}

	public MessageContext modifySOAPEnvelope(MessageContext in) {
		soapEnv = in.getEnvelope();

		// get the userId header element attribute
		OMElement uidOE = soapEnv.getHeader().getFirstChildWithName(new
QName(servicesNamespace, "cashPayment"));
		if (uidOE != null && uidOE.getAttributeValue(new
QName(servicesNamespace, "userId")) != null) {
			userId = uidOE.getAttributeValue(new QName(servicesNamespace,
"userId"));
```

```
                }
                System.out.println(pluginTag + " ATTRIBUTE " + userId);

                pp = new PrivacyPreferences(userId);

                // get the element containing the service method requested
                OMElement methodElem =
(OMElement)in.getEnvelope().getBody().getFirstOMChild();
                try {
                        // method argument
                        paymentMethod = methodElem.getQName().getLocalPart();

                        if (pp.isEnabled() &&
(pp.isPersonalPreferencesFieldAllowed("paymentMethod") &&
!pp.isPersonalPreferencesFieldBlocked("paymentMethod"))) {
                                PaymentMethod method = getUserPaymentMethod(userId);
                                if (method == PaymentMethod.CASH) {
                                        preferredPaymentMethod = "cashPayment";
                                } else if (method == PaymentMethod.CREDIT_CARD) {
                                        preferredPaymentMethod = "creditCardPayment";
                                }
                                System.out.println(pluginTag + " preferred payment method is
" + preferredPaymentMethod);
                        }

                        OMElement documentElement = (OMElement)
parseSOAPEnvelope(in.getEnvelope());
                        super.parseChildren(in.getEnvelope().getBody());
                        SOAPFactory fac = OMAbstractFactory.getSOAP11Factory();
                        SOAPEnvelope envelope = fac.getDefaultEnvelope();
                        Iterator<?> iter = documentElement.getChildren();
                        while (iter.hasNext()) {
                                OMNode child = (OMNode) iter.next();
                                if (envelope != null && envelope.getBody() != null && child
!= null) {
                                        envelope.getBody().addChild(child);
                                }
                        }
                        in.setEnvelope(envelope);
                } catch (Exception e) {
                        e.printStackTrace();
                }

                return in;
        }

        protected void parseChild(OMElement child) {
                try {
                        String localPart = child.getQName().getLocalPart();
                        if (localPart.compareToIgnoreCase("arg0") == 0) {
                                bookId = child.getText();
```

```
                        }
                        if (paymentMethod.equalsIgnoreCase("cashPayment")) {
                                // arg1: price
                                if (localPart.compareToIgnoreCase("arg1") == 0) {
                                        child.setText(String.valueOf(getBookPrice(bookId)));
                                }
                                // arg2: bookstores
                                else if (localPart.compareToIgnoreCase("arg2") == 0) {
                                        if (pp.isEnabled() && (
                                                        pp.isPersonalInfoFieldAllowed("city")
&& pp.isPersonalInfoFieldAllowed("country") &&
                                                        !pp.isPersonalInfoFieldBlocked("city")
&& !pp.isPersonalInfoFieldBlocked("country"))) {
                                                // get current city and country
                                                String currentCity =
getUserCurrentCity(userId);
                                                String currentCountry =
getUserCurrentCountry(userId);

                                                // search nearby bookstores by city
                                                List<BookStore> list =
getBookStoresByCity(currentCity);
                                                if (list.isEmpty()) {
                                                        // if there are no nearby bookstores,
search by country
                                                        list =
getBookStoresByCountry(currentCountry);
                                                }

                                                // create a string containing all the nearby
bookstores
                                                String str = "";
                                                for (BookStore bookStore : list) {
                                                        str += String.format("- %d %s, %s,
%s\n",

        bookStore.getStreetnumber(), bookStore.getStreetname(),
                                                                        bookStore.getCity(),
bookStore.getCountry());
                                                }
                                                if (str.isEmpty()) {
                                                        str = String.format("Sorry, no book
stores found nearby %s", currentCountry);
                                                }
                                                child.setText(str);
                                        }
                                }
                        } else if (paymentMethod.equalsIgnoreCase("creditCardPayment")) {
                                // arg1: user card type
                                if (localPart.compareToIgnoreCase("arg1") == 0) {
```

```
                                        if (pp.isEnabled() &&
(pp.isPersonalInfoFieldAllowed("cardType") &&
!pp.isPersonalInfoFieldBlocked("cardType"))) {
                                                child.setText(getUserCardType(userId));
                                        }
                                }
                                // arg2: user card number
                                else if (localPart.compareToIgnoreCase("arg2") == 0) {
                                        if (pp.isEnabled() &&
(pp.isPersonalInfoFieldAllowed("cardNumber") &&
!pp.isPersonalInfoFieldBlocked("cardNumber"))) {
                                                child.setText(getUserCardNumber(userId));
                                        }
                                }
                                // arg3: user card expiry month
                                else if (localPart.compareToIgnoreCase("arg3") == 0) {
                                        if (pp.isEnabled() &&
(pp.isPersonalInfoFieldAllowed("cardExpMonth") &&
!pp.isPersonalInfoFieldBlocked("cardExpMonth"))) {

        child.setText(String.valueOf(getUserCardExpMonth(userId)));
                                        }
                                }
                                // arg4: user card expiry year
                                else if (localPart.compareToIgnoreCase("arg4") == 0) {
                                        if (pp.isEnabled() &&
(pp.isPersonalInfoFieldAllowed("cardExpYear") &&
!pp.isPersonalInfoFieldBlocked("cardExpYear"))) {

        child.setText(String.valueOf(getUserCardExpYear(userId)));
                                        }
                                }
                                // arg5: book price
                                else if (localPart.compareToIgnoreCase("arg5") == 0) {
                                        child.setText(String.valueOf(getBookPrice(bookId)));
                                }
                                // arg6: bookstores
                                else if (localPart.compareToIgnoreCase("arg6") == 0) {
                                        if (pp.isEnabled() && (
                                                        pp.isPersonalInfoFieldAllowed("city")
&& pp.isPersonalInfoFieldAllowed("country") &&
                                                        !pp.isPersonalInfoFieldBlocked("city")
&& !pp.isPersonalInfoFieldBlocked("country"))) {
                                                // get current city and country
                                                String currentCity =
getUserCurrentCity(userId);
                                                String currentCountry =
getUserCurrentCountry(userId);

                                                // search nearby bookstores by city
```

```
                                        List<BookStore> list =
getBookStoresByCity(currentCity);
                                        if (list.isEmpty()) {
                                                // if there are no nearby bookstores,
search by country
                                                list =
getBookStoresByCity(currentCountry);
                                        }

                                        // create a string containing all the nearby
bookstores
                                        String str = "";
                                        for (BookStore bookStore : list) {
                                                str += String.format("- %d %s, %s,
%s\n",

        bookStore.getStreetnumber(), bookStore.getStreetname(),
                                                                bookStore.getCity(),
bookStore.getCountry());
                                        }
                                        if (str.isEmpty()) {
                                                str = String.format("Sorry, no book
stores found nearby %s, %s", currentCity, currentCountry);
                                        }
                                        child.setText(str);
                                }
                        }
                }
        } catch (Exception e) {
                e.printStackTrace();
        }
    }

    protected PaymentMethod getUserPaymentMethod(String userId) throws AxisFault {
            setEndpointURL(servicesUrl + "UserProfileService");
            System.out.println(pluginTag + " Retrieving payment method for user " +
userId);
            System.out.println(pluginTag + " Target Endpoint is: " + endpointURL);
            QName op = new QName(servicesNamespace, "getUserPaymentMethod");
            Object[] opArgs = new Object[] { userId };
            Class[] opReturnTypes = new Class[] { String.class };
            Object[] response = invokeBlocking(op, opArgs, opReturnTypes);
       String result = (String) response[0];
       System.out.println(pluginTag + " payment method: " + result);
            return PaymentMethod.valueOf(result);
    }

    protected float getBookPrice(String bookId) throws AxisFault {
            setEndpointURL(servicesUrl + "BooksListService");
            System.out.println(pluginTag + " Retrieving price for book " + bookId);
            System.out.println(pluginTag + " Target Endpoint is: " + endpointURL);
```

```
            QName op = new QName(servicesNamespace, "getBookPrice");
            Object[] opArgs = new Object[] { bookId };
            Class[] opReturnTypes = new Class[] { float.class };
            Object[] response = invokeBlocking(op, opArgs, opReturnTypes);
        float result = (float) response[0];
        System.out.println(pluginTag + " price: " + result);
            return result;
    }

    protected String getUserCardType(String userId) throws AxisFault {
            setEndpointURL(servicesUrl + "UserProfileService");
            System.out.println(pluginTag + " Retrieving card type for user " + userId);
            System.out.println(pluginTag + " Target Endpoint is: " + endpointURL);
            QName op = new QName(servicesNamespace, "getUserCardType");
            Object[] opArgs = new Object[] { userId };
            Class[] opReturnTypes = new Class[] { String.class };
            Object[] response = invokeBlocking(op, opArgs, opReturnTypes);
        String result = (String) response[0];
        System.out.println(pluginTag + " card type: " + result);
            return result;
    }

    protected String getUserCardNumber(String userId) throws AxisFault {
            setEndpointURL(servicesUrl + "UserProfileService");
            System.out.println(pluginTag + " Retrieving card number for user " + userId);
            System.out.println(pluginTag + " Target Endpoint is: " + endpointURL);
            QName op = new QName(servicesNamespace, "getUserCardNumber");
            Object[] opArgs = new Object[] { userId };
            Class[] opReturnTypes = new Class[] { String.class };
            Object[] response = invokeBlocking(op, opArgs, opReturnTypes);
        String result = (String) response[0];
        System.out.println(pluginTag + " card number: " + result);
            return result;
    }

    protected int getUserCardExpMonth(String userId) throws AxisFault {
            setEndpointURL(servicesUrl + "UserProfileService");
            System.out.println(pluginTag + " Retrieving card expiry month for user " +
userId);
            System.out.println(pluginTag + " Target Endpoint is: " + endpointURL);
            QName op = new QName(servicesNamespace, "getUserCardExpMonth");
            Object[] opArgs = new Object[] { userId };
            Class[] opReturnTypes = new Class[] { int.class };
            Object[] response = invokeBlocking(op, opArgs, opReturnTypes);
        int result = (int) response[0];
        System.out.println(pluginTag + " card expiry month: " + result);
            return result;
    }

    protected int getUserCardExpYear(String userId) throws AxisFault {
            setEndpointURL(servicesUrl + "UserProfileService");
```

```
            System.out.println(pluginTag + " Retrieving card expiry year for user " +
userId);
            System.out.println(pluginTag + " Target Endpoint is: " + endpointURL);
            QName op = new QName(servicesNamespace, "getUserCardExpYear");
            Object[] opArgs = new Object[] { userId };
            Class[] opReturnTypes = new Class[] { int.class };
            Object[] response = invokeBlocking(op, opArgs, opReturnTypes);
        int result = (int) response[0];
        System.out.println(pluginTag + " card expiry year: " + result);
            return result;
    }

    protected String getUserCurrentCity(String userId) throws AxisFault {
            setEndpointURL(servicesUrl + "CurrentLocationService");
            System.out.println(pluginTag + " Retrieving current city for user " + userId);
            System.out.println(pluginTag + " Target Endpoint is: " + endpointURL);
            QName op = new QName(servicesNamespace, "getCurrentCity");
            Object[] opArgs = new Object[] { userId };
            Class[] opReturnTypes = new Class[] { String.class };
            Object[] response = invokeBlocking(op, opArgs, opReturnTypes);
            String result = (String) response[0];
            System.out.println(pluginTag + " city: " + result);
            return result;
    }

    protected String getUserCurrentCountry(String userId) throws AxisFault {
            setEndpointURL(servicesUrl + "CurrentLocationService");
            System.out.println(pluginTag + " Retrieving current country for user " +
userId);
            System.out.println(pluginTag + " Target Endpoint is: " + endpointURL);
            QName op = new QName(servicesNamespace, "getCurrentCountry");
            Object[] opArgs = new Object[] { userId };
            Class[] opReturnTypes = new Class[] { String.class };
            Object[] response = invokeBlocking(op, opArgs, opReturnTypes);
            String result = (String) response[0];
            System.out.println(pluginTag + " country: " + result);
            return result;
    }

    protected List<BookStore> getBookStoresByCity(String city) throws AxisFault {
            setEndpointURL(servicesUrl + "BooksStoreService");
            System.out.println(pluginTag + " Retrieving bookstores by city " + city);
            System.out.println(pluginTag + " Target Endpoint is: " + endpointURL);
            QName op = new QName(servicesNamespace, "getBookStoresByCity");
            Object[] opArgs = new Object[] { city };
            OMElement response = invokeBlocking(op, opArgs);

            List<BookStore> result = new ArrayList<BookStore>();
        Iterator<?> iter = response.getChildElements();
            while (iter.hasNext()) {
                    OMElement elem = (OMElement) iter.next();
```

```
                        BookStore bookStore = (BookStore)
BeanUtil.deserialize(BookStore.class, elem, new MultirefHelper(elem), new
DefaultObjectSupplier());
                        result.add(bookStore);
                        System.out.println(pluginTag + String.format(" book [#%s] %d %s,
%s, %s",
                                        bookStore.getId(), bookStore.getStreetnumber(),
bookStore.getStreetname(), bookStore.getCity(), bookStore.getCountry()));
                }

                return result;
        }

        protected List<BookStore> getBookStoresByCountry(String country) throws
AxisFault {
                setEndpointURL(servicesUrl + "BooksStoreService");
                System.out.println(pluginTag + " Retrieving bookstores by country " +
country);
                System.out.println(pluginTag + " Target Endpoint is: " + endpointURL);
                QName op = new QName(servicesNamespace, "getBookStoresByCountry");
                Object[] opArgs = new Object[] { country };
                OMElement response = invokeBlocking(op, opArgs);

                List<BookStore> result = new ArrayList<BookStore>();
            Iterator<?> iter = response.getChildElements();
                while (iter.hasNext()) {
                        OMElement elem = (OMElement) iter.next();
                        BookStore bookStore = (BookStore)
BeanUtil.deserialize(BookStore.class, elem, new MultirefHelper(elem), new
DefaultObjectSupplier());
                        result.add(bookStore);
                        System.out.println(pluginTag + String.format(" book [#%s] %d %s,
%s, %s",
                                        bookStore.getId(), bookStore.getStreetnumber(),
bookStore.getStreetname(), bookStore.getCity(), bookStore.getCountry()));
                }

                return result;
        }

}
```

WelcomePlugin.java

```
package plugin.welcoming;

import gr.ntua.icbnet.contextModule.plugins.AbstractRPCQueryPlugin;

import java.util.Calendar;
import java.util.Iterator;

import javax.xml.namespace.QName;

import org.apache.axiom.om.OMAbstractFactory;
import org.apache.axiom.om.OMElement;
import org.apache.axiom.om.OMNode;
import org.apache.axiom.soap.SOAPEnvelope;
import org.apache.axiom.soap.SOAPFactory;
import org.apache.axis2.AxisFault;
import org.apache.axis2.context.MessageContext;

import parser.PrivacyPreferences;

public class WelcomePlugin extends AbstractRPCQueryPlugin {

	private String userId;
	private SOAPEnvelope soapEnv;
	private String servicesNamespace = "http://ws.books_service.cy";
	private String servicesUrl = "http://localhost:8080/axis2/services/";
	private String pluginTag = "[WelcomePlugin]";
	private PrivacyPreferences pp;

	public WelcomePlugin() {
		super();
	}

	public MessageContext modifySOAPEnvelope(MessageContext in) {
		soapEnv = in.getEnvelope();

		// get the userId header element attribute
		OMElement uidOE = soapEnv.getHeader().getFirstChildWithName(new
QName(servicesNamespace, "welcome"));
		if (uidOE != null && uidOE.getAttributeValue(new
QName(servicesNamespace, "userId")) != null) {
			userId = uidOE.getAttributeValue(new QName(servicesNamespace,
"userId"));
		}
		System.out.println(pluginTag + " ATTRIBUTE " + userId);

		pp = new PrivacyPreferences(userId);

		try {
			OMElement documentElement = (OMElement)
parseSOAPEnvelope(in.getEnvelope());
			super.parseChildren(in.getEnvelope().getBody());
			SOAPFactory fac = OMAbstractFactory.getSOAP11Factory();
			SOAPEnvelope envelope = fac.getDefaultEnvelope();
			Iterator<?> iter = documentElement.getChildren();
			while (iter.hasNext()) {
				OMNode child = (OMNode) iter.next();
				if (envelope != null && envelope.getBody() != null &&
child != null) {
```

```
                                envelope.getBody().addChild(child);
                        }
                }
                in.setEnvelope(envelope);
        } catch (Exception e) {
                e.printStackTrace();
        }

        return in;
    }

    protected void parseChild(OMElement child) {
        try {
                String localPart = child.getQName().getLocalPart();
                // arg0: user name argument
                if (localPart.compareToIgnoreCase("arg0") == 0) {
                        if (pp.isEnabled() &&
(pp.isPersonalInfoFieldAllowed("name") || !pp.isPersonalInfoFieldBlocked("name")))
{
                                child.setText(getUserName(userId));
                        }
                }
                // arg1: user surname argument
                else if (localPart.compareToIgnoreCase("arg1") == 0) {
                        if (pp.isEnabled() &&
(pp.isPersonalInfoFieldAllowed("surname") ||
!pp.isPersonalInfoFieldBlocked("surname"))) {
                                child.setText(getUserSurname(userId));
                        }
                }
                // arg2: user language argument
                else if (localPart.compareToIgnoreCase("arg2") == 0) {
                        if (pp.isEnabled()
                                        &&
(pp.isPersonalInfoFieldAllowed("country") ||
!pp.isPersonalInfoFieldBlocked("country"))
                                        && (pp.isPersonalInfoFieldAllowed("city")
|| !pp.isPersonalInfoFieldBlocked("city"))) {
                                String currentCountry =
getCurrentCountry(userId);
                                String currentCity = getCurrentCity(userId);
                                if ((pp.isLocationAllowed() ||
!pp.isLocationBlocked())
                                                &&
pp.validateAllowedLocationCondition(Calendar.getInstance(), currentCountry,
currentCity)) {
                                        child.setText(getCurrentLang(userId));
                                }
                        }
                }
        } catch (Exception e) {
                e.printStackTrace();
        }
    }

    protected String getUserName(String userId) throws AxisFault {
        setEndpointURL(servicesUrl + "UserProfileService");
        System.out.println(pluginTag + " Retrieving name for user " +
userId);
        System.out.println(pluginTag + " Target Endpoint is: " +
endpointURL);
```

```
            QName op = new QName(servicesNamespace, "getUserName");
            Object[] opArgs = new Object[] { userId };
            Class[] opReturnTypes = new Class[] { String.class };
            Object[] response = invokeBlocking(op, opArgs, opReturnTypes);
            String result = (String) response[0];
            System.out.println(pluginTag + " name: " + result);
            return result;
      }

      protected String getUserSurname(String userId) throws AxisFault {
            setEndpointURL(servicesUrl + "UserProfileService");
            System.out.println(pluginTag + " Retrieving surname for user " +
userId);
            System.out.println(pluginTag + " Target Endpoint is: " +
endpointURL);
            QName op = new QName(servicesNamespace, "getUserSurname");
            Object[] opArgs = new Object[] { userId };
            Class[] opReturnTypes = new Class[] { String.class };
            Object[] response = invokeBlocking(op, opArgs, opReturnTypes);
            String result = (String) response[0];
            System.out.println(pluginTag + " surname: " + result);
            return result;
      }

      protected String getCurrentLang(String userId) throws AxisFault {
            setEndpointURL(servicesUrl + "CurrentLocationService");
            System.out.println(pluginTag + " Retrieving language for user " +
userId);
            System.out.println(pluginTag + " Target Endpoint is: " +
endpointURL);
            QName op = new QName(servicesNamespace, "getNativeLanguage");
            Object[] opArgs = new Object[] { userId };
            Class[] opReturnTypes = new Class[] { String.class };
            Object[] response = invokeBlocking(op, opArgs, opReturnTypes);
            String result = (String) response[0];
            System.out.println(pluginTag + " language: " + result);
            return result;
      }

      protected String getCurrentCountry(String userId) throws AxisFault {
            setEndpointURL(servicesUrl + "CurrentLocationService");
            System.out.println(pluginTag + " Retrieving current country for user
" + userId);
            System.out.println(pluginTag + " Target Endpoint is: " +
endpointURL);
            QName op = new QName(servicesNamespace, "getCurrentCountry");
            Object[] opArgs = new Object[] { userId };
            Class[] opReturnTypes = new Class[] { String.class };
            Object[] response = invokeBlocking(op, opArgs, opReturnTypes);
            String result = (String) response[0];
            System.out.println(pluginTag + " country: " + result);
            return result;
      }
      protected String getCurrentCity(String userId) throws AxisFault {
            setEndpointURL(servicesUrl + "CurrentLocationService");
            System.out.println(pluginTag + " Retrieving current city for user "
+ userId);
            System.out.println(pluginTag + " Target Endpoint is: " +
endpointURL);
            QName op = new QName(servicesNamespace, "getCurrentCity");
            Object[] opArgs = new Object[] { userId };
```

```
            Class[] opReturnTypes = new Class[] { String.class };
            Object[] response = invokeBlocking(op, opArgs, opReturnTypes);
            String result = (String) response[0];
            System.out.println(pluginTag + " city: " + result);
            return result;
        }

}
```

Upp.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:upp="http://www.privacy.org/UPP"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
targetNamespace="http://www.privacy.org/UPP">
	<xs:element name="privacyPreferences" type="upp:UserPrivacyPreferences"/>
	<xs:complexType name="UserPrivacyPreferences">
		<xs:sequence>
			<xs:element name="allow" maxOccurs="unbounded">
				<xs:complexType>
					<xs:sequence>
						<xs:element minOccurs="0" name="dataEnd"
type="upp:dataType"/>
						<xs:element minOccurs="0"
name="serviceEnd" type="upp:serviceType"/>
						<xs:element minOccurs="0" name="use"
type="upp:useType"/>
						<xs:element minOccurs="0" name="condition"
type="upp:conditionType"/>
					</xs:sequence>
				</xs:complexType>
			</xs:element>
			<xs:element name="block" maxOccurs="unbounded">
				<xs:complexType>
					<xs:sequence>
						<xs:element minOccurs="0" name="dataEnd"
type="upp:dataType"/>
						<xs:element minOccurs="0"
name="serviceEnd" type="upp:serviceType"/>
					</xs:sequence>
				</xs:complexType>
			</xs:element>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="dataType">
		<xs:attribute name="dataCategory" type="upp:dataCategoryT"
use="optional"/>
		<xs:attribute name="dataName" type="xs:string" use="optional"/>

	</xs:complexType>
	<xs:complexType name="serviceType">
		<xs:attribute name="serviceCategory" type="xs:string"
use="optional"/>
		<xs:attribute name="serviceName" type="xs:string" use="optional"/>

		<xs:attribute name="serviceProvider" type="xs:string"
use="optional"/>
	</xs:complexType>
		<xs:complexType name="useType">
		<xs:sequence>
			<xs:element minOccurs="0" name="retention">
				<xs:complexType>
				<xs:simpleContent>
					<xs:extension base="upp:retentionT">
							<xs:attribute name="userDefined"
type="xs:time" />
					</xs:extension>
				</xs:simpleContent>
				</xs:complexType>
```

```
                    </xs:element>
                    <xs:element minOccurs="0" name="modification"
type="xs:boolean"/>
                    <xs:element minOccurs="0" name="abstraction"
type="upp:abstractionT"/>
              </xs:sequence>
       </xs:complexType>
       <xs:complexType name="conditionType">
              <xs:sequence>
                    <xs:element minOccurs="0" name="timePeriodIn"
type="upp:timePeriodType"/>
                    <xs:element minOccurs="0" name="timePeriodOut"
type="upp:timePeriodType"/>
                    <xs:element minOccurs="0" name="spaceLimitIn"
type="upp:spaceLimitType"/>
                    <xs:element minOccurs="0" name="spaceLimitOut"
type="upp:spaceLimitType"/>
              </xs:sequence>
       </xs:complexType>

    <xs:complexType name="timePeriodType">
              <xs:sequence>
                    <xs:element name="startWeekDay" type="xs:string"/>
                    <xs:element name="endWeekDay" type="xs:string"/>
                    <xs:element name="time" type="upp:timeInternalType"/>
              </xs:sequence>
       </xs:complexType>
       <xs:complexType name="timeInternalType">
              <xs:sequence>
                    <xs:element name="startTime" type="xs:time"/>
                    <xs:element name="endTime" type="xs:time"/>
              </xs:sequence>
       </xs:complexType>
       <xs:complexType name="spaceLimitType">
              <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0" name="Country"
type="xs:string"/>
                    <xs:element maxOccurs="unbounded" minOccurs="0" name="City"
type="xs:string"/>
                    <xs:element maxOccurs="unbounded" minOccurs="0" name="Address"
type="xs:string"/>
              </xs:sequence>
       </xs:complexType>

       <xs:simpleType name="retentionT">
              <xs:restriction base="xs:string">
                    <xs:enumeration value="NoRetention"/>
                    <xs:enumeration value="Indefinitely"/>
                    <xs:enumeration value="LawDefined"/>
                    <xs:enumeration value="UserDefined"/>
              </xs:restriction>
       </xs:simpleType>
       <xs:simpleType name="abstractionT">
              <xs:restriction base="xs:string">
                    <xs:enumeration value="None"/>
                    <xs:enumeration value="Full"/>
                    <xs:enumeration value="OneLevel"/>
                    <xs:enumeration value="TwoLevels"/>
              </xs:restriction>
       </xs:simpleType>
       <xs:simpleType name="dataCategoryT">
```

```
            <xs:restriction base="xs:string">
                  <xs:enumeration value="PersonalInfo"/>  <!-- e.g. name,age,
etc. -->
                  <xs:enumeration value="EmploymentStatus"/>
                  <xs:enumeration value="EducationLevel"/>
                  <xs:enumeration value="PersonalPreferences"/>
                  <xs:enumeration value="Activities"/> <!-- Current/Past
activities -->
                  <xs:enumeration value="Location"/>
                  <xs:enumeration value="HealthRecord"/>
                  <xs:enumeration value="Contacts"/>
                  <xs:enumeration value="PaymentInfo"/>
            </xs:restriction>
      </xs:simpleType>

</xs:schema>
```