ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Η χρήση των ηλεκτρονικών υπηρεσιών (e-services) στην ολοκλήρωση συστημάτων Ηλεκτρονικού Εμπορίου

Του φοιτητή

Βράκα Ιωάννη

Αρ. Μητρώου: 093508

Επιβλέπων καθηγητής

Τεκτονίδης Δημήτριος

Θεσσαλονίκη 2011

ΠΡΟΛΟΓΟΣ

Σταδιακά η δημιουργία πληροφοριακών συστημάτων απομακρύνεται από την στερεοτυπική προσέγγιση της δημιουργίας μεγάλων και πολύπλοκων εφαρμογών που θα προσφέρουν την επιθυμητή λειτουργικότητα. Η ευελιξία, η επεκτασιμότητα, η επαναχρησιμοποίηση και η δυνατότητα διασύνδεσης νέων και υπαρχόντων εφαρμογών με κατανάλωση περιορισμένων πόρων, τόσο σε χρόνο όσο και σε κόστος ανάπτυξης, έχουν οδήγηση στην δημιουργία της υπηρεσιοστεφούς αρχιτεκτονικής. Παράλληλα στον επιχειρηματικό χώρο εντείνεται ο ανταγωνισμός με τις επιχειρήσεις να επιδιώκουν υψηλή εξωστρέφια επιδεικνύοντας και παρέχοντας υπηρεσίες προς τους χρήστες του παγκόσμιου ιστού.

Σε αυτή την πτυχιακή εργασία θα μελετηθεί η υπηρεσιοστεφής αρχιτεκτονικής και τεχνολογίες υλοποίησής της μέσα από την δημιουργία διεπαφών διασύνδεσης ενός ηλεκτρονικού καταστήματος και ενός εμπορικού πληροφοριακού συστήματος.

ΠΕΡΙΛΗΨΗ

Στο πρώτο μέρος αυτής της πτυχιακής εργασίας θα αναλυθούν τα κύρια χαρακτηριστικά μιας αρχιτεκτονικής λογισμικού και θα παρουσιαστεί η υπηρεσιοστεφής αρχιτεκτονική SOA (Service Oriented Architecture) ως η αρχιτεκτονική που μπορεί να αντιμετωπίσεις τις σύγχρονες προκλήσεις στον ευμετάβλητο επιχειρησιακό περιβάλλον του σήμερα. Έπειτα θα παρουσιαστούν τα χαρακτηριστικά μιας από τις κύριες τεχνολογίες υλοποίησής της, των Υπηρεσιών Ιστού (Web Services).

Στην συνέχεια χρησιμοποιώντας το παράδειγμα διασύνδεσης ενός ηλεκτρονικού καταστήματος με ένα εμπορικό πληροφοριακό σύστημα, του VirtueMart και του Soft1 ERP, θα παρουσιαστούν τα στάδια σχεδιασμού και υλοποίησης των υπηρεσιών ιστού με την χρήση SOAP, WSDL και PHP. Θα αναλυθεί η δομή των αρχείων WSDL, η ανταλλαγή μηνυμάτων και δεδομένων μεταξύ του καταναλωτή της υπηρεσίας και του SOAP Server, και η δημιουργία του SOAP Server σε PHP.

Τέλος θα παρουσιαστεί η εφαρμογή ενδιάμεσου λογισμικού (middleware) που δημιουργήθηκε σε C# για την διασύνδεση του ηλεκτρονικού καταστήματος με το ERP. Η εφαρμογή θα καταναλώνει τις υπηρεσίες που δημιουργήθηκαν για να επικοινωνεί με το ηλεκτρονικό κατάστημα, ενώ για την επικοινωνία με το ERP θα κάνει χρήση βιβλιοθηκών .NET που παρέχονται από το ERP.

ABSTRACT

In the first part of the thesis it will be analysed the main characteristics of a software architecture based on web services. The SOA architecture (Service Oriented Architecture) will be presented as the architecture that can meet the challenges that software development is called to face in the modern continuously evolving business environment. Also there will be an analysis of the characteristics of one of the main technologies for the implementation of SOA, the Web Services.

In the second part of the thesis the case of the integration of an e-shop with a business information system will be analysed in order to present the development steps that need to be taken in order to design and create web services with the use of SOAP, WSDL and PHP. An analysis of the structure of an WSDL file will be included as well as a presentation of the messages exchanged between the client and the SOAP Server, ending with an overview of the development of SOAP Servers in PHP. In this case the e-shop system that will be used is the VirtueMart, and the business information system is the Soft1 ERP.

Finally there is a presentation of the interface, the design and the methods of the middleware application that has been created as part of this project for the integration of the e-shop and the ERP. The application will consume the web services that have been created for the e-shop and will use the .NET libraries provided with the ERP in order to communicate with the ERP.

ΕΥΧΑΡΙΣΤΙΕΣ

Σε όλους που στάθηκαν δίπλα μου σε αυτή την προσπάθειά μου.

Κατάλογος περιεχομένων

Ευρετήριο πινάκων

Ευρετήριο σχημάτων

ΕΙΣΑΓΩΓΗ

Στην παρούσα πτυχιακή εργασία καλύπτεται το θέμα της χρήσης υπηρεσιών ιστού για την διασύνδεση εφαρμογών και επέκταση της λειτουργικότητά τους. Το ζήτημα προσεγγίζεται τόσο θεωρητικά όσο και πρακτικά με την παρουσίαση των βασικών αρχών που διέπουν την υπηρεσιοστραφή αρχιτεκτονική (SOA) και τις υπηρεσίες ιστού (Web Services), αλλά και με την διεξοδική παρουσίαση του τρόπου δημιουργίας υπηρεσιών ιστού και λογισμικού διασύνδεσης για την συνένωση λειτουργιών ενός ηλεκτρονικού καταστήματος και ενός εμπορικού πληροφοριακού συστήματος επιχείρησης.

Η οργάνωση των κεφαλαίων αποσκοπεί στην προοδευτική ανάλυση του θέματος “οικοδομώντας” σε έννοιες και μεθοδολογίες που παρουσιάζονται από κεφάλαιο σε κεφάλαιο. Στο πρώτο κεφάλαιο αναλύονται τα βασικά χαρακτηριστικά μιας αρχιτεκτονικής λογισμικού, οι προκλήσεις που παρουσιάζονται στα σύγχρονα πληροφοριακά συστήματα και γιατί η υπηρεσιοστεφής αρχιτεκτονική (SOA) μπορεί να τις καλύψει. Το δεύτερο κεφάλαιο προχωράει στην ανάλυση της υπηρεσιοστεφούς αρχιτεκτονικής και στην υλοποίησή της με χρήση των υπηρεσιών ιστού (Web Services). Το επόμενο κεφάλαιο αφιερώνεται στην ανάλυση της δομής του WSDL αρχείου που είναι βασικό τόσο για την δημιουργία, όσο και για την κατανόηση του τρόπου διασύνδεσης και ανταλλαγής μηνυμάτων μεταξύ του καταναλωτή μιας υπηρεσίας και του πάροχου. Αφού έχουν μπει οι βάσεις στα πρώτα τρία κεφάλαια, τα επόμενα κεφάλαια ασχολούνται με τον σχεδιασμό και την υλοποίηση των υπηρεσιών ιστού και του ενδιάμεσου λογισμικού για την επίτευξη της διασύνδεσης του ηλεκτρονικού καταστήματος και του εμπορικού πληροφοριακού συστήματος. Στο τέταρτο κεφάλαιο παρουσιάζεται η λειτουργικότητα, οι δομές δεδομένων και ο βασικός σχεδιασμός της λύσης. Στο πέμπτο κεφάλαιο ομαδοποιούνται οι λειτουργίες σε τέσσερις υπηρεσίες ιστού και

παρουσιάζονται οι δομές των WSDL αρχείων τους. Το επόμενο κεφάλαιο καλύπτει την υλοποίηση των υπηρεσιών ιστού για το VirtueMart σε PHP αναλύοντας συγχρόνων και τον τρόπο προσπέλασης των παραμέτρων εισαγωγής και τον τρόπο δόμησης των αποτελεσμάτων/απαντήσεων της υπηρεσίας. Εν τέλη στο τελευταίο κεφάλαιο γίνεται η επισκόπηση της δομής και λειτουργίας του ενδιάμεσου λογισμικού και του τρόπου κατανάλωσης υπηρεσιών ιστού με την χρήση της C# και του Visual Studio 2008.

Θεωρείται ότι ο αναγνώστης έχει γνώσεις XML, PHP, SQL και C#.

ΚΕΦΑΛΑΙΟ 1

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΛΟΓΙΣΜΙΚΟΥ

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο εξετάζουμε το υπόβαθρο πάνω στο οποίο θα αναλύσουμε την υπηρεσιοστρεφής αρχιτεκτονική και την υλοποίησή της με Web Services. Αρχικά δίνεται ο ορισμός της έννοιας της αρχιτεκτονικής λογισμικού και οι βασικές ιδιότητές της. Στην συνέχεια εξετάζουμε την πρόκληση που καλούνται να αντιμετωπίσουν τα σύγχρονα πληροφοριακά συστήματα στο συνεχώς μεταβαλλόμενο επιχειρηματικό περιβάλλον, προσδίδοντας στην επιχείρηση την δυνατότητα να προσαρμόζεται γρήγορα και με χαμηλό κόστος ώστε να κεφαλοποιεί το επιχειρηματικό πλεονέκτημα που πιθανώς παρουσιάζεται. Το κεφάλαιο κλείνει με μια εισαγωγή στην υπηρεσιοστρεφή αρχιτεκτονική.

1.1 Τι είναι μια αρχιτεκτονική λογισμικού

Η αρχιτεκτονική λογισμικού ενός πληροφοριακού συστήματος συμπεριλαμβάνει την περιγραφή των δομών που χρησιμοποιούνται για την δημιουργία συστημάτων (συστατικά λογισμικού), τις ιδιότητές αυτών των συστατικών που είναι διαθέσιμες εξωτερικά, οι αλληλεπιδράσεις μεταξύ τους καθώς και αρχές, πρότυπα και βέλτιστες πρακτικές σχετικά με την σχεδίαση και εξέλιξής τους στο μέλλον. Οι εξωτερικά διαθέσιμες ιδιότητες του συστήματος είναι παραδοχές που άλλα συστατικά λογισμικού μπορούν να χρησιμοποιήσουν όπως οι διαθέσιμες διεπαφές και υπηρεσίες, χαρακτηριστικά επίδοσης, διαχείριση σφαλμάτων, διαχείριση κοινόχρηστων πόρων κ.α. (Papazoglou 2008)

Πιο συγκεκριμένα η αρχιτεκτονική λογισμικού είναι η δομή του συστήματος σε υψηλό επίπεδο – συμπεριλαμβανομένων και κατανεμημένων και υπηρεσιοστρεφών συστημάτων – που συνήθως ορίζεται ως μια συλλογή λειτουργικών συστατικών και των μεταξύ τους διασυνδέσεων. Καθορίζονται τα συστατικά (components) και τους ανατίθενται λειτουργικότητα που στην συνέχεια γίνεται προσβάσιμη σε συστατικά-πελάτες μέσω αμοιβαίως προσδιορισμένων διεπαφών. Οι διασυνδέσεις των συστατικών καθορίζουν τους μηχανισμούς επικοινωνίας και ελέγχου, και υποστηρίζουν όλες τις αλληλεπιδράσεις που είναι απαραίτητες για να επιτευχθεί λειτουργικότητα συστήματος.

1.2 Βασικές ιδιότητες αρχιτεκτονικής λογισμικού

Μια αρχιτεκτονική λογισμικού αποτελεί την βάση για την δημιουργία επιτυχημένων πληροφοριακών συστημάτων, αντιπροσωπεύει ένα αφηρημένο επαναχρησιμοποιούμενο μοντέλο που μπορεί να εφαρμοστεί σε διάφορα συστήματα. Μερικές από τις βασικές ιδιότητες μια αρχιτεκτονικής λογισμικού είναι (Papazoglou 2008):

- Αποτελεί μια υψηλού επιπέδου θεώρηση ολόκληρου του συστήματος που όμως παρέχει αρκετές πληροφορίες για τον σχηματισμό μιας βασικής ανάλυσης, την λήψη αποφάσεων, και κατ' επέκταση την μείωση των λαθών σχεδίασης.
- Η δομή πρέπει να υποστηρίζει την απαιτούμενη λειτουργικότητα του συστήματος. Επομένως, κατά την σχεδίαση της αρχιτεκτονικής πρέπει να ληφθούν υπόψιν τα δυναμικά χαρακτηριστικά του συστήματος.
- Πρέπει να συμμορφώνεται με τις ποιοτικές ιδιότητες του συστήματος (καταγεγραμμένες σε συμφωνητικά παροχής υπηρεσιών – SLAs – ή αλλιώς μη λειτουργικές απαιτήσεις συστήματος). Αυτές μεταξύ άλλων περιλαμβάνουν την απόδοση του συστήματος, την ασφάλεια,

την διαλειτουργικότητα, απαιτήσεις αξιοπιστίας συσχετισμένες με την τρέχουσα λειτουργικότητα, την ευκαμψία του συστήματος ή απαιτήσεις επεκτασιμότητας συσχετισμένες με την κάλυψη μελλοντικής λειτουργικότητας με ένα συμβατικό κόστος αλλαγής. Καθώς μερικές από τις ποιοτικές ιδιότητες του συστήματος μπορεί να αντικρούονται πρέπει να καθοριστούν οι ανάλογοι συμβιβασμοί που πρέπει να γίνουν. Οι συμβιβασμοί αυτοί πρέπει να γίνουν σε συνδυασμό με εναλλακτικές λύσεις λαμβάνοντας υπόψιν τις σχετικές προτεραιότητες του εν λόγω συστήματος.

- Στο επίπεδο της αρχιτεκτονικής πρέπει να αποκρυφτούν όλες οι λεπτομέρειες υλοποίησης.

Δύο από τα βασικά συστατικά μιας αρχιτεκτονικής λογισμικού είναι οι λειτουργικές απαιτήσεις της και οι ποιοτικές ιδιότητες του συστήματος. Οι λειτουργικές απαιτήσεις καλύπτουν την επιδιωκόμενη συμπεριφορά του συστήματος και μπορεί να οριστεί ως υπηρεσίες, διεργασίες, ή συναρτήσεις που το σύστημα πρέπει να εκτελέσει.

1.3 Η πρόκληση των σύγχρονων πληροφοριακών συστημάτων

Σε μεγάλες επιχειρήσεις είναι συνηθισμένο φαινόμενο να υπάρχουν διάφορα ετερογενή πληροφοριακά συστήματα. Τα συστήματα αυτά μπορεί να είναι κατασκευασμένα σε διαφορετικές πλατφόρμες, με διαφορετική αρχιτεκτονική προσέγγιση και χρησιμοποιώντας διαφορετικές τεχνολογίες. Ακόμη μπορεί να έχουν σχεδιαστεί για την κάλυψη συγκεκριμένων αναγκών, να χρησιμοποιούνται μόνο από συγκεκριμένα τμήματα της επιχείρησης και το κάθε ένα τους να χρειάζεται συντήρηση και βελτίωση ανάλογα με τις μεταβολές των αναγκών της επιχείρησης. Καθώς τα συστήματα αυτά εξελίσσονται παράλληλα και ανεξάρτητα παρουσιάζεται το πρόβλημα ότι το κόστος συντήρησης και εξέλιξής τους μεγαλώνει συνεχώς ενώ παράλληλα το επιχειρηματικό όφελος που προσφέρει

κάθε εξέλιξή τους μειώνεται. Δηλαδή οι επιχειρήσεις ξοδεύουν περισσότερα αποκομίζοντας μικρότερο όφελος.

Μέρος αυτού του προβλήματος είναι το ότι οι επιχειρήσεις δεν αποφασίζουν και δεν σχεδιάζουν τα συστήματά τους συγκεντρωτικά, αντιθέτως λαμβάνουν αποφάσεις ανταποκρινόμενες σε εξωτερικές μεταβολές του γενικότερου επιχειρηματικού περιβάλλοντος. Οι μεταβολές αυτές μπορεί να προκαλέσουν αλλαγές στις εσωτερικές επιχειρησιακές διεργασίες προσθέτοντας σε αυτές νέα χαρακτηριστικά ή ακόμη μπορεί να προκαλέσουν και την ολική αναδιάρθρωσή τους. Ακόμη μπορεί να απαιτηθεί η προσωρινή μεταβολή κάποιας διεργασίας για να γίνει δυνατή η αντιμετώπιση κάποιας νέας κατάστασης. Το πληροφοριακό σύστημα θα πρέπει να είναι σε θέση να ανταποκριθεί στις νέες απαιτήσεις γρήγορα και με χαμηλό κόστος ώστε να επιτευχθεί το απαιτούμενο επιχειρηματικό πλεονέκτημα.

Επίσης οι τεχνολογικές εξελίξεις στον τομέα των πληροφοριακών συστημάτων και της πληροφορικής γενικότερα επιτάσσουν την χρήση τους στην κατασκευή νέων συστημάτων. Το κόστος όμως ανάπτυξής νέων συστημάτων πολλές φορές είναι απαγορευτικά υψηλό αναγκάζοντας τις επιχειρήσεις να επιλέξουν να μεταβάλουν τα ήδη υπάρχοντα συστήματα ώστε να χρησιμοποιήσουν τις νέες τεχνολογίες που είναι διαθέσιμες, με αποτέλεσμα το πρόβλημα να διογκώνεται και η πληροφοριακή υποδομή των επιχειρήσεων να γίνεται όλο και πιο δύσκολο να ακολουθήσει τις επιχειρησιακές ανάγκες που παρουσιάζονται.

1.4 Η αντιμετώπιση της πρόκλησης

Για να αντιμετωπιστούν τα παραπάνω προβλήματα στην πραγματικότητα χρειαζόμαστε να βρούμε απαντήσεις σε δύο επίπεδα, στο τεχνολογικό και στο αρχιτεκτονικό επίπεδο.

Η τεχνολογική προσέγγιση πρέπει να απλοποιεί το τρόπο με τον οποίο ενοποιούμε πληροφοριακά συστήματα επιτρέποντας την συνέχιση χρήσης συστημάτων που υπάρχουν ήδη στις επιχειρήσεις. Ενώ η αρχιτεκτονική προσέγγιση πρέπει να αναδιοργανώσει τον τρόπο που οι επιχειρήσεις δημιουργούν, χρησιμοποιούν, διαχειρίζονται και εξελίσσουν τα πληροφοριακά συστήματά τους. Η Υπηρεσιοστρεφής Αρχιτεκτονική (Service Oriented Architecture – SOA), παρέχει λύσεις και στα δύο επίπεδα

1.5 Υπηρεσιοστρεφής Αρχιτεκτονική (Service Oriented Architecture – SOA)

Η Υπηρεσιοστρεφής Αρχιτεκτονική εστιάζεται στην δημιουργία πληροφοριακών συστημάτων που παρέχουν υπηρεσίες στους τελικούς χρήστες του συστήματος ή σε άλλες υπηρεσίες μέσω διεπαφών. Η SOA αναδιοργανώνει τα συστήματα και την υποδομή που ήδη υπάρχουν στην επιχείρηση δημιουργώντας ένα δίκτυο υπηρεσιών οι οποίες είναι διαθέσιμες, αλλά και ανιχνεύσιμες, μέσω τυποποιημένων διεπαφών και πρωτοκόλλων ανταλλαγής μηνυμάτων. Στην συνέχεια οι υπηρεσίες μπορούν να χρησιμοποιηθούν τόσο από νέα συστήματα, όσο και από τα υπάρχουσα, παρέχοντας διασύνδεση μεταξύ συστημάτων διαφορετικών τεχνολογιών (Papazoglou 2008).

1.6 Ο στόχος της SOA

Ο βασικός στόχος της SOA είναι να προσφέρει διαλειτουργικότητα μεταξύ υπαρχόντων τεχνολογιών και επεκτασιμότητα σε μελλοντικές χρήσεις και αρχιτεκτονικές. Η SOA αντιμετωπίζει το πρόβλημα της διαλειτουργικότητας μετατρέποντας στατικά και μονολιθικά συστήματα σε ευέλικτα και αυτοτελή συστατικά υπηρεσιών που είναι διαθέσιμα μέσω τυποποιημένων πρωτοκόλλων επικοινωνίας. Οι υπηρεσίες μπορούν να κληθούν αυτοδύναμα ή να συγκεραστούν δημιουργώντας λύσεις υπηρεσιών ή επιχειρησιακές διαδικασίες πολλαπλών επιπέδων. Κάθε συστατικό υπηρεσίας μπορεί να εξελιχθεί αυτοτελώς χωρίς να απαιτηθεί κάποια άλλη αλλαγή στο σύστημα.

1.7 Βασικές αρχές της SOA

Προτυποποίηση λειτουργικής περιγραφής

Για να γίνουν διαθέσιμες οι υπηρεσίες στους χρήστες πρέπει να δημοσιεύσουν τον σκοπό και την λειτουργικότητά τους. Η προτυποποίηση του τρόπου περιγραφής των λειτουργιών, των τύπων δεδομένων, των μοντέλων δεδομένων και τον τρόπο που διάφορες πολιτικές υπηρεσιών συνθέτονται και αλληλεπιδρούν είναι βασικής σημασίας.

Χαλαρή διασύνδεση

Η διασύνδεση αναφέρεται στην σχέση που υπάρχει μεταξύ δύο πραγμάτων και είναι ανάλογη με την εξάρτηση. Η αρχή αυτή τονίζει την ανάγκη χαλαρής διασύνδεσης (εξάρτησης) μεταξύ της περιγραφής της υπηρεσίας, της υλοποίησής της, και των καταναλωτών της. Προωθεί την ανεξάρτητη σχεδίαση και ανάπτυξη της λογικής της υπηρεσίας και της υλοποίησής της ενώ συγχρόνως εγγυάται την διαλειτουργικότητα στους πελάτες της.

Αφαιρετικότητα

Οι υπηρεσίες πρέπει να κρύβουν όσο το δυνατόν περισσότερο την εσωτερική τους λειτουργία. Πετυχαίνοντας το καταφέρνουμε να δημιουργήσουμε χαμηλή διασυνδεσιμότητα όπως περιγράφτηκε παραπάνω. Η αφαιρετικότητα των υπηρεσιών παίζει σημαντικό ρόλο και στην δυνατότητα σύνθεσης τους σε λειτουργικές μονάδες.

Επαναχρησιμοποίηση

Η αρχή της επαναχρησιμοποίησης είναι βασική για τη SOA. Η υπηρεσία αναλύεται και σχεδιάζεται ως μια αυτοδύναμη λειτουργική μονάδα που είναι ανεξάρτητη από οποιαδήποτε λειτουργικότητα του συστήματος έξω από αυτή. Έτσι θεωρείτε ως ένας πόρος του συστήματος που είναι διαθέσιμος για επαναχρησιμοποίηση από διαφορετικά μέρη του συστήματος.

Αυτονομία

Για να μπορεί μια υπηρεσία να λειτουργεί με συνέπεια και αξιοπιστία, πρέπει η εσωτερική της λειτουργικότητα να ελέγχει το εσωτερικό της περιβάλλον και πόρους. Με την επίτευξη αυτονομίας των υπηρεσιών διασφαλίζεται η αξιοπιστία και τη προβλεψιμότητα της λειτουργία τους.

Άνευ κατάστασης

Η διαχείριση δεδομένων κατάστασης μεταξύ του καταναλωτή και της υπηρεσίας μπορεί να επηρεάσει αρνητικά την διαθεσιμότητα και την επεκτασιμότητα της υπηρεσίας. Για αυτόν τον λόγο οι υπηρεσίες πρέπει να σχεδιάζονται με τέτοιο τρόπο ώστε να διατηρούν την κατάστασή τους μόνο όταν είναι απολύτως απαραίτητο.

Ανακτησιμότητα

Η υπηρεσίες που αναπτύσσονται από ένα πάροχο θα πρέπει να μπορούν να ανακτηθούν εύκολα. Η σχεδίαση τους πρέπει διασφαλίζει την επακριβής περιγραφή της λειτουργικότητας και του τρόπου διασύνδεσής με αυτές, ανεξαρτήτως αν υπάρχει κάποιος μηχανισμός αυτοματοποιημένης ανάκτησης (όπως είναι το μητρώο υπηρεσιών) στο σύστημα.

Συνθεσιμότητα

Καθώς η πολυπλοκότητα των υπηρεσιοστρεφών συστημάτων μεγαλώνει, αναπτύσσεται και η σύνθεση υπηρεσιών. Η δυνατότητα της σύνθεσης υπηρεσιών για την δημιουργία νέων μονάδων λειτουργικότητας είναι από τις βασικότερες ανάγκες που καλείται να καλύψει η SOA. Η σχεδίαση υπηρεσιών πρέπει να προβλέπει την πιθανή χρήση τους σε πολύπλοκες συνθέσεις υπηρεσιών για να αποφευχθεί μελλοντική επανασχεδίασή τους. Οι υπηρεσίες πρέπει να σχεδιαστούν και να υλοποιηθούν ως μέρος συνθέσεων ακόμη και αν αρχικά δεν θα χρησιμοποιηθούν σε κάποια σύνθεση.

Διαλειτουργικότητα

Στην πραγματικότητα η αρχή της διαλειτουργικότητας εμπεριέχεται σε όλες τις παραπάνω αρχές και είναι μια βασική έννοια της SOA. Χωρίς αυτή απλά δεν μπορεί να υπάρξει η SOA. Παρακάτω βρίσκονται μερικά παραδείγματα για το πως η αρχή της διαλειτουργικότητας υφίσταται σε όλες τις παραπάνω αρχές.

- Η προτυποποίηση της λειτουργικής περιγραφής εγγυάται έως ένα βαθμό την διαλειτουργικότητα που αφορά την εναρμόνιση των μοντέλων δεδομένων

- Με την χαλαρή διασύνδεση των υπηρεσιών προστατεύουμε την διαλειτουργικότητά τους κάνοντάς τες λιγότερο εξαρτημένες σε άλλες υπηρεσίες και ευκολότερα χρησιμοποιούμενες από άλλους πελάτες υπηρεσιών.
- Η αφαιρετικότητα των υπηρεσιών θέτει την λειτουργική περιγραφή της υπηρεσίας ως την μοναδική δίοδο επικοινωνίας και μεταφοράς με το εξωτερικό περιβάλλον διασφαλίζοντας την μακροχρόνια συνέπεια και διαλειτουργικότητά τους, επιτρέποντας στην εσωτερική δομή και λειτουργία τους να εξελιχθεί ανεξάρτητα.
- Η σχεδίαση υπηρεσιών για επαναχρησιμοποίηση συνεπάγεται υψηλού επιπέδου διαλειτουργικότητα μεταξύ τους και με πιθανές υπηρεσίες καταναλωτές.
- Με την αυξημένη αυτονομία μιας υπηρεσίας επιτυγχάνουμε την συνέπεια της λειτουργικότητάς της, αυξάνοντας συγχρόνως και την διαλειτουργικότητά της
- Δίνοντας έμφαση στη σχεδίαση άνευ κατάστασης, η διαθεσιμότητα και επεκτασιμότητα της υπηρεσίας αυξάνεται, επιτρέποντας κατ' επέκταση την συχνή αξιόπιστη χρήση της.
- Η ανακτησιμότητα των υπηρεσιών επιτρέπει τον εύκολο εντοπισμό τους από αυτούς που επιθυμούν να την χρησιμοποιήσουν.
- Για να μπορέσουμε να επιτύχουμε σύνθεση υπηρεσιών είναι απαραίτητο οι υπηρεσίες να είναι διαλειτουργικές. Η επιτυχής εκπλήρωση των απαιτήσεων για συνθεσιμότητα συχνά συνδέεται άμεσα με τον βαθμό στον οποίο οι υπηρεσίες είναι τυποποιημένες και ο τρόπος ανταλλαγής δεδομένων μεταξύ υπηρεσιών είναι βελτιστοποιημένος.

Ένας βασικός στόχος της υπηρεσιοστρεφής αρχιτεκτονικής είναι η έννοια της διαλειτουργικότητας να διέπει όλο το φάσμα μελέτης, σχεδίασης και υλοποίησης του συστήματος, θέτοντάς την ως το βασικό κοινό χαρακτηριστικό σχεδίασης των υπηρεσιών.

Φυσικά υπάρχουν διάφορα επίπεδα διαλειτουργικότητας που μπορούν να επιτευχθούν, το σε πιο επίπεδο μπορεί να φτάσει ένα σύστημα εξαρτάται από το πόσο υλοποιήθηκαν οι αρχές της υπηρεσιοστεφούς σχεδίασης και επίσης κατά πόσο ώριμη είναι η επιλεγμένη τεχνολογία υλοποίησης του συστήματος (SOA Principles, 2010).

ΕΠΙΛΟΓΟΣ

Σε αυτό το κεφάλαιο παρουσιάστηκαν τα χαρακτηριστικά μιας αρχιτεκτονικής λογισμικού και ποιες προκλήσεις υπάρχουν στο επιχειρηματικό γίγνεσθαι του σήμερα. Στην συνέχεια αναλύθηκαν οι στόχοι και τα βασικά στοιχεία της υπηρεσιοστραφούς αρχιτεκτονικής που καλείται να προσφέρει λύσεις στον δαιδαλώδη κόσμο της ανάπτυξης λογισμικού και της ολοκλήρωσης και διασύνδεσης πληροφοριακών συστημάτων. Στο επόμενο κεφάλαιο θα αναλυθεί για ποιον λόγο η χαμηλή διασύνδεση είναι απαραίτητη στην κατασκευή συστατικών λογισμικού, θα εξετασθεί η έννοια της υπηρεσίας και θα παρουσιαστούν τα βασικά στοιχεία της υπηρεσιοστραφούς αρχιτεκτονικής και των υπηρεσιών ιστού.

ΚΕΦΑΛΑΙΟ 2

SOA ΚΑΙ WEB SERVICES

ΕΙΣΑΓΩΓΗ

Στο παρόν κεφάλαιο εξετάζεται γιατί η χαμηλή διασύνδεση (εξάρτηση) μεταξύ των συστατικών ενός συστήματος είναι απαραίτητη, τις διάφορες τεχνολογίες δόμησης λογισμικού και κατανεμημένων πληροφοριακών συστημάτων, και το πως η υπηρεσιοστρεφής προσέγγιση αποτελεί μια επέκτασή. Στην συνέχεια αναλύεται η έννοια της υπηρεσίας τόσο στην καθημερινή ζωή όσο και ως λογισμικό και τέλος θα παρουσιάζονται τα βασικά στοιχεία της υπηρεσιοστραφούς αρχιτεκτονικής και των υπηρεσιών ιστού.

2.1 Χαμηλή διασύνδεση, διαλειτουργικότητα και τεχνολογίες κατανεμημένων πληροφοριακών συστημάτων

Όπως εξετάσαμε στο προηγούμενο κεφάλαιο τα σύγχρονα πληροφοριακά συστήματα καλούνται να κάνουν χρήση προϋπαρχόντων συστημάτων εντός μιας επιχείρησης, προσφέροντας συγχρόνως διαλειτουργικότητα, προσαρμοστικότητα, επεκτασιμότητα και ανεξαρτησία ανάπτυξης ως βασικά στοιχεία της συνολικής αρχιτεκτονικής θεώρησης. Για να επιτευχθεί το επιθυμητό αποτέλεσμα πρέπει να υπάρχει χαμηλή διασύνδεση μεταξύ των υπηρεσιών/λειτουργιών που προσφέρονται από το σύστημα και του καταναλωτή που τις χρησιμοποιεί. Η έννοια της χαμηλής διασύνδεσης απαγορεύει οποιαδήποτε γνώση ή υπόθεση σχετικά με την πλατφόρμα πάνω στην οποία ο καταναλωτής ή η υπηρεσία τρέχουν, ιδιαιτερότητες για την υλοποίησή τους ή των πρωτοκόλλων που χρησιμοποιούνται για την επικοινωνία μεταξύ τους (Weerawarana 2005).

2.1.1 Η ευθραυστότητα των Αντικειμενοσταφή Συστημάτων

Συνήθως τα υπάρχοντα κατανεμημένα υπολογιστικά συστήματα είναι υλοποιημένα με αντικειμενοσταφής τεχνολογίες. Σε αυτές τις περιπτώσεις μια υπηρεσία γίνεται διαθέσιμη ως μια μέθοδος μιας κλάσης που υλοποιείται από ένα αντικείμενο. Ο καταναλωτής που επιθυμεί να χρησιμοποιήσει την υπηρεσία πρέπει να χρησιμοποιήσει ολόκληρο το αντικείμενο στην εφαρμογή του. Δηλαδή κάποιος που χρειάζεται μόνο μια λειτουργία πρέπει να χρησιμοποιήσει ολόκληρη την κλάση ή μια ολόκληρη ιεραρχία κλάσεων. Όταν η κλάση που χρησιμοποιεί ή κάποια κλάση ανωτέρου βαθμού αλλάξει πρέπει να γίνει και μια αντίστοιχη αλλαγή στην εφαρμογή που την χρησιμοποιεί. Ο καταναλωτής και η υπηρεσία είναι στενά συνδεδεμένα, κάτι που σημαίνει ότι η εφαρμογή του καταναλωτή είναι εύθραυστη εξαιτίας της εξάρτησής αυτής.

2.1.2 Έλλειψη διαλειτουργικότητας

Οι διάφορες τεχνολογίες κατανεμημένων υπολογιστικών συστημάτων όπως η CORBA (Common Object Request Broker Architecture), η J2EE (Java 2 Platform Enterprise Edition), και η COM (Component Object Model) βασίζονται σε διαφορετικά και ασύμβατα αντικειμενοστραφή μοντέλα, με αποτέλεσμα να υπάρχει δυσκολία στην υλοποίηση διαλειτουργικών μονάδων λογισμικού. Η διαφορές μεταξύ τους επεκτείνονται και σε υψηλότερα επίπεδα όπως η διαχείριση συναλλαγών, η ασφάλεια και ο τρόπος ανταλλαγής μηνυμάτων.

2.1.3 Ενδιάμεσο λογισμικό βασισμένο σε μηνύματα

Ως αποτέλεσμα των παραπάνω προβλημάτων διαλειτουργικότητας δημιουργούνται εφαρμογές ενδιάμεσου λογισμικού. Ενώ η ανάγκη εύκολης ενσωμάτωσης εφαρμογών από διαφορετικά ενδιάμεσα λογισμικά έχει γίνει ένα από τα βασικά ζητούμενα στην κατασκευή πληροφοριακών συστημάτων (Enterprise Application Integration – EAI). Μια βασική προσέγγιση επίτευξης μιας

τέτοιας ενσωμάτωσης ενδιάμεσου λογισμικού είναι η μηνυματοστρεφή προσέγγιση.

2.1.4 Προσαρμογείς και κανάλια

Κατά την μηνυματοστρεφή προσέγγιση υλοποιείται ανταλλαγή μηνυμάτων μεταξύ των εφαρμογών που ενοποιούνται. Για αυτό τον σκοπό δημιουργούνται προσαρμογείς που καλύπτουν τις εφαρμογές και αναλαμβάνουν την μετατροπή των μηνυμάτων που θα μεταφερθούν μέσω του καναλιού. Το κανάλι εγγυάται την μεταφορά των μηνυμάτων προσφέροντας χαρακτηριστικά ποιότητας όπως “μοναδική μεταφορά”. Επίσης το κανάλι προσφέρει υπηρεσίες μεταγλώττισης των μηνυμάτων μετατρέποντάς τα σε μια μορφή που είναι συμβατή με τον προσαρμογέα προορισμού (Weerawarana 2005).

Στο παρακάτω σχήμα απεικονίζεται η παραπάνω προσέγγιση. Υπάρχουν δύο εφαρμογές που πρέπει να επικοινωνήσουν, η Α και η Β. Η εφαρμογή Α μέσω του προσαρμογέα Α, που είναι ο προσαρμογέας πηγή, στέλνει ένα μήνυμα της μορφή Μ. Το κανάλι αναλαμβάνει την σωστή μεταφορά του προς την προσαρμογέα Β και συγχρόνως μετατρέπει το μήνυμα σε μορφή Μ'. Το μήνυμα Μ' καταλήγει στον προσαρμογέα Β, που ονομάζεται και προσαρμογέας προορισμός, ο οποίος γνωρίζει πως να μεταφέρει το μήνυμα στην εφαρμογή Β.

Σχήμα 1: Μηνυματοστρεφής προσέγγιση

Χρησιμοποιώντας την παραπάνω μηνυματοστραφή αρχιτεκτονική οι εφαρμογές Α και Β παραμένουν ελαφρά διασυνδεδεμένες. Για παράδειγμα η μορφή του

μηνύματος Μ της εφαρμογής Α μπορεί να αλλάξει χωρίς να επηρεαστεί η εφαρμογή Β, η επικοινωνία θα συνεχίσει να υφίσταται εφόσον το κανάλι μπορεί να μετατρέψει το μήνυμα στην μορφή Μ' που είναι αποδεκτή από τον προσαρμογέα Β.

2.1.5 Πρότυπα αλληλεπίδρασης

Η σύγχρονη επικοινωνία μεταξύ αίτησης και απάντησης δεν προσφέρει την απαραίτητη αλληλεπίδραση χαμηλής διασύνδεσης που απαιτούν οι ανάγκες των σύγχρονων κατανεμημένων συστημάτων. Για αυτό θα πρέπει η προσφερόμενη τεχνολογία να υλοποιεί διαφόρων τύπων ασύγχρονη επικοινωνία.

Για παράδειγμα η εφαρμογή που στέλνει την αίτηση θα πρέπει μετά την αποστολή να απελευθερώνεται και να μην περιμένει την απάντηση στο αίτημά της. Ακόμη δεν θα πρέπει να ασχολείται με το αν υπάρχει πρόβλημα σύνδεση, αλλά το κανάλι θα πρέπει να εγγυάται την σωστή μεταφορά του μηνύματος ακόμη και αν απαιτηθεί επαναποστολή του.

Μια άλλη λειτουργία που πρέπει να υλοποιηθεί είναι η αποστολή του μηνύματος σε πολλούς προορισμούς συγχρόνως. Έτσι επιτυγχάνεται ακόμη πιο χαλαρή διασυνδεσιμότητα καθώς ο αποστολέας δεν γνωρίζει το ποιος αποδέκτης θα απαντήσει στην αίτηση του.

Ακόμη θα πρέπει να υποστηρίζεται λειτουργικότητα συναλλαγών και προδιαγραφές ασφαλείας. Μια συναλλαγή μπορεί να αποτελείται από πολλαπλά ασύγχρονα αιτήματα τα οποία θα πρέπει να διεκπεραιωθούν στο σύνολό τους.

2.2 Τι είναι μια υπηρεσία

2.2.1 Η υπηρεσία στην καθημερινή ζωή

Στην καθημερινή μας ζωή συναντάμε την έννοια της υπηρεσίας γύρω μας συνεχώς. Χρησιμοποιούμε υπηρεσίες παροχής ηλεκτρικού, ύδρευσης, σταθερής και κινητής τηλεφωνίας, μέσων μαζικής μεταφοράς κτλ. Κάθε υπηρεσία που χρησιμοποιούμε έχει κάποια βασικά χαρακτηριστικά. Πρώτα απ' όλα μπορούμε να θεωρήσουμε μια υπηρεσία ως ένα πακέτο λειτουργικότητας που είναι δημόσια γνωστό. Ο καταναλωτής δεν χρειάζεται να γνωρίζει τον τρόπο με τον οποίο οι υπηρεσίες παράγονται και παρέχονται, για παράδειγμα κάποιος που καταναλώνει ηλεκτρικό ρεύμα δεν χρειάζεται να γνωρίζει πως παράγεται και μεταφέρεται, απλά το χρησιμοποιεί. Συνήθως ο καταναλωτής και ο παροχέας έχουν συμφωνήσει τα χαρακτηριστικά της υπηρεσίας, όπως η διαθεσιμότητα και το κόστος της. Οι υπηρεσίες δημοσιεύονται για να προσελκύσουν καταναλωτές και είναι ανακτήσιμες με βάση τα χαρακτηριστικά τους και την περιγραφή που έχουν δημοσιεύσει. Τέλος κάποιος μπορεί να συνθέσει διάφορες υπηρεσίες δημιουργώντας ένα πακέτο υπηρεσιών, για παράδειγμα παρέχονται πακέτα σταθερής τηλεφωνίας, κινητής τηλεφωνίας και σύνδεσης στο διαδίκτυο.

Τα παραπάνω χαρακτηριστικά των υπηρεσιών μεταφέρονται σε επίπεδο λογισμικού.

2.2.2 Η υπηρεσία ως λογισμικό – Software as a Service (SaaS)

Το υπηρεσιοστρεφές μοντέλο δόμησης λογισμικού αποτελεί μια εξέλιξη στον σχεδιασμό λογισμικού. Αρχικά η ιδέα της δημιουργίας δομικών μονάδων λογισμικού με χρηστικότητα που ξεπερνούσε τα όρια μιας συγκεκριμένης εφαρμογής υλοποιήθηκε με τον σχεδιασμό functions και την ομαδοποίηση τους σε πακέτα λογισμικού (packages). Στην συνέχεια παρουσιάστηκε ο

αντικειμενοστραφής προγραμματισμός που εισήγαγε τις έννοιες των αντικειμένων και των κλάσεων. Με την χρήση της κληρονομικότητας, του πολυμορφισμού και των άλλων ιδιοτήτων των κλάσεων επιτεύχθηκε η απαιτούμενη ευελιξία και επαναχρησιμοποίηση των κλάσεων. Τα συστατικά λογισμικού (components) ακολούθησαν παρέχοντας την δυνατότητα επαναχρησιμοποίησης συγκεκριμένης λειτουργικότητας που μπορούσε να παραμετροποιηθεί μέσω των ιδιοτήτων του συστατικού. Ένα επιπλέον επίπεδο αφαιρετικότητας στα παραπάνω μοντέλα δόμησης μονάδων λογισμικού προσφέρεται με την χρήση υπηρεσιών, επιτρέποντας στον χρήστη καταναλωτή να αγνοεί λεπτομέρειες της μονάδας λογισμικού που παρέχει την υπηρεσία και τον τρόπο υλοποίησής της (Weerawarana 2005).

2.2.3 Δημοσίευση υπηρεσιών

Μια υπηρεσία είναι διαθέσιμη σε μια συγκεκριμένη διεύθυνση του δικτύου. Η υπηρεσία δέχεται μηνύματα, στέλνει μηνύματα και προσφέρει λειτουργικότητα σύμφωνη με τις προδιαγραφές της. Η λειτουργικότητα της υπηρεσίας, καθώς και οι πολιτικές χρήσης της, οι προϋποθέσεις και οι περιορισμοί της περιγράφονται με την χρήση ενός XML αρχείου και της περιγραφικής γλώσσας WSDL. Το αρχείο αυτό δημοσιεύεται έτσι ώστε οι χρήστες καταναλωτές να έχουν όλες τις απαραίτητες πληροφορίες που χρειάζονται για να συνδεθούν και να κάνουν χρήση της λειτουργικότητάς της. Ακόμη δημοσιεύονται η σημασιολογική περιγραφή της υπηρεσίας, και κάθε απαραίτητη πληροφορία που χρειάζεται το περιβάλλον των καταναλωτών για να συνδεθεί, όπως το πρωτόκολλο μεταφοράς και πληροφορίες κρυπτογράφησης.

2.2.4 Σύνθεση υπηρεσιών για δημιουργία λύσεων

Ένα από τα πλεονεκτήματα του υπηρεσιοστρεφούς προσέγγισης είναι ότι προσφέρεται η δυνατότητα δημιουργίας νέων υπηρεσιών με χρήση υπαρχόντων.

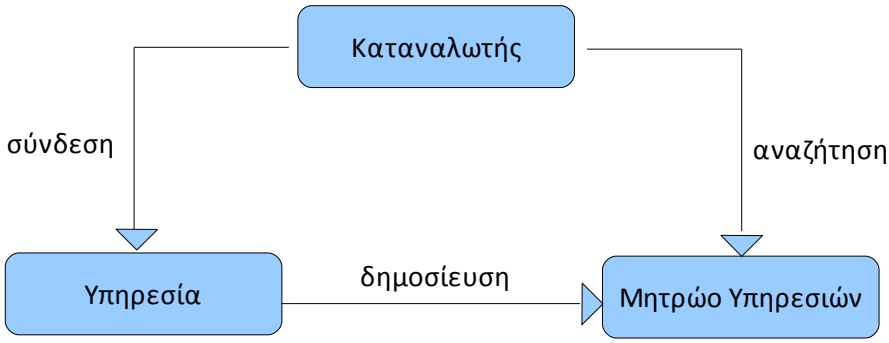
Η τεχνολογία της “Χωρογραφίας” (choreography) υλοποιεί την σύνθεση και οργάνωση υπηρεσιών σε νέες υπηρεσίες με περισσότερη λειτουργικότητα. Καθορίζει ποιες υπηρεσίες θα χρησιμοποιηθούν, με ποια σειρά και σύμφωνα με ποιες συνθήκες. Αν θεωρήσουμε ότι μια επιχειρηματική διεργασία είναι μια σειρά διακριτών λειτουργιών, με την χρήση της χωρογραφίας είναι δυνατόν να δημιουργηθεί μια υπηρεσία που θα εμπεριέχει άλλες υπηρεσίας για υλοποίηση της κάθε λειτουργίας της διεργασίας. Καθώς μια το αποτέλεσμα μιας χωρογραφίας είναι διατίθεται ως υπηρεσία, μπορούν να δημιουργηθούν χωρογραφίες που θα συνθέτουν χωρογραφίες και υπηρεσίες μαζί. Επειδή με αυτόν τον τρόπο πετυχαίνεται η μεταφορά μιας επιχειρηματικής διαδικασίας, η τελική εφαρμογή ονομάζεται και “Λύση” του επιχειρηματικού προβλήματος. Μια τέτοια λύση περιέχει εκτός της χωρογραφίας υπηρεσιών και άλλα στοιχεία όπως μια διεπαφή με τον χρήστη και επιπλέον κανόνες και συνθήκες λειτουργίας.

2.3 Βασικές στοιχεία της Υπηρεσιοστρεφούς Αρχιτεκτονικής (SOA) και των Υπηρεσιών Ιστού (Web Services)

Παρόλο που η υπηρεσιοστρεφής αρχιτεκτονική SOA, ως αρχιτεκτονική θεώρηση, είναι ανεξάρτητη από οποιαδήποτε συγκεκριμένη τεχνολογία συνηθίζεται για την μελέτη της να χρησιμοποιείται η υλοποίησή της με υπηρεσίες ιστού (Web Services). Σε κάθε περίπτωση όμως πρέπει να γίνει ξεκάθαρο ότι η SOA και οι υπηρεσίες ιστού δεν είναι συνώνυμες έννοιες. Οι υπηρεσίες ιστού παρόλαυτά αποτελούν την κύρια υλοποίησή της SOA, και τα πρότυπά τους καλύπτουν το βασικό υπόβαθρο για την επίτευξη διαλειτουργικότητας, ποιότητας υπηρεσίας, σημασιολογικής περιγραφής του συστήματος, ασφάλειας, και διαχείρισης και αξιόπιστης ανταλλαγής μηνυμάτων.

Υπάρχουν τρεις βασικές έννοιες στην SOA, ο πάροχος της υπηρεσίας, το μητρώο υπηρεσιών και ο πελάτης υπηρεσίας, με τρεις βασικές λειτουργίες τη Δημοσίευση,

την Αναζήτηση και την Σύνδεση (Publish/Find/Bind). Ο πάροχος υπηρεσιών δημιουργεί την λειτουργικότητα που επιθυμεί να κοινοποιήσει και στην συνέχεια την δημοσιεύει στο μητρώο υπηρεσιών (Publish). Ο πελάτης αναζητά (Find) μια υπηρεσία με συγκεκριμένα χαρακτηριστικά στο μητρώο υπηρεσιών και στην συνέχεια θέτει το αίτημά του στην επιλεγμένη υπηρεσία (Bind) (Weerawarana 2005).



Σχήμα 2:Δημοσίευση /Αναζήτηση/Σύνδεση

Αναλυτικότερα, ο παροχέας υπηρεσίας πρέπει πρώτα να δημιουργήσει μια περιληπτική περιγραφή της υπηρεσίας που θα περιέχει τον τρόπο με τον οποίο κάποιος μπορεί να την χρησιμοποιήσει. Έπειτα πρέπει να την δημοσιεύσει στο μητρώο υπηρεσιών έτσι ώστε οι καταναλωτές υπηρεσιών να είναι σε θέση να γνωρίζουν την λειτουργικότητα της υπηρεσίες και τον τρόπο διασύνδεσης σε αυτή. Οι καταναλωτές πρέπει να έχουν έναν τρόπο αναζήτησης υπηρεσιών για να μπορέσουν να διαλέξουν την κατάλληλη υπηρεσία που χρειάζονται, και τέλος χρησιμοποιώντας την πληροφορία που ανέκτησαν να καταναλώσουν την υπηρεσία.

Για να επιτευχθεί η παραπάνω λειτουργικότητα γίνεται χρήση των βασικών προδιαγραφών των υπηρεσιών ιστού όπως το SOAP, η WSDL, το UDDI και η BPEL.

ΕΠΙΛΟΓΟΣ

Παραπάνω παρουσιάστηκε η μετάβαση των τεχνολογιών κατασκευής δομημένου λογισμικού και κατανεμημένων συστημάτων, και η επέκτασή τους με την χρήση της υπηρεσιοστρεφούς αρχιτεκτονικής και των υπηρεσιών ιστού. Στο επόμενο κεφάλαιο θα παρουσιαστεί η δομή και ο τρόπος σύνθεσης του WSDL αρχείου για την δημιουργία υπηρεσιών ιστού.

ΚΕΦΑΛΑΙΟ 3

ΕΙΣΑΓΩΓΗ ΣΤΗΝ WSDL

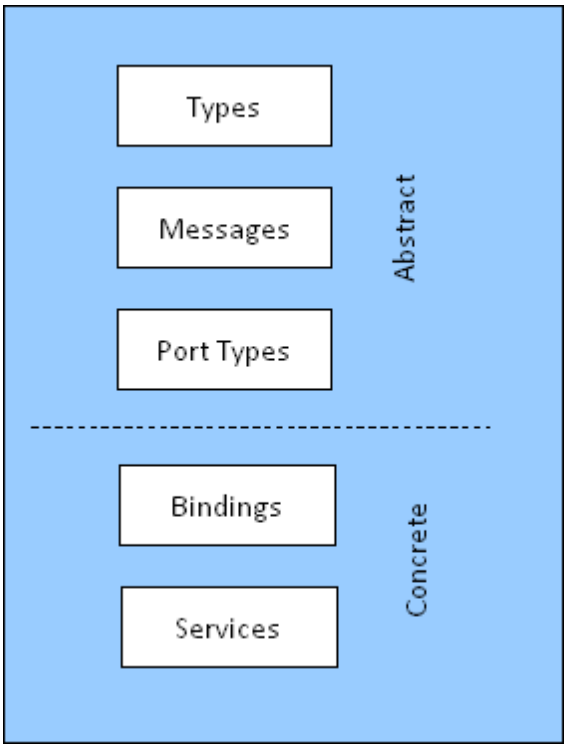
ΕΙΣΑΓΩΓΗ

Η WSDL είναι μια περιγραφική γλώσσα βασισμένη στην XML. Με την χρήση της ορίζεται το πως μπορεί κάποιος να έχει πρόσβαση σε μια υπηρεσία, ποιες λειτουργίες είναι διαθέσιμες, και ποια μηνύματα και με ποια δομή ανταλλάσσονται στην επικοινωνία μεταξύ του καταναλωτή και του παρόχου της υπηρεσίας (Richards 2006) (Web Services Description Language (WSDL)).

Για την καλύτερη κατανόηση της δομής και χρήσης ενός WSDL εγγράφου θα χρησιμοποιηθούν αποσπάσματα του VirtueMart_Products.wsdl που ασχολούνται με την ανάκτηση και την ενημέρωση των “Κατασκευαστών”. Η έκδοση της WSDL που χρησιμοποιείται είναι η WSDL 1.1.

3.1 Δομή της WSDL

Η δομή ενός WSDL αρχείο χωρίζεται σε πέντε τμήματα, που χωρίζονται σε δύο μέρη. Τα τμήματα “Types”, “Messages”, “PortTypes” απαρτίζουν το “Abstract” μέρος του αρχείου, ενώ τα τμήματα “Bindings” και “Services” απαρτίζουν το “Concrete” μέρος τους. Το παρακάτω σχήμα παρουσιάζει την δομή ενός WSDL αρχείου.



Σχήμα 3: Η δομή των WSDL αρχείων

Η δομή του αρχείου σε XML,

```
<?xml version ='1.0' encoding ='UTF-8' ?>
<definitions name='VirtueMart_Products'
  xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
     <types>
     </types>
     <message name="ManufacturersRequest">
     </message>
     <portType name="VirtueMart_Products_Service">
     </portType>
     <binding name="VirtueMart_Products_SOAP"
type="tns:VirtueMart_Products_Service">
     </binding>
```

```
    <service name="VirtueMart_Products_Srv">
    </service>
</definitions>
```

Αρχικά ορίζεται η έκδοση xml καθώς και η κωδικοσελίδα που θα χρησιμοποιηθούν για την σύνταξη του αρχείου

```
<?xml version ='1.0' encoding ='UTF-8' ?>
```

3.2 <definitions>

Το στοιχείο <definitions> περικλύει όλους τους ορισμούς που αφορούν μια συγκεκριμένη περιοχή ονομάτων, που ορίζεται με την ιδιότητα “targetNamespace”. Για να χρησιμοποιηθούν στοιχεία από άλλες περιοχές ονομάτων πρέπει να υπάρξει ορισμός τους με την ιδιότητα “xmlns”. Για να αποφευθούν πιθανές διενέξεις ονομάτων στοιχείων συγχρόνως προσδίδονται προθέματα για κάθε περιοχή ονομάτων χρησιμοποιώντας την “xmlns:*prefix*="*URI*"” σύνταξη όπως παρακάτω:

```
<wsdl:definitions name='VirtueMart_Products'
  targetNamespace='http://www.my-uni-
project.info/joomla/components/com_virtuemart_soa_services/VirtueM
art_Products'
  xmlns:tns='http://www.my-uni-
project.info/joomla/components/com_virtuemart_soa_services/VirtueM
art_Products'
  xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
  xmlns:xsd='http://www.w3.org/2001/XMLSchema'
  xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
  xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" >
```

3.3 <types>

Το στοιχείο <types> χρησιμοποιείται για την περιγραφή των δομών δεδομένων που θα χρησιμοποιηθούν αργότερα κατά τον ορισμό των μηνυμάτων. Για τον ορισμό των δομών μπορούν να χρησιμοποιηθούν διάφορες γλώσσες όπως η XML Schema, η RelaxNG και DTD, ή ακόμη και γλώσσες που δεν ορίζουν XML δομές όπως η MIME και η OMG IDL. Για να γίνει δυνατή η χρήση της XML Schema ορίζεται το στοιχείο <xsd:schema> (η XML Schema θα χρησιμοποιηθεί στα παραδείγματά σε αυτό το κεφάλαιο).

```
<wsdl:types>
     <xsd:schema elementFormDefault="qualified"
targetNamespace='http://www.my-uni-project.info/joomla/components/
com_virtuemart_soa_services/VirtueMart_Products' >
```

Στην συνέχεια ορίζονται οι δομές δεδομένων. Στο παράδειγμα που ακολουθεί ορίζονται οι παράμετροι που χρειάζονται για την σύνδεση στην υπηρεσία:

```
<xsd:complexType name="LoginParams">
     <xsd:sequence>
          <xsd:element minOccurs="1" maxOccurs="1" name="User"
          type="xsd:string"/>
          <xsd:element minOccurs="1" maxOccurs="1"
          name="Password"  type="xsd:string"/>
     </xsd:sequence>
</xsd:complexType>
```

Οι παράμετροι αποτελούνται από ένα στοιχείο “User” και ένα στοιχείο “Passwod”, και τα δύο είναι τύπου “string” και είναι υποχρεωτικά.

Παρόμοια μπορούν να οριστούν αντικείμενα. Για παράδειγμα το αντικείμενο “Κατασκευαστής” (Manufacturer):

```
<xsd:complexType name="Manufacturer">
```

```
     <xsd:sequence>
          <xsd:element minOccurs="1" maxOccurs="1"
           name="Manufacturer_Id" type="xsd:integer" />
          <xsd:element minOccurs="1" maxOccurs="1" name="Name"
           type="xsd:string"  />
     </xsd:sequence>
</xsd:complexType>
```

Ακόμη μπορούν να οριστούν πίνακες. Ο ορισμός πινάκων βασικών τύπων και πινάκων άλλων δομών γίνεται με παρόμοιο τρόπο:

```
<xsd:complexType name="ArrayOfActionStatus">
     <xsd:sequence>
          <xsd:element minOccurs="0" maxOccurs="unbounded"
          name="actionStatus" nillable="true"
          type="xsd:string" />
     </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="ArrayOfManufacturers">
     <xsd:sequence>
          <xsd:element minOccurs="0" maxOccurs="unbounded"
          name="Manufacturer" nillable="true"
          type="tns:Manufacturer" />
     </xsd:sequence>
</xsd:complexType>
```

Στην ιδιότητα “type” κάνουμε αναφορά του προθέματος της περιοχής ονομάτων που χρησιμοποιείται. Στο πρώτο παράδειγμα είναι “xsd:” δηλαδή XML Schema, ενώ στο δεύτερο αναφέρεται στο τρέχουσα περιοχή ονομάτων (target namespace) με την χρήση του προθέματος “tns:”.

Στα προηγούμενα τέσσερα παραδείγματα έχουν οριστεί δομές με την χρήση του στοιχείου <xsd:compexType> που μπορούν να χρησιμοποιηθούν μόνο μέσα στο τρέχον τμήμα του αρχείου WSDL, δηλαδή το <types>. Για τον ορισμό δομών που θα είναι διαθέσιμες και στα υπόλοιπα τμήματα του WSDL αρχείου πρέπει να χρησιμοποιηθεί το στοιχείο <xsd:element> που θα περικλείει το <xsd:compexType>.

```
<xsd:element name="ManufacturersLogin">
     <xsd:complexType>
          <xsd:sequence>
               <xsd:element  minOccurs="1" maxOccurs="1"
               name="LoginParams" type="tns:LoginParams"/>
          </xsd:sequence>
     </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="Manufacturers">
     <xsd:complexType>
          <xsd:sequence>
               <xsd:element  minOccurs="1" maxOccurs="1"
               name="ManufacturerList"
               type="tns:ArrayOfManufacturers"/>
          </xsd:sequence>
     </xsd:complexType>
</xsd:element>
```

Ακολουθώντας τις παραπάνω αρχές μπορούν να οριστούν όσο πολύπλοκες δομές χρειάζονται. Στο παρακάτω παράδειγμα ορίζεται το στοιχείο “ModifyManufacturersLogin” που περιέχει ένα στοιχείο τύπου “LoginParams”, ένα στοιχείο τύπου “string” και ένα στοιχείο τύπου ArrayOfManufacturers που είναι ένας πίνακας αντικειμένων “Manufacturers”.

```
<xsd:element name="ModifyManufacturersLogin">
     <xsd:complexType>
          <xsd:sequence>
               <xsd:element  minOccurs="0" maxOccurs="1"
               name="LoginParams" type="tns:LoginParams"/>
               <xsd:element  minOccurs="0" maxOccurs="1"
               name="ModType" type="xsd:string"/>
               <xsd:element  minOccurs="0" maxOccurs="1"
               name="ManufacturersToModify"
               type="tns:ArrayOfManufacturers"/>
          </xsd:sequence>
     </xsd:complexType>
</xsd:element>
```

3.4 <messages>

Το στοιχείο <messages> χρησιμοποιείται για να οριστεί η δομή των μηνυμάτων που ανταλλάσσονται μεταξύ του καταναλωτή και της υπηρεσίας. Η σύνταξη αυτού του στοιχείου εξαρτάται από τον τύπο διασύνδεσης που θα οριστεί αργότερα στο αρχείο με το στοιχείο <binding>, στο παράδειγμά επιλέχθηκε ο τύπος “Document/literal”. Συνεχίζοντας τα παραπάνω παραδείγματα θα οριστούν δύο μηνύματα, το ένα μήνυμα θα μεταφέρει τις παραμέτρους σύνδεσης προς την υπηρεσία και το άλλο θα είναι η απάντηση της υπηρεσίας προς τον καταναλωτή, δηλαδή μια λίστα με τους κατασκευαστές.

```
<wsdl:message name="ManufacturersRequest">
     <wsdl:part name="parameters"
     element="tns:ManufacturersLogin" />
</wsdl:message>

<wsdl:message name="ManufacturersResponse">
     <wsdl:part name="results" element="tns:Manufacturers" />
```

```
</wsdl:message>
```

Σε αυτό το σημείο ορίζονται τα μηνύματα χωρίς να αναφέρεται η χρήση τους που θα οριστεί αργότερα σε άλλο στοιχείο του WSDL αρχείου. Οι περιορισμοί που υπάρχουν για τον ορισμό των μηνυμάτων είναι οι παρακάτω:

- δεν πρέπει να υπάρχουν μηνύματα με το ίδιο όνομα “name”
- δεν πρέπει να υπάρχουν στοιχεία <part> με το ίδιο όνομα “name” μέσα σε ένα μήνυμα
- όταν χρησιμοποιείται τύπος διασύνδεσης “Document/literal” η ιδιότητα “element” είναι υποχρεωτική (στην περίπτωση που γίνεται χρήση διασύνδεσης τύπου “RPC/encoded” τότε η ιδιότητα “type” είναι υποχρεωτική και η “element” δεν χρησιμοποιείται)
- παρόλο που σύμφωνα με το πρότυπο WS-I επιτρέπεται η χρήση περισσότερων του ενός στοιχείου <part> σε ένα μήνυμα έχει γίνει κοινά αποδεκτό ότι χρησιμοποιείται μόνο ένα (για αυτόν τον λόγο όταν χρειαζόμαστε να περάσουμε μηνύματα που έχουν πολλά στοιχεία φροντίζουμε να δημιουργήσουμε το κατάλληλο στοιχείο <element> στο τμήμα <types> του WSDL αρχείου). Η λόγος είναι επειδή το SOAP μήνυμα που δημιουργείται πρέπει να έχει μόνο ένα στοιχείο παιδί (πάλι σύμφωνα με το WS-I πρότυπο). Περισσότερη ανάλυση για το πως δημιουργούνται τα SOAP μηνύματα ανάλογα με το τύπο διασύνδεσης που επιλέγουμε υπάρχει παρακάτω στην παράγραφο που αναλύουμε το στοιχείο <binding>
- Η ιδιότητα “element” του στοιχείου <part> μπορεί να είναι είτε απλού τύπου της XML Schema, είτε κάποιου τύπου που έχει οριστεί στο τμήμα <types> με το στοιχείο <element>.

Υπάρχει δυνατότητα να οριστεί ένα κενό μήνυμα, για παράδειγμα για να πραγματοποιηθεί μια κλήση σε μια υπηρεσία χωρίς να περάσει καμία παράμετρος. Σε αυτή την περίπτωση απλά δεν ορίζεται κανένα <part> στοιχείο.

3.5 <portType>

Με το στοιχείο <portType> ορίζονται οι λειτουργίες που είναι διαθέσιμες από την υπηρεσία μαζί με τα μηνύματα που θα ανταλλαχθούν με κάθε μία από αυτές. Για τον ορισμό κάθε λειτουργίας χρησιμοποιείται το στοιχείο <operation>. Η ιδιότητα “name” του στοιχείου <operation> πρέπει να είναι μοναδική, κάθε στοιχείο εσωκλείει τα μηνύματα που θα χρησιμοποιηθούν από την λειτουργία με την χρήση των στοιχείων <input> για μηνύματα που εισαγωγής της υπηρεσίας, <output> για μηνύματα που στέλνει η υπηρεσία προς τον καταναλωτή, και <fault> που μεταφέρει μηνύματα λάθους σε επίπεδο εφαρμογής της υπηρεσίας. Δεν υπάρχει όριο στο πόσες λειτουργίες μπορούν να οριστούν σε ένα στοιχείο <portType>. Επιστρέφοντας στο παράδειγμά, θα ορισθεί η λειτουργία “getManufacturers” που θα ανακτά τους κατασκευαστές. Το μήνυμα που πρέπει να σταλεί είναι το “ManufacturersRequest” και αυτό που θα επιστραφεί είναι το “ModifyManufacturersResponse”, τα οποία ορίστηκαν παραπάνω.

```
<wsdl:portType name="VirtueMart_Products_Service">
     <wsdl:operation name="getManufacturers">
          <wsdl:input  message="tns:ManufacturersRequest"/>
          <wsdl:output message="tns:ManufacturersResponse"/>
     </wsdl:operation>
</wsdl:portType>
```

Οι λειτουργίες χωρίζονται σε τέσσερις ομάδες ανάλογα με το είδος των μηνυμάτων που εμπεριέχουν:

- Απλής μετάβασης

- Αμφίδρομες
- Αναζήτηση απάντησης
- Κοινοποίησης

Οι δύο πρώτοι τύποι λειτουργιών είναι οι πιο διαδεδομένοι και οι μόνοι από τους τέσσερις που υποστηρίζονται από το πρότυπο WS-I Basic Profile (το πρότυπο WS-Addressing καλύπτει και τους άλλους δύο τύπους που μπορούν να είναι ιδιαίτερα χρήσιμοι όταν γίνεται ασύγχρονη επικοινωνία ή επικοινωνία μεταξύ υπηρεσιών).

Οι λειτουργίες απλής μετάβασης εμπεριέχουν μόνο <input> μηνύματα και χρησιμοποιούνται όταν ο καταναλωτής στέλνει κάποιο μήνυμα στην υπηρεσία χωρίς να περιμένει κάποια απάντηση από αυτή ούτε κάποιο μήνυμα επιβεβαίωσης λήψης των δεδομένων. Συνήθως χρησιμοποιείται για μεταφορά μη σημαντικών δεδομένων. Σε μια τέτοια περίπτωση η σύνταξη θα ήταν:

```
<wsdl:operation name="myOperation">
     <wsdl:input  message="tns:myRequest"/>
</wsdl:operation>
```

Οι αμφίδρομες λειτουργίες είναι ο τύπος που χρησιμοποιείται περισσότερο. Ο καταναλωτής πραγματοποιεί μια κλήση στέλνοντας ένα μήνυμα και περιμένει την απάντηση της υπηρεσίας. Σε αυτόν τον τύπο λειτουργίας ορίζονται μηνύματα τύπου <input> και <output>, ενώ προαιρετικά μπορεί να οριστούν και μηνύματα τύπου <fault>:

```
<wsdl:operation name="myOperation">
     <wsdl:input  message="tns:myRequest"/>
     <wsdl:output  message="tns:myResponse"/>
```

```
     <wsdl:fault  message="tns:myFault"/>
</wsdl:operation>
```

Σε μια λειτουργία μπορούν να οριστούν περισσότερα του ενός στοιχεία <fault> με μόνο περιορισμό να μην υπάρχουν <fault> στοιχεία με το ίδιο όνομα “name” στην ίδια λειτουργία (η ιδιότητα “name” για τα <fault> στοιχεία είναι υποχρεωτική, ενώ για τα <input> και <output> είναι προαιρετική). Η χρήση τους βοηθάει στο να επιστραφούν μηνύματα σφαλμάτων που προκύπτουν κατά την εκτέλεση της υπηρεσία και να ενημερωθεί κατάλληλα ο καταναλωτής.

3.6 <binding>

Με τα στοιχεία που έχουν αναλυθεί μέχρι στιγμής έχουν περιγραφεί οι δομές, τα μηνύματα και οι λειτουργίες μια υπηρεσίας, όπως επίσης και το ποια μηνύματα χρησιμοποιούνται από κάθε λειτουργία. Τα στοιχεία <types>, <messages> και <portTypes> ανήκουν στο “αφηρημένο” (abstract) τμήμα του WSDL αρχείου, ενώ τα επόμενα στοιχεία <binding> και <services> ανήκουν στο “συγκεκριμένο” (concrete) τμήμα. Στο “αφηρημένο” τμήμα περιγράφεται το “Τι” κάνει η υπηρεσία, ενώ το “συγκεκριμένο” τμήμα ορίζει το “Πως” με το στοιχείο <binding>, και το “Που” με το στοιχείο <service> που θα αναλυθεί παρακάτω.

Το στοιχείο <binding> ορίζει το πως θα μορφοποιηθούν τα μηνύματα που έχουν ήδη οριστεί παραπάνω και ποιο πρωτόκολλο θα χρησιμοποιηθεί για την μεταφορά τους. Η WSDL υποστηρίζει τρεις τρόπους μορφοποίησης με την χρήση SOAP, HTTP και MIME. Θα αναλυθεί μόνο η SOAP μορφοποίησης. Με την χρήση της SOAP υπάρχει η δυνατότητα ορισμού δύο τύπων μορφοποιήσης “operation styles” του “Document” και του “RPC”, με έναν περαιτέρω διαχωρισμό μορφοποίησης μεταξύ “encoded” και “literal”.

3.6.1 RPC/encoded

Όπως αναφέρθηκε παραπάνω η σύνταξη των μηνυμάτων που ορίζονται με το <message> στοιχείο εξαρτάται από τον τύπο μορφοποίησης <binding> που χρησιμοποιείται. Αν είχε επιλεχθεί η RPC/Encoded μορφοποίηση θα μπορούσε να οριστεί το μήνυμα “ManufacturersRequest” ως εξής:

```
<wsdl:message name="ManufacturersRequest">
     <wsdl:part name="User" type="xsd:string"/>
     <wsdl:part name="Password" type="xsd:string"/>
</wsdl:message>
```

Δηλαδή ότι το “ManufacturersRequest” μήνυμα έχει δύο παραμέτρους την “User” και την “Password” τύπου “string”.

Όταν δημιουργείται το SOAP μήνυμα από τον παραπάνω ορισμό η XML δομή που δημιουργείται χρησιμοποιεί τα ονόματα των παραμέτρων ως ονόματα στοιχείων που εσωκλείουν τις τιμές τους, εν συνεχεία τα στοιχεία αυτά περικλείονται από ένα στοιχείο με όνομα το όνομα του στοιχείου <binding>. Αν υποθέσουμε ότι ο χρήσης είναι ο “admin”, ο κωδικός “1234” και το όνομα “name” του στοιχείου <binding> “VirtueMart_Products_SOAP” τότε το μήνυμα που θα δημιουργηθεί θα είναι:

```
<VirtueMart_Products_SOAP>
     <User xsi:type=“xsd:string”>admin</User>
     <Password xsi:type=“xsd:string”>1234</Password>
</VirtueMart_Products_SOAP>
```

3.6.2 RPC/literal

Η σύνταξη του μηνύματος για την μορφοποίηση RPC/literal είναι ίδια με την μορφοποίηση RPC/encoded που περιγράφτηκε στην προηγούμενη παράγραφο. Το μήνυμα όμως που δημιουργείται διαφέρει επειδή δεν συμπεριλαμβάνει τον τύπο των δεδομένων. Το μήνυμα του παραπάνω παραδείγματος θα ήταν:

```
<VirtueMart_Products_SOAP>
     <User>admin</User>
     <Password>1234</Password>
</VirtueMart_Products_SOAP>
```

Η εμφανής διαφορά μεταξύ των δύο μορφών είναι ότι με την RPC/literal μεταφέρονται λιγότερα δεδομένα πάνω από το μέσω μεταφοράς κερδίζοντας σε ταχύτητα και οικονομία μέσου. Η σημαντικότερη διαφορά όμως είναι ότι η μορφή RPC/literal είναι σύμφωνη με το πρότυπο WS-I, ενώ η RPC/encoded δεν είναι. Έτσι με την χρήση της RPC/literal μορφοποίησης μεγιστοποιείται η διαλειτουργικότητα της υπηρεσίας.

3.6.3 Document/encoded

Η μορφοποίηση Document/encoded δεν υποστηρίζεται από το πρότυπο WS-I και συνήθως δεν χρησιμοποιείται οπότε δεν θα αναλυθεί.

3.6.4 Document/literal

Η μορφοποίηση Document/literal είναι η μορφοποίηση που χρησιμοποιείται συχνότερα (επιπλέον κατά τον σχεδιασμό της WSDL 2.0 θεωρήθηκε ότι αυτή η μορφοποίηση θα κυριαρχήσει στον χώρο). Η σύνταξη του στοιχείου <message> είναι παρόμοια με την σύνταξη στην μορφοποίηση RPC μόνο που αντί της ιδιότητας “type” χρησιμοποιείτε η ιδιότητα “element”, η τιμή της οποίας αναφέρεται σε δομή που έχει οριστεί στο τμήμα <types> με την χρήση του στοιχείου

<element>. Λόγο περιορισμού από το πρότυπο WS-I σχετικά με την μορφοποίηση των SOAP μηνυμάτων επιτρέπεται μόνο ένα στοιχείο <part> σε κάθε μήνυμα.

```
<wsdl:message name="ManufacturersRequest">
     <wsdl:part name="parameters"
          element="tns:ManufacturersLogin" />
</wsdl:message>
```

Όταν δημιουργείται το μήνυμα που θα αποσταλεί στην υπηρεσία δημιουργούνται στοιχεία με το όνομα των στοιχείων της δομής που ορίζεται στην ιδιότητα “element” και εσωκλείονται σε ένα στοιχείο με το όνομα της δομής. Για το παραπάνω ορισμό θα δημιουργηθεί το εξής μήνυμα:

```
<ManufacturersLogin>
     <User>admin</User>
     <Password>1234</Password>
</ManufacturersLogin>
```

3.6.5 Σύνταξη του στοιχείου <binding>

Αρχικά ορίζεται το όνομα του στοιχείου <binding> με την ιδιότητα “name”. Το όνομα αυτό πρέπει να είναι μοναδικό ανάμεσα στα στοιχεία <binding> που ορίζονται, αν και το πιο πιθανόν είναι ότι θα οριστεί μόνο ένα στοιχείο <binding> σε κάθε WSDL αρχείο. Με την ιδιότητα “type” ορίζεται σε ποιο στοιχείο “portType” αναφέρεται. Επιστρέφοντας στο αρχικό παράδειγμα:

```
<wsdl:binding name="VirtueMart_Products_SOAP"
 type="tns:VirtueMart_Products_Service">
     ...
     ...
</wsdl:binding>
```

Στο εσωτερικό της σύνταξης ορίζεται το στοιχείο <soap:binding> με την παρακάτω μορφή:

```
<soap:binding transport="uri" style="rpc|document"?>
```

Η ιδιότητα "transport" καθορίζει το πρωτόκολλο μεταφοράς που θα χρησιμοποιηθεί. Τις περισσότερες φορές θα είναι HTTP που περιγράφεται από την τιμή "http://schemas.xmlsoap.org/soap/http". Άλλα πρωτόκολλα που υποστηρίζονται είναι τα SMTP και FTP. Με την ιδιότητα "style" ορίζεται ο προεπιλεγμένος τύπος μορφοποίησης του μηνύματος όπως αναλύθηκε παραπάνω. Αν δεν οριστεί η ιδιότητα "style" τότε θεωρείται ότι έχει την τιμή "document". Στο παράδειγμά που ακολουθούν επιλέχθηκε ως πρωτόκολλο μεταφοράς το HTTP και τύπο μορφοποίησης η "document" επιλέγοντας να μην χρησιμοποιηθεί η ιδιότητα "style":

```
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
```

Στην συνέχεια ορίζεται το στοιχείο <wsdl:operation> το οποίο περιγράφει σε ποια λειτουργία αντιστοιχεί ο ορισμός. Η συσχέτιση επιτυγχάνεται με την ιδιότητα "name" που πρέπει να έχει την ίδια τιμή με ένα στοιχείο <operation> που έχει οριστεί στο τμήμα <portType>.

```
<wsdl:operation name="getManufacturers">
     ...
     ...
</wsdl:operation>
```

Εσωτερικά του <wsdl:operation> ορίζεται το στοιχείο <soap:operation> το οποίο έχει δύο ιδιότητες, την "soapAction" και την "style". Με την "soapAction" ορίζεται η τιμή που θα έχει η "SOAPAction" κεφαλίδα "header" του SOAP μηνύματος που θα

δημιουργηθεί. Με την “style” ιδιότητα ορίζεται ο τύπος μορφοποίησης της συγκεκριμένης λειτουργίας, αν δεν οριστεί τότε θεωρείται ότι έχει την τιμή που ορίστηκε με την ιδιότητα “style” στο στοιχείο <soap:binding>, αν και αυτή δεν έχει οριστεί τότε θεωρείται ότι είναι “document”.

```
<soap:operation soapAction="http://www.my-uni-project.info/joomla/
components/com_virtuemart_soa_services/getManufacturers"
style="document" />
```

Το επόμενα στοιχεία που πρέπει να οριστούν είναι τα <wsdl:input>, <wsdl:output> και <wsdl:fault>. Το ποια θα οριστούν εξαρτάτε από ποια στοιχεία <input>, <output> και <fault> είχαν οριστεί στην συσχετιζόμενη λειτουργία <operation> μέσα στο στοιχείο <portType>. Η μόνη ιδιότητα που έχουν είναι η “name” και πρέπει να είναι η ίδια με την ιδιότητα “name” των στοιχείων της συσχετιζόμενης λειτουργίας εφόσον έχει οριστεί (η ιδιότητα “name” είναι υποχρεωτική μόνο για τα στοιχεία <fault>).

Στο εσωτερικό των <input> και <output> ορίζεται το στοιχείο <soap:body> με σύνταξη:

```
<soap:body parts="nmtokens"? use="literal|encoded"?
encodingStyle="uri-list"? Namespace="uri"?>
```

Με την ιδιότητα “parts” ορίζονται ποια στοιχεία <part> του μηνύματος θα συμπεριληφθούν, αν δεν ορισθεί θεωρείται ότι θα συμπεριληφθούν όλα. Με την ιδιότητα “use” ορίζεται την μορφοποίηση που θα έχει το SOAP μήνυμα που θα δημιουργηθεί (literal ή encoded). Όταν χρησιμοποιείται η “literal” μορφοποίηση τότε δεν χρειάζεται να οριστεί κάποια άλλη ιδιότητα, αντίθετα όμως όταν χρησιμοποιείται η “encoded” μορφοποίηση πρέπει να οριστεί τόσο η “encodingStyle” όσο και η “Namespace”. Στην περίπτωση χρήσης του SOAP η

τιμή της “encodingStyle” είναι “http://schemas.xmlsoap.org/soap/encoding/”. Η τιμή της “Namespace” συνήθως είναι ίδια με το “targetNamespace”.

Στο εσωτερικό του στοιχείου <fault> ορίζεται το στοιχείο <soap:fault> με σύνταξη:

```
<soap:fault name="nmtoken" use="literal|encoded"
encodingStyle="uri-list"? namespace="uri"?>
```

Η μόνη διαφορά είναι ότι αντί της ιδιότητας “parts” υπάρχει η ιδιότητα “name” που πρέπει να έχει την ίδια τιμή με το αντίστοιχο στοιχείο <fault> της συσχετιζόμενης λειτουργίας.

Επιπλέον προαιρετικά στοιχεία που απλά θα αναφερθεί η σύνταξή τους και μπορούν να οριστούν είναι τα <soap:header> και <soap:headerfault> τα οποία ορίζουν μηνύματα που μεταφέρονται μέσα στο στοιχείο <soapenv:Header> του SOAP μηνύματος. Η σύνταξή τους είναι:

```
<soap:header message="qname" part="nmtoken" use="literal|encoded"
encodingStyle="uri-list"? namespace="uri"?>*

<soap:headerfault message="qname" part="nmtoken" use="literal|
encoded"encodingStyle="uri-list"? namespace="uri"?/>*
```

3.6.6 Παράδειγμα <binding>

Επιστρέφοντας στο παράδειγμά το στοιχείο <binding> ορίζεται ως εξής:

```
<wsdl:binding name="VirtueMart_Products_SOAP"
type="tns:VirtueMart_Products_Service">
     <soap:binding
     transport="http://schemas.xmlsoap.org/soap/http" />
     <wsdl:operation name="getManufacturers">
```

```
          <soap:operation soapAction="http://www.my-uni-
project.info/joomla/components/com_virtuemart_soa_services/getManu
facturers" style="document" />
          <wsdl:input>
               <soap:body use="literal" />
          </wsdl:input>
          <wsdl:output>
               <soap:body use="literal" />
          </wsdl:output>
     </wsdl:operation>
</wsdl:binding>
```

3.7 <service>

Με το στοιχείο <service> ορίζεται το “Πού” βρίσκεται μια υπηρεσία. Παρόλο που σε ένα WSDL αρχείο μπορούν να οριστούν πολλαπλά <service> στοιχεία αποτελεί κοινή πρακτική να ορίζεται μόνο ένα. Το στοιχείο <service> έχει μόνο μια ιδιότητα την “name”, εσωτερικά του <service> ορίζεται στοιχεία <port>. Το πόσα στοιχεία <port> ορίζονται εξαρτάται από το πόσα στοιχεία <binding> υπάρχουν στο WSDL αρχείο και από τον αριθμό των διευθύνσεων που προσφέρουν την υπηρεσία. Το στοιχείο <port> έχει δύο ιδιότητες την “name” που αποτελεί και το όνομα του “end point” της υπηρεσίας, και την “binding” που πρέπει να έχει την ίδια τιμή με την ιδιότητα “name” από κάποιο από τα στοιχεία <binding> που έχουν ορισθεί στο αρχείο. Κάθε στοιχείο <port> πρέπει να εσωκλείει μόνο ένα στοιχείο <soap:address> το οποίο έχει μόνο μια ιδιότητα την “location” που ορίζει σε ποια διεύθυνση είναι προσβάσιμη η υπηρεσία. Στο παράδειγμά μας το στοιχείο <service> ορίζεται ως εξής:

```
<wsdl:service name="VirtueMart_Products_Srv">
     <wsdl:port name="VirtueMart_Products_SOAP"
      binding="tns:VirtueMart_Products_SOAP">
```

```
          <soap:address location="http://www.my-uni-project.info/
joomla/components/com_virtuemart_soa_services/VirtueMart_Products.
php" />
     </wsdl:port>
</wsdl:service>
```

ΕΠΙΛΟΓΟΣ

Σε αυτό το κεφάλαιο αναλύθηκε η δομή του WSDL αρχείου και αναφέρθηκαν οι διάφορες μορφοποιήσεις των παραγόμενων SOAP μηνυμάτων. Η ορθή κατανόηση της δομής του είναι βασικό στοιχείο για την κατανόηση του τρόπου ανταλλαγής των μηνυμάτων μεταξύ του καταναλωτή και μιας υπηρεσίας ιστού. Η δημιουργία του WSDL αρχείου είναι το δυσκολότερο βήμα στην δημιουργία μιας υπηρεσίας ιστού για αυτό έχουν αναπτυχθεί εργαλεία που βοηθούν στην σύνθεσή του, ένα από αυτά είναι το XMLSpy από την Altova, ενώ το Visual Studio 2008 δημιουργεί αυτόματα το WSDL αρχείο αναλύοντας την δομή της υπηρεσίας ιστού. Στο επόμενο κεφάλαιο θα παρουσιαστεί η λειτουργικότητα, οι δομές δεδομένων και ο βασικός σχεδιασμός της εφαρμογής διασύνδεσης του ηλεκτρονικού καταστήματος και του εμπορικού πληροφοριακού συστήματος.

ΚΕΦΑΛΑΙΟ 4

ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ, ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΒΑΣΙΚΟΣ ΣΧΕΔΙΑΣΜΟΣ

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο θα παρουσιαστεί η λειτουργικότητα της προτεινόμενης λύσης διασύνδεσης μεταξύ ενός ERP και του ηλεκτρονικού καταστήματος της επιχείρησης, οι δομές δεδομένων που θα πρέπει να υλοποιηθούν, καθώς και ένας βασικός σχεδιασμός της λύσης.

4.1 Λειτουργικότητα

Το ενδιάμεσο λογισμικό διασύνδεσης θα παρέχει την παρακάτω λειτουργικότητα:

- Δυνατότητα αντιστοίχισης των πινάκων: Χώρες, Νομίσματα, κατηγορίες ΦΠΑ, Καταστάσεις Παραγγελίας, Τρόπων Αποστολής και Τρόπων Πληρωμής.
- Εισαγωγή στο ERP των στοιχείων νέων πελατών και ενημέρωση του ERP με πιθανές μεταγενέστερες μεταβολές των στοιχείων των πελατών
- Εισαγωγή στο ηλεκτρονικό κατάστημα των Κατηγοριών Ειδών και των Κατασκευαστών Ειδών
- Εισαγωγή και ενημέρωση στο ηλεκτρονικό κατάστημα των ειδών και του αποθέματός τους
- Εισαγωγή στο ERP των παραγγελιών
- Ενημέρωση στο ηλεκτρονικό κατάστημα της κατάστασης των παραγγελιών

4.2 Δομές δεδομένων

Για να γίνει δυνατή η σχεδίαση του ενδιάμεσο λογισμικό χωρίς να επηρεαστεί από συγκεκριμένες ανάγκες και ιδιαιτερότητες συγκεκριμένων προϊόντων θα πρέπει να οριστούν οι δομές δεδομένων του ενδιάμεσου λογισμικού. Στην φάση της υλοποίησης οι δομές αυτές θα χρησιμοποιηθούν για να μεταφερθούν δεδομένα μεταξύ των ετερογενών σχημάτων των βάσεων του ERP και του ηλεκτρονικού καταστήματος.

Πίνακας 1: Δομές δεδομένων - Χώρα (Country)

Χώρα (Country)	
Πεδίο	*Τύπος*
Country	integer
Name	string

Πίνακας 2: Δομές δεδομένων - Νόμισμα (Currency)

Νόμισμα (Currency)	
Πεδίο	*Τύπος*
Currency	integer
Name	string

Πίνακας 3: Δομές δεδομένων - ΦΠΑ (Vat)

ΦΠΑ (Vat)	
Πεδίο	*Τύπος*
Vat	integer
Name	string

Πίνακας 4: Δομές δεδομένων - Κατάσταση Παραγγελίας (Status)

Κατάσταση Παραγγελίας (Status)	
Πεδίο	*Τύπος*
Status	integer
Name	string

Πίνακας 5: Δομές δεδομένων - Τρόπος Αποστολής (Shippment)

Τρόπος Αποστολής (Shippment)	
Πεδίο	*Τύπος*
Shippment	integer
Name	string

Πίνακας 6: Δομές δεδομένων - Τρόπος Πληρωμής (Payment)

Τρόπος Πληρωμής (Payment)	
Πεδίο	*Τύπος*
Payment	integer
Name	string

Πίνακας 7: Δομές δεδομένων - Πελάτες (Customer)

Πελάτες (Customer)	
Πεδίο	*Τύπος*
Customer_Id	string
VatNo	string
CompanyName	string
BillingName	string
Email	string
Address_1	string
Address_2	string
City	string
Zip	string
Country	string
Phone_1	string
Phone_2	string
Fax	string
Permissions	string

Πίνακας 8: Δομές δεδομένων - Κατηγορίες (Category)

Κατηγορίες (Category)	
Πεδίο	*Τύπος*
Category_Id	integer
Parent_Category	integer
Name	string
Description	string
IsActive	string
category_thumb_image	string
category_full_image	string
products_per_row	string

Πίνακας 9: Δομές δεδομένων - Κατασκευαστές (Manufacturer)

Κατασκευαστές (Manufacturer)	
Πεδίο	*Τύπος*
Manufacturer_Id	integer
Name	string

Πίνακας 10: Δομές δεδομένων - Είδη (Item)

Είδη (Item)	
Πεδίο	*Τύπος*
Item_Id	integer
Code	string
Name	string
Category	integer
Manufacturer	integer
IsActive	string
Currency	integer
Vat	integer
Unit	string

DescriptionShort	string
Description	string
Price	float
DiscPrc	float
DiscVal	float
Stock	float
Availability	string
Item_thumb_Image	string
Item_Full_Image	string

Πίνακας 11: Δομές δεδομένων - Κατάσταση Παραγγελίας (OrderStatus)

Κατάσταση Παραγγελίας (OrderStatus)	
Πεδίο	*Τύπος*
OrderNumber	string
Status	string

Πίνακας 12: Δομές δεδομένων - Παραγγελίες - Βασικά στοιχεία (OrderHeader)

Παραγγελίες – Βασικά στοιχεία (OrderHeader)	
Πεδίο	*Τύπος*
CustomerId	string
OrderNumber	string
OrderDate	string
Shippment	integer
Payment	integer
PaymentMethodDiscount	float
CustomerNotes	string
Currency	integer
Status	integer
ItemsValue	float
ShippingValue	float

ShippingVatValue	float
TotalDiscount	float
TotalVatValue	float
TotalValue	float

Πίνακας 13: Δομές δεδομένων - Παραγγελίες - Γραμμές ειδών (OrderItem)

Παραγγελίες – Γραμμές ειδών (OrderItem)	
Πεδίο	*Τύπος*
OrderNumber	string
ProductCode	string
Quantity	float
Price	float
Total	float

Πίνακας 14: Δομές δεδομένων - Παραγγελίες - Στοιχεία τιμολόησης (BillingInfo)

Παραγγελίες – Στοιχεία τιμολόγησης (BillingInfo)	
Πεδίο	*Τύπος*
OrderNumber	string
Company	string
VatNo	string
Email	string
Name	string
Address_1	string
Address_2	string
City	string
Zip	string
Country	string
Phone	string
Mobile	string
Fax	string

Πίνακας 15: Δομές δεδομένων - Παραγγελίες - Στοιχεία αποστολής (ShippingInfo)

Παραγγελίες – Στοιχεία αποστολής (ShippingInfo)	
Πεδίο	Τύπος
OrderNumber	string
Name	string
Address_1	string
Address_2	string
City	string
Zip	string
Country	string
Phone	string
Mobile	string
Fax	string

4.3 Βασικός Σχεδιασμός

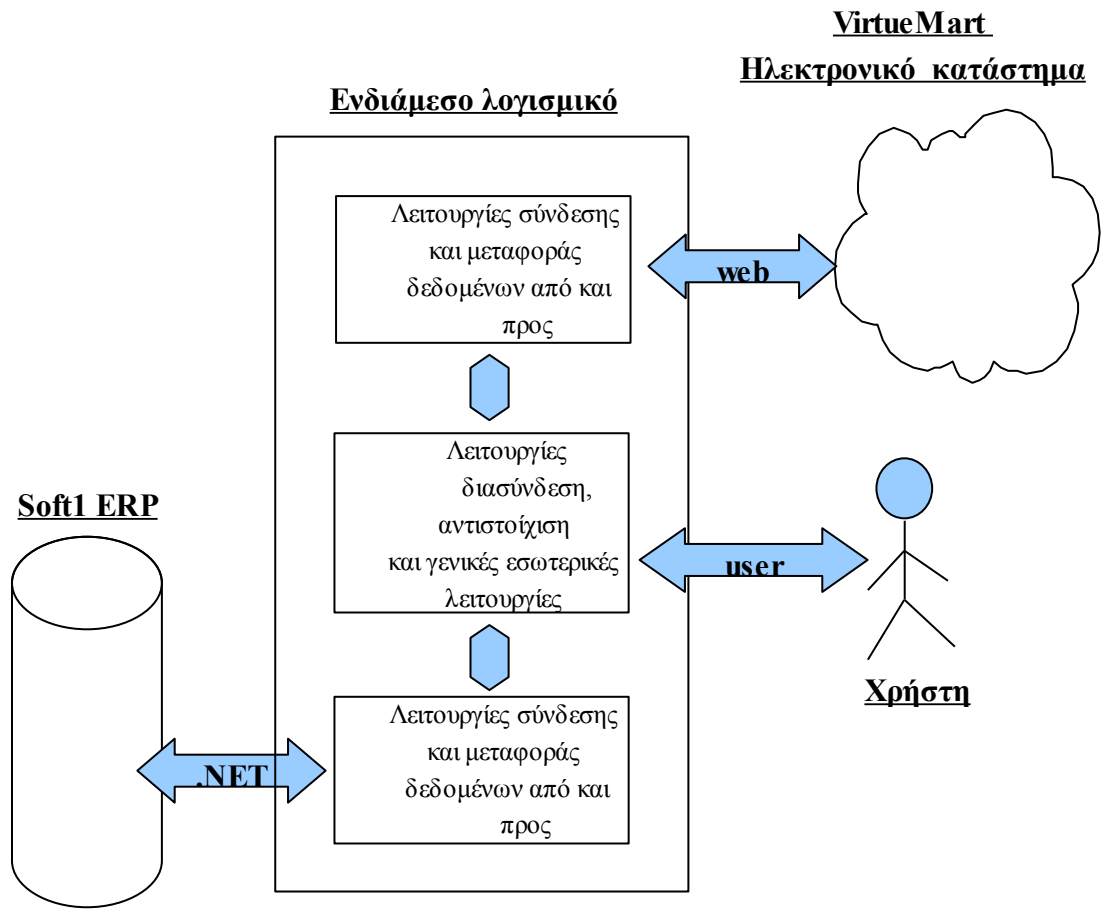
Το ενδιάμεσο λογισμικό που θα σχεδιαστεί θα ομαδοποιεί ως προς τον χρήστη (διεπαφή χρήστη) τις λειτουργίες σε τρεις ενότητες:

- Παράμετροι διασύνδεσης με το ERP και το ηλεκτρονικό κατάστημα
- Αντιστοιχίσεις βασικών αρχείων ERP και ηλεκτρονικού καταστήματος
- Λειτουργίες ενημέρωσης ERP και ηλεκτρονικού καταστήματος

Σχεδιαστικά θα χωρίζεται πάλι σε τρεις ενότητες:

- Λειτουργίες σύνδεσης και μεταφοράς δεδομένων από και προς το ηλεκτρονικό κατάστημα
- Λειτουργίες σύνδεσης και μεταφοράς δεδομένων από και προς το ERP

- Λειτουργίες διασύνδεση, αντιστοίχιση και γενικές εσωτερικές λειτουργίες του



Σχήμα 4: Βασικός σχεδιασμός του ενδιάμεσου λογισμικού

ενδιάμεσου λογισμικού.

4.4 Προϊόντα που χρησιμοποιήθηκαν

Η υλοποίησης του ενδιάμεσου λογισμικού έγινε σε C# χρησιμοποιώντας το Microsoft Visual Studio 2008. Το ηλεκτρονικό κατάστημα είναι το VirtueMart. Το VirtueMart είναι ένα ηλεκτρονικό κατάστημα Ανοιχτού Λογισμικού (VirtueMart 2010) που χρησιμοποιείται μαζί με το CMS (Content Managment System) Joomla (Joomla 2010), και είναι γραμμένο σε PHP και χρησιμοποιεί ως DBMS τη MySql. Το ERP που χρησιμοποιήθηκε είναι το Soft1 ERP από την εταιρεία SoftOne

(Softone 2010). Για την διασύνδεση με το ηλεκτρονικό κατάστημα δημιουργήθηκαν υπηρεσίες ιστού (web services) ενώ για την διασύνδεση με το ERP χρησιμοποιήθηκαν οι βιβλιοθήκες διασύνδεσης .NET (Softone Developers Network 2010) που παρέχει η εταιρεία.

ΕΠΙΛΟΓΟΣ

Στο παρών κεφάλαιο ορίστηκαν οι δομές, οι λειτουργίες και ο βασικός σχεδιασμός του ενδιάμεσου λογισμικού. Στο επόμενο κεφάλαιο θα αναλυθεί ο σχεδιασμός των WSDL αρχείων των υπηρεσιών ιστού που δημιουργήθηκαν.

ΚΕΦΑΛΑΙΟ 5

ΣΧΕΔΙΑΣΜΟΣ ΤΩΝ WSDL ΑΡΧΕΙΩΝ

ΕΙΣΑΓΩΓΗ

Το πρώτο βήμα για την δημιουργία των υπηρεσιών ιστού που θα χρησιμοποιήσει το ενδιάμεσο λογισμικό για να επικοινωνεί με το ηλεκτρονικό κατάστημα είναι να δημιουργηθούν τα WSDL αρχεία. Στο παρών κεφάλαιο αρχικά ορίζονται πόσες υπηρεσίες ιστού και με ποιες μεθόδους θα υλοποιηθούν, και στη συνέχεια σχεδιάζονται τα WSDL αρχεία.

5.1 Ομαδοποίηση λειτουργιών σε υπηρεσίες ιστού και οι μέθοδοι τους

Η λειτουργίες που απαιτούνται ομαδοποιούνται σε τέσσερις ομάδες, και κατ' επέκταση σε τέσσερις υπηρεσίες ιστού.

5.1.1 VirtueMart_Store_Srv

Η υπηρεσία αυτή ανακτά από το ηλεκτρονικό κατάστημα τους παρακάτω βασικούς πίνακες:

Χώρες

Νομίσματα

ΦΠΑ

Καταστάσεις Παραγγελίας

Τρόποι Αποστολής

Τρόποι Πληρωμής

Έχει μόνο μια μέθοδο την “getStoreBasicParameters”. Δέχεται ως παραμέτρους το όνομα του χρήστη και τον κωδικό πρόσβασης (ο χρήστης πρέπει να είναι ο Διαχειριστής του ηλεκτρονικού καταστήματος), και επιστρέφει ένα αντικείμενο που θα εμπεριέχει πίνακες με τις παραπάνω πληροφορίες

5.1.2 VirtueMart_Customers_Srv

Η υπηρεσία αυτή ανακτά από το ηλεκτρονικό κατάστημα την λίστα των Πελατών. Έχει μόνο μια μέθοδο την “getCustomers”. Δέχεται ως παραμέτρους το όνομα του χρήστη και τον κωδικό πρόσβασης (ο χρήστης πρέπει να είναι ο Διαχειριστής του ηλεκτρονικού καταστήματος), και επιστρέφει ένα αντικείμενο με την λίστα των πελατών.

5.1.3 VirtueMart_Products_Srv

Η υπηρεσία αυτή ανακτά από το ηλεκτρονικό κατάστημα τις Κατηγορίες ειδών, τους Κατασκευαστές και τα Είδη. Ακόμη προσθέτει και μεταβάλει τα στοιχεία των κατηγοριών, των κατασκευαστών και των ειδών του ηλεκτρονικού καταστήματος.

Έχει τις εξής μεθόδους, “getCategories”, “getManufacturers”, “getItems”, “modifyCategories”, “modifyManufacturers”, “modifyItems”.

Οι μέθοδοι “getCategories”, “getManufacturers” και “getItems” δέχονται ως παραμέτρους το όνομα του χρήστη και τον κωδικό πρόσβασης (ο χρήστης πρέπει να είναι ο Διαχειριστής του ηλεκτρονικού καταστήματος), και επιστρέφει ένα αντικείμενο με την λίστα των κατηγοριών, των κατασκευαστών και των ειδών αντίστοιχα.

Οι μέθοδοι “modifyCategories”, “modifyManufacturers” και “modifyItems” δέχονται ως παραμέτρους το όνομα του χρήστη, τον κωδικό πρόσβασης (ο χρήστης πρέπει να είναι ο Διαχειριστής του ηλεκτρονικού καταστήματος), τον τύπο μεταβολής “ADD” ή “UPDATE”, και μια λίστα με τα στοιχεία που θα πρέπει να προστεθούν/ενημερωθούν. Επιστρέφει ένα αντικείμενο με την λίστα των αποτελεσμάτων των μεταβολών.

5.1.4 VirtueMart_Orders_Srv

Η υπηρεσία αυτή ανακτά από το ηλεκτρονικό κατάστημα την λίστα των Παραγγελιών, προσφέρει την δυνατότητα να γίνει ανάκτηση όλων των παραγγελιών ή εναλλακτικά των παραγγελιών που είναι μεταγενέστερες από μια συγκεκριμένη παραγγελία. Ακόμη προσφέρει την δυνατότητα ενημέρωσης της κατάστασης παραγγελίας. Έχει δύο μεθόδους την “getOrders” και την “setOrderStatus”.

Η “getOrders” δέχεται ως παραμέτρους το όνομα του χρήστη, τον κωδικό πρόσβασης (ο χρήστης πρέπει να είναι ο Διαχειριστής του ηλεκτρονικού καταστήματος), και προαιρετικά έναν αριθμό παραγγελίας για την ανάκτηση νεότερων παραγγελιών. Επιστρέφει ένα αντικείμενο με την λίστα των παραγγελιών.

Η “setOrderStatus” δέχονται ως παραμέτρους το όνομα του χρήστη, τον κωδικό πρόσβασης (ο χρήστης πρέπει να είναι ο Διαχειριστής του ηλεκτρονικού καταστήματος), τον αριθμό παραγγελίας της οποίας η κατάσταση θα ενημερωθεί και την τιμή της νέας κατάστασής της. Επιστρέφει το αποτέλεσμα της ενέργειας.

5.2 Δημιουργία των WSDL αρχείων

Σε προηγούμενο κεφάλαιο αναλύθηκε ο τρόπος δημιουργίας ενός WSDL αρχείου, οπότε εδώ παρουσιάζεται η δομή των αρχείων χωρίς να παραθέτεται ο κώδικας, ακόμη για δομές που έχουν αναφερθεί στο κεφάλαιο “Λειτουργικότητα, δομές δεδομένων και βασικός σχεδιασμός” δεν γίνει αναφορά στα πεδία τους.

5.2.1 VirtueMart_Store_Srv

WSDL location: “http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/VirtueMart_Store.wsdl”

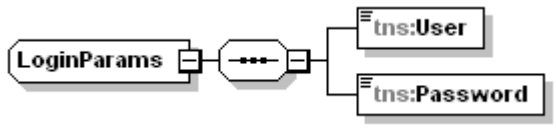
targetNamespace: “http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/VirtueMart_Store”

Τύποι <types>

LoginParams: Παράμετροι σύνδεσης χρήστη στο ηλεκτρονικό κατάστημα

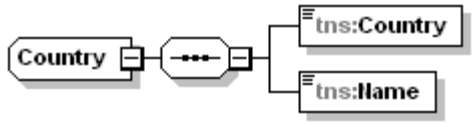
Πίνακας 16: WSDL VirtueMart_Store_Srv <types> - LoginParams

Πεδίο	Τύπος
User	string
Password	string



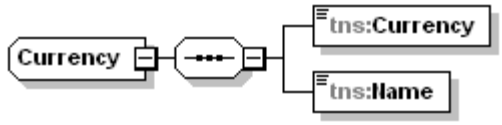
Σχήμα 5: LoginParams

Country: βλέπε δομή δεδομένων “**Χώρα**”



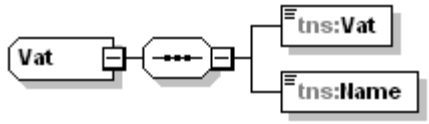
Σχήμα 6: Country

Currency: βλέπε δομή δεδομένων "**Νόμισμα**"



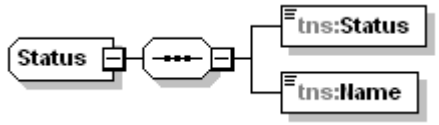
Σχήμα 7: Currency

Vat: βλέπε δομή δεδομένων "**ΦΠΑ**"



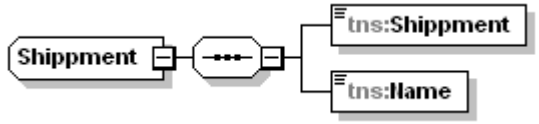
Σχήμα 8: Vat

Status: βλέπε δομή δεδομένων "**Κατάσταση Παραγγελίας**"



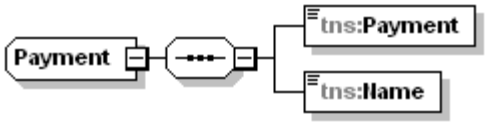
Σχήμα 9: Status

Shippment: βλέπε δομή δεδομένων "**Τρόπος Αποστολής**"



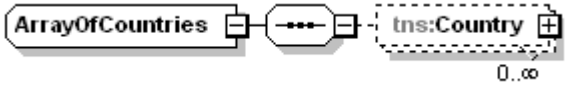
Σχήμα 10: Shippment

Payment: βλέπε δομή δεδομένων "**Τρόπος Πληρωμή**"



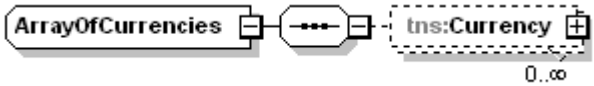
Σχήμα 11: Payment

ArrayOfCountries: Πίνακας χωρών



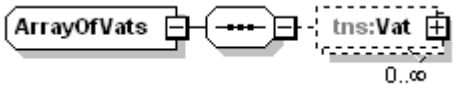
Σχήμα 12: ArrayOfCountries

ArrayOfCurrencies: Πίνακας νομισμάτων



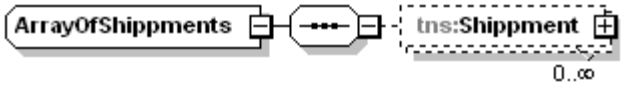
Σχήμα 13: ArrayOfCurrencies

ArrayOfVats: Πίνακας ΦΠΑ



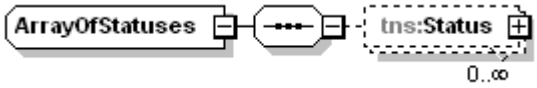
Σχήμα 14: ArrayOfVats

ArrayOfStatuses: Πίνακας καταστάσεων παραγγελίας



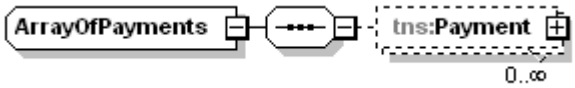
Σχήμα 15: ArrayOfStatuses

ArrayOfShippments: Πίνακας τρόπων αποστολής



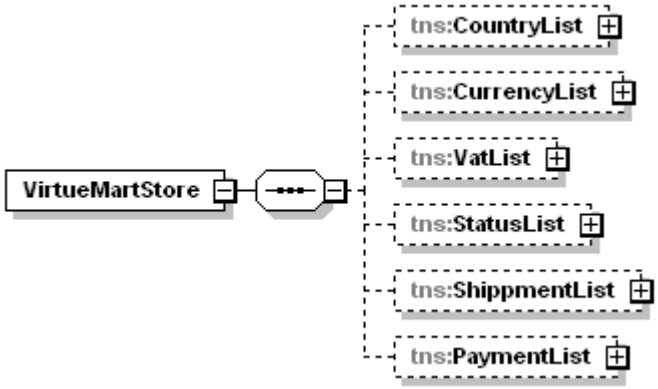
Σχήμα 16: ArrayOfShippments

ArrayOfPayments: Πίνακας τρόπων πληρωμής



Σχήμα 17: ArrayOfPayments

VirtueMartStore: Δομή που περιέχει τον πίνακα χωρών, νομισμάτων, ΦΠΑ, καταστάσεων παραγγελίας, τρόπων αποστολής και τρόπων πληρωμών.



Σχήμα 18: VirtueMartStore

VirtueMartLogin: Δομή που εμπεριέχει τις παραμέτρους σύνδεσης χρήστη.

Σχήμα 19: VirtueMartLogin

Μηνύματα <message>

Πίνακας 17: WSDL VirtueMart_Store_Srv <message> - StoreRequest

StoreRequest	
Parts Name	Parts element
parameters	tns:VirtueMartLogin

Πίνακας 18: WSDL VirtueMart_Store_Srv <message> - StoreResponse

StoreResponse	
Parts Name	Parts element
results	tns:VirtueMartStore

Port types <portType>

Πίνακας 19: WSDL VirtueMart_Store_Srv <portType> - VirtueMart_Store_Service

VirtueMart_Store_Service	
operation	getStoreBasicParameters
input	tns:StoreRequest
output	tns:StoreResponse

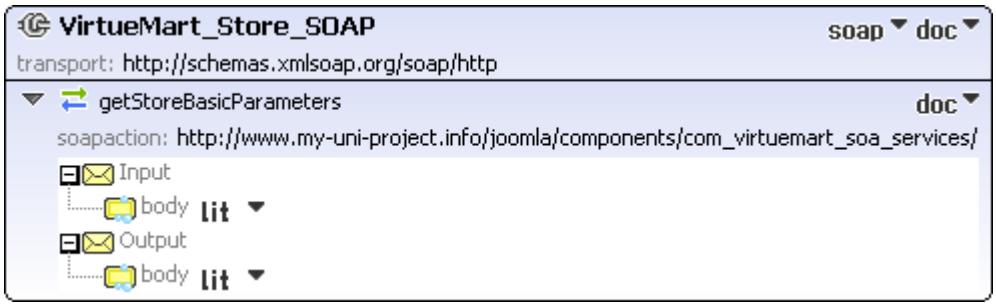
Σχήμα 20: VirtueMart_Store_Service

Bindings <binding>

Πίνακας 20: WSDL VirtueMart_Store_Srv <binding> - VirtueMart_Store_SOAP

VirtueMart_Store_SOAP	
type	tns:VirtueMart_Store_Service
transport	"http://schemas.xmlsoap.org/soap/http"

operation	getStoreBasicParameters
soapAction	http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/
style	document
input	literal
output	literal

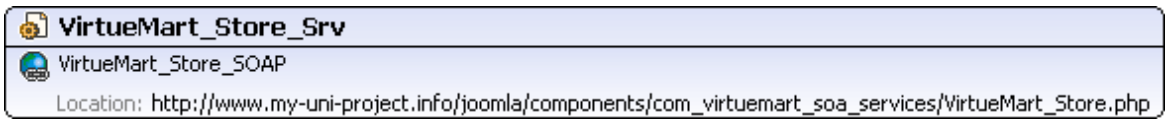


Σχήμα 21: VirtueMart_Store_SOAP

Υπηρεσίες <service>

Πίνακας 21: WSDL VirtueMart_Store_Srv <service> - VirtueMart_Store_Srv

VirtueMart_Store_Srv	
Port name	VirtueMart_Store_SOAP
Port binding	tns:VirtueMart_Store_SOAP
location	http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/VirtueMart_Store.php



Σχήμα 22: VirtueMart_Store_Srv

5.2.2 VirtueMart_Customers_Srv

WSDL location: “http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/VirtueMart_Customers.wsdl”

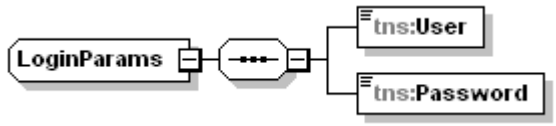
targetNamespace: “http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/VirtueMart_Customers”

Τύποι <types>

LoginParams: Παράμετροι σύνδεσης χρήστη στο ηλεκτρονικό κατάστημα

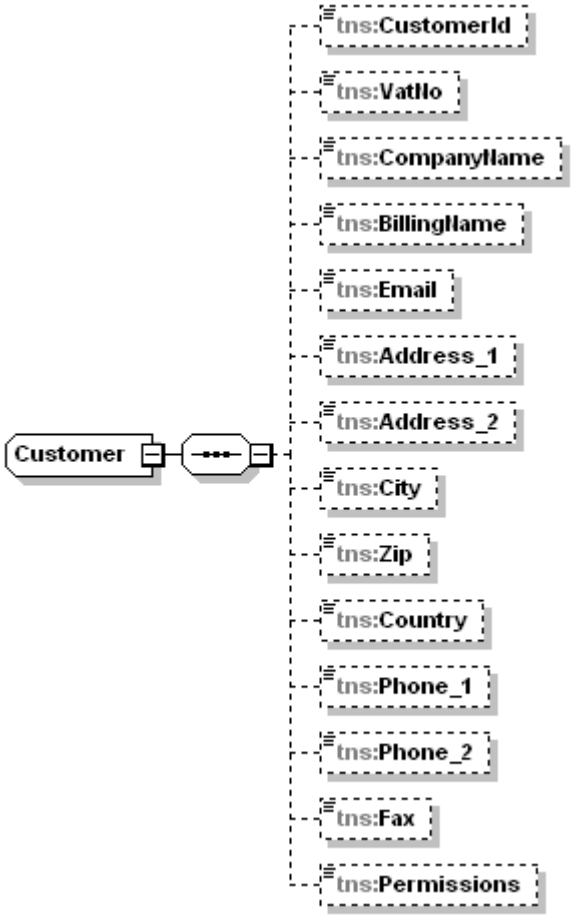
Πίνακας 22: WSDL VirtueMart_Customers_Srv <types> - LoginParams

Πεδίο	Τύπος
User	string
Password	string



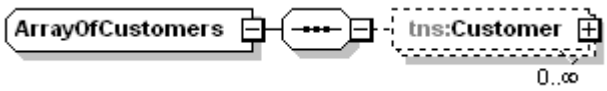
Σχήμα 23: LoginParams

Customer: βλέπε δομή δεδομένων “**Πελάτες**”



Σχήμα 24: Customer

ArrayOfCustomers: Πίνακας πελατών



Σχήμα 25: ArrayOfCustomers

Customers: Δομή που περιέχει τον πίνακα των πελατών.

Σχήμα 26: Customers

VirtueMartLogin: Δομή που εμπεριέχει τις παραμέτρους σύνδεσης χρήστη.

Σχήμα 27: VirtueMartLogin

Μηνύματα <message>

Πίνακας 23: WSDL VirtueMart_Customers_Srv <message> - CustomerRequest

CustomersRequest	
Parts Name	Parts element
parameters	tns:VirtueMartLogin

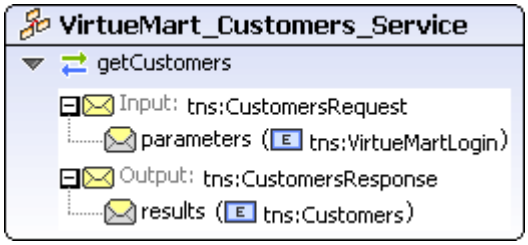
Πίνακας 24: WSDL VirtueMart_Customers_Srv <message> - CustomerResponse

CustomersResponse	
Parts Name	Parts element
results	tns:Customers

Port types <portType>

Πίνακας 25: WSDL VirtueMart_Customers_Srv <portType> - VirtueMart_Customers_Service

VirtueMart_Customers_Service	
operation	getCustomers
input	tns:CustomersRequest
output	tns:CustomersResponse

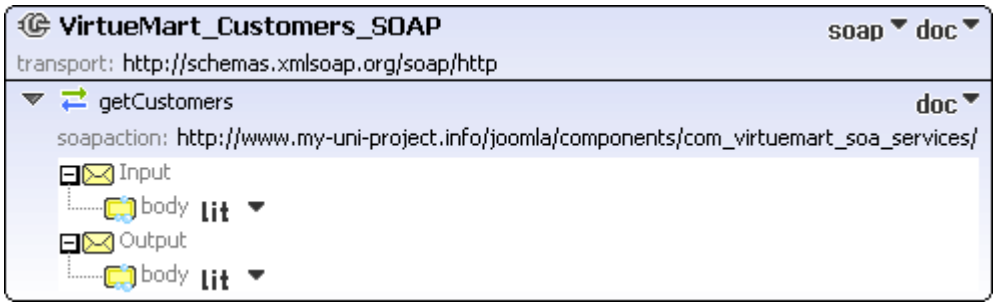


Σχήμα 28: VirtueMart_Customers_Service

Bindings <binding>

Πίνακας 26: WSDL VirtueMart_Customers_Srv <binding> - VirtueMart_Customers_SOAP

VirtueMart_Customers_SOAP	
type	tns:VirtueMart_Customers_Service
transport	"http://schemas.xmlsoap.org/soap/http"
operation	getCustomers
soapAction	http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/
style	document
input	literal
output	literal



Σχήμα 29: VirtueMart_Customers_SOAP

Υπηρεσίες <service>

Πίνακας 27: WSDL VirtueMart_Customers_Srv <binding> - VirtueMart_Customers_Srv

VirtueMart_Customers_Srv	
Port name	VirtueMart_Customers_Srv_SOAP
Port binding	tns:VirtueMart_Customers_SOAP
location	http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/VirtueMart_Customers.php

Σχήμα 30: VirtueMart_Customers_Srv

5.2.3 VirtueMart_Products_Srv

WSDL location: "http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/VirtueMart_Products.wsdl"

targetNamespace: "http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/VirtueMart_Products"

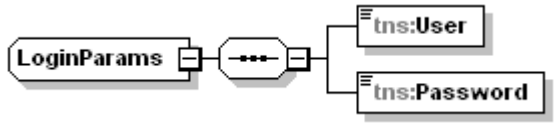
Τύποι <types>

LoginParams: Παράμετροι σύνδεσης χρήστη στο ηλεκτρονικό κατάστημα

Πίνακας 28: WSDL VirtueMart_Products_Srv <types> - LoginParams

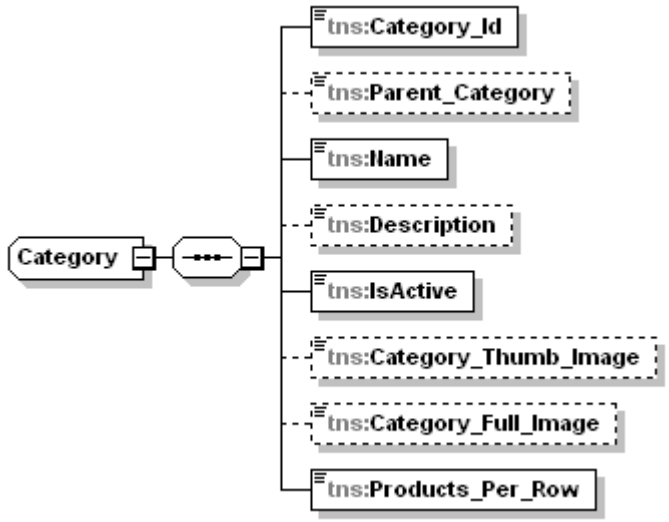
Πεδίο	Τύπος
User	string

Password	string



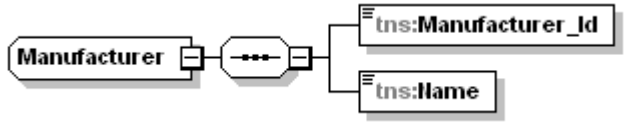
Σχήμα 31: LoginParams

Category: βλέπε δομή δεδομένων "**Κατηγορίες**"



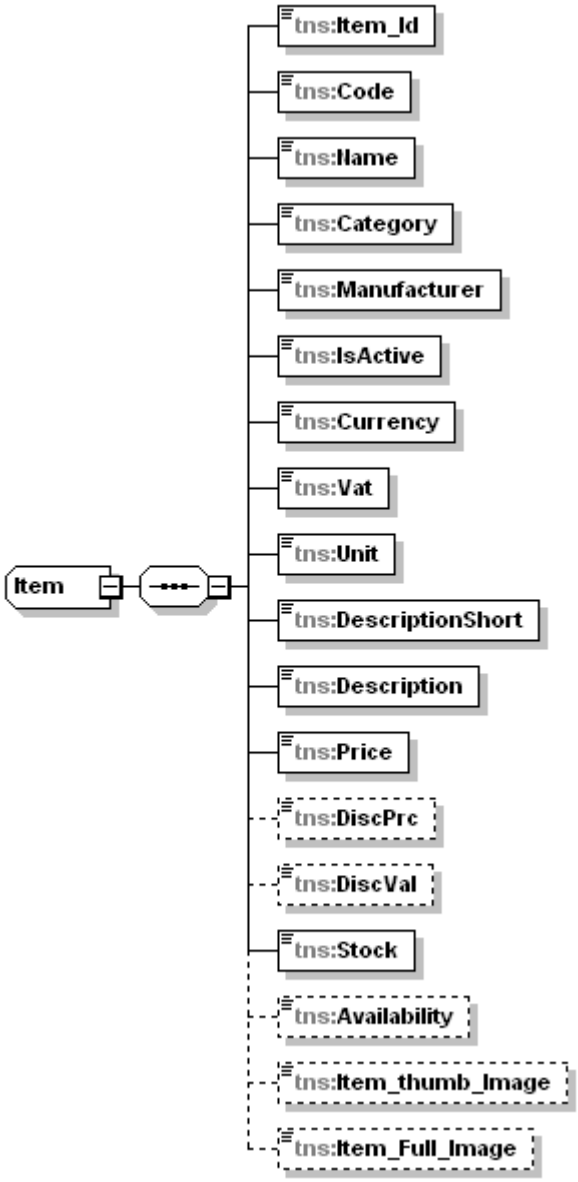
Σχήμα 32: Category

Manufacturer: βλέπε δομή δεδομένων "**Κατασκευαστές**"



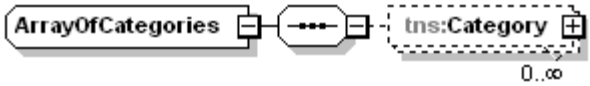
Σχήμα 33: Manufacturer

Item: βλέπε δομή δεδομένων "**Είδη**"



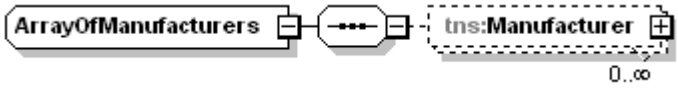
Σχήμα 34: Item

ArrayOfCategories: Πίνακας κατηγοριών



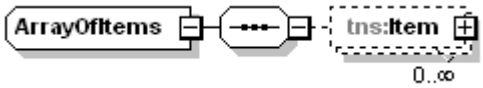
Σχήμα 35: ArrayOfCategories

ArrayOfManufacturers: Πίνακας κατασκευαστών



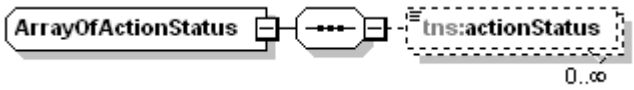
Σχήμα 36: ArrayOfManufacturers

ArrayOfItems: Πίνακας ειδών



Σχήμα 37: ArrayOfItems

ArrayOfActionStatus: Πίνακας καταστάσεων ενεργειών (τύπου string)



Σχήμα 38: ArrayOfActionStatus

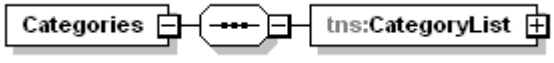
Categories: Δομή που περιέχει τον πίνακα των κατηγοριών.

Σχήμα 39: Categories

Manufacturers: Δομή που περιέχει τον πίνακα των κατασκευαστών.

Σχήμα 40: Manufacturers

Items: Δομή που περιέχει τον πίνακα των ειδών.



Σχήμα 41: Items

CategoriesLogin: Δομή που εμπεριέχει τις παραμέτρους σύνδεσης χρήστη όταν ανακτούμε τις κατηγορίες.

Σχήμα 42: CategoriesLogin

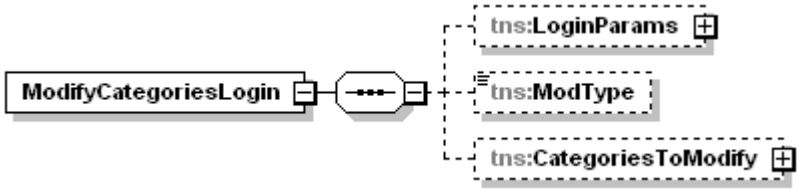
ManufacturersLogin: Δομή που εμπεριέχει τις παραμέτρους σύνδεσης χρήστη όταν ανακτούμε τους κατασκευαστές.

Σχήμα 43: ManufacturersLogin

ItemsLogin: Δομή που εμπεριέχει τις παραμέτρους σύνδεσης χρήστη όταν ανακτούμε τα είδη.

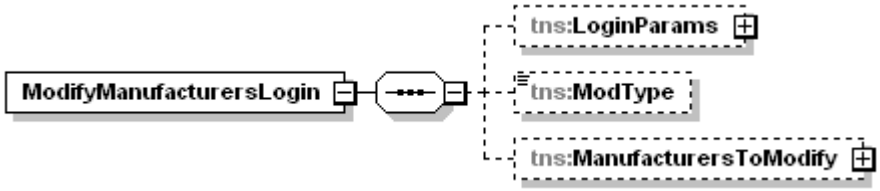
Σχήμα 44: ItemsLogin

ModifyCategoriesLogin: Δομή που εμπεριέχει τις παραμέτρους σύνδεσης χρήστη, τον τύπο ενημέρωσης και την λίστα των κατηγοριών προς ενημέρωσης.



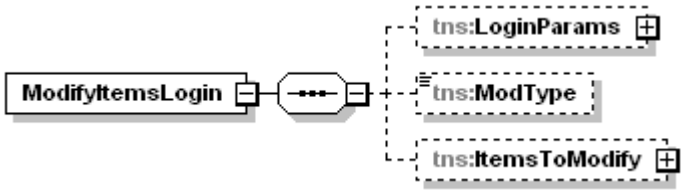
Σχήμα 45: ModifyCategoriesLogin

ModifyManufacturersLogin: Δομή που εμπεριέχει τις παραμέτρους σύνδεσης χρήστη, τον τύπο ενημέρωσης και την λίστα των κατασκευαστών προς ενημέρωσης.



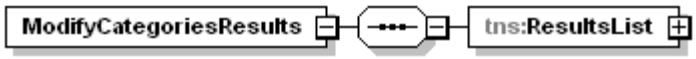
Σχήμα 46: ModifyManufacturersLogin

ModifyItemsLogin: Δομή που εμπεριέχει τις παραμέτρους σύνδεσης χρήστη, τον τύπο ενημέρωσης και το είδος προς ενημέρωσης.



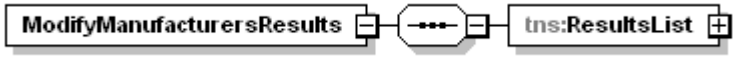
Σχήμα 47: ModifyItemsLogin

ModifyCategoriesResults: Δομή που εμπεριέχει τα αποτελέσματα της ενημέρωσης των κατηγοριών



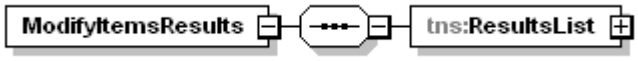
Σχήμα 48: ModifyCategoriesResults

ModifyManufacturersResults: Δομή που εμπεριέχει τα αποτελέσματα της ενημέρωσης των κατασκευαστών



Σχήμα 49: ModifyManufacturersResults

ModifyItemsResults: Δομή που εμπεριέχει τα αποτελέσματα της ενημέρωσης του είδους



Σχήμα 50: ModifyItemsResults

Μηνύματα <message>

Πίνακας 29: WSDL VirtueMart_Products_Srv <message> - CategoriesRequest

CategoriesRequest	
Parts Name	Parts element
parameters	tns:CategoriesLogin

Πίνακας 30: WSDL VirtueMart_Products_Srv <message> - CategoriesResponse

CategoriesResponse	
Parts Name	Parts element
results	tns:Categories

Πίνακας 31: WSDL VirtueMart_Products_Srv <message> - ManufacturersRequest

ManufacturersRequest	
Parts Name	Parts element
parameters	tns:ManufacturersLogin

Πίνακας 32: WSDL VirtueMart_Products_Srv <message> - ManufacturersResponse

ManufacturersResponse	
Parts Name	Parts element
results	tns:Manufacturers

Πίνακας 33: WSDL VirtueMart_Products_Srv <message> - ItemsRequest

ItemsRequest	
Parts Name	Parts element
parameters	tns:ItemsLogin

Πίνακας 34: WSDL VirtueMart_Products_Srv <message> - ItemsResponse

ItemsResponse	
Parts Name	Parts element
results	tns:Items

Πίνακας 35: WSDL VirtueMart_Products_Srv <message> - ModifyCategoriesRequest

ModifyCategoriesRequest	
Parts Name	Parts element
parameters	tns:ModifyCategoriesLogin

Πίνακας 36: WSDL VirtueMart_Products_Srv <message> - ModifyCategoriesResponse

ModifyCategoriesResponse	
Parts Name	Parts element
results	tns:ModifyCategoriesResults

Πίνακας 37: WSDL VirtueMart_Products_Srv <message> - ModifyManufacturersRequest

ModifyManufacturersRequest	
Parts Name	Parts element
parameters	tns:ModifyManufacturersLogin

Πίνακας 38: WSDL VirtueMart_Products_Srv <message> - ModifyManufacturersResponse

ModifyManufacturersResponse	
Parts Name	Parts element
results	tns:ModifyManufacturersResults

Πίνακας 39: WSDL VirtueMart_Products_Srv <message> - ModifyItemsRequest

ModifyItemsRequest	
Parts Name	Parts element
parameters	tns:ModifyItemsLogin

Πίνακας 40: WSDL VirtueMart_Products_Srv <message> - ModifyItemsResponse

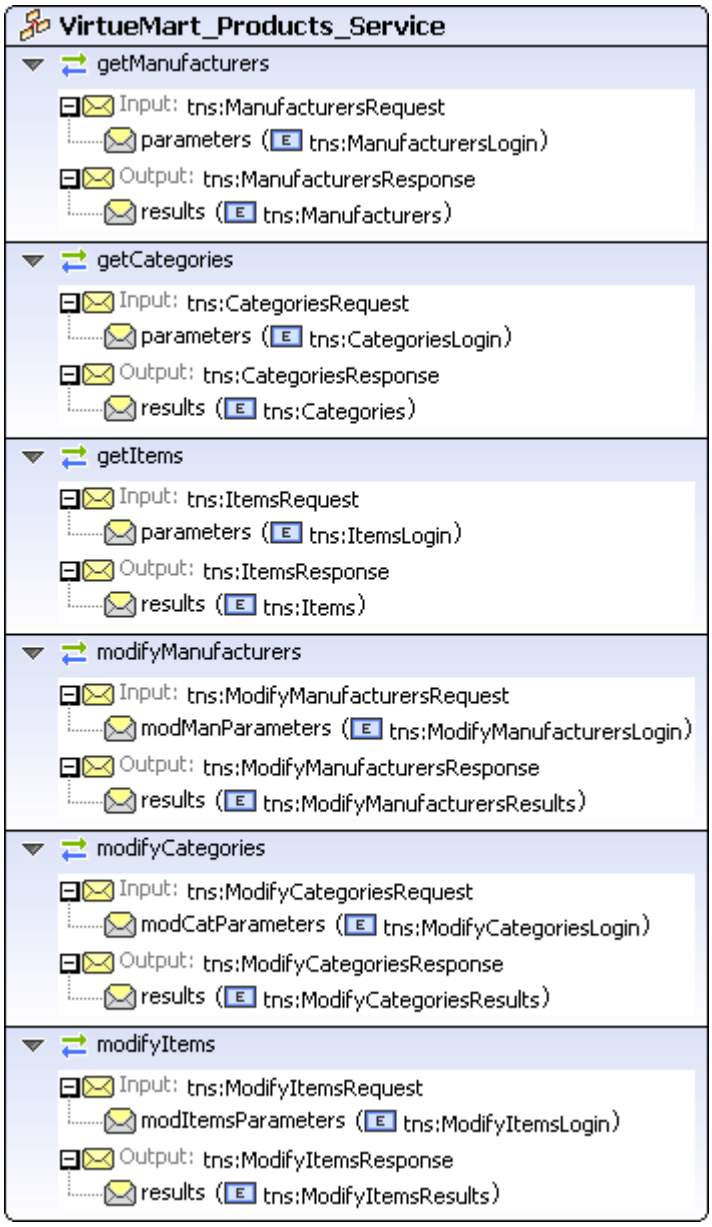
ModifyItemsResponse	
Parts Name	Parts element
results	tns:ModifyItemsResults

Port types <portType>

Πίνακας 41: WSDL VirtueMart_Products_Srv <portTypes> - VirtueMart_Products_Service

VirtueMart_Products_Service	
operation	getCategories
input	tns:CategoriesRequest
output	tns:CategoriesResponse
operation	getManufacturers
input	tns:ManufacturersRequest
output	tns:ManufacturersResponse
operation	getItems
input	tns:ItemsRequest
output	tns:ItemsResponse
operation	modifyCategories
input	tns:ModifyCategoriesRequest
output	tns:ModifyCategoriesResponse
operation	modifyManufacturers
input	tns:ModifyManufacturersRequest
output	tns:ModifyManufacturersResponse
operation	modifyItems
input	tns:ModifyItemsRequest

output	tns:ModifyItemsResponse



Σχήμα 51: VirtueMart_Products_Service

Bindings <binding>

Πίνακας 42: WSDL VirtueMart_Products_Srv <binding> - VirtueMart_Products_SOAP

VirtueMart_Products_SOAP	
type	tns:VirtueMart_Products_Service
transport	"http://schemas.xmlsoap.org/soap/http"
operation	getCategories
soapAction	http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/getCategories
style	document
input	literal
output	literal
operation	getManufacturers
soapAction	http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/getManufacturers
style	document
input	literal
output	literal
operation	getItems
soapAction	http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/getItems
style	document
input	literal
output	literal
operation	modifyCategories
soapAction	http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/modifyCategories

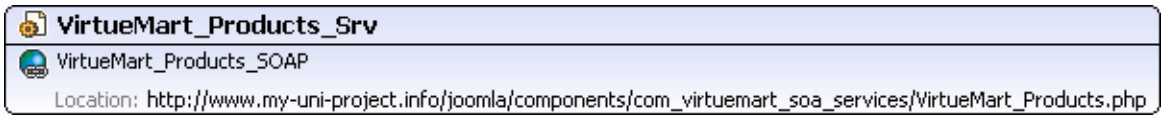
style	document
input	literal
output	literal
operation	modifyManufacturers
soapAction	http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/modifyManufacturers
style	document
input	literal
output	literal
operation	modifyItems
soapAction	http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/modifyItems
style	document
input	literal
output	literal

Σχήμα 52: VirtueMart_Products_SOAP

Υπηρεσίες <service>

Πίνακας 43: WSDL VirtueMart_Products_Srv <service> - VirtueMart_Products_Srv

VirtueMart_Products_Srv	
Port name	VirtueMart_Customers_Srv_SOAP
Port binding	tns:VirtueMart_Products_SOAP
location	http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/VirtueMart_Products.php



Σχήμα 53: VirtueMart_Products_Srv

5.2.4 VirtueMart_Orders_Srv

WSDL location: "http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/VirtueMart_Orders.wsdl"

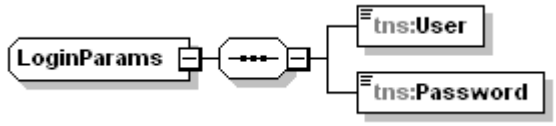
targetNamespace: "http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/VirtueMart_ Orders"

Τύποι <types>

LoginParams: Παράμετροι σύνδεσης χρήστη στο ηλεκτρονικό κατάστημα

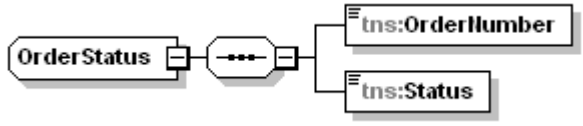
Πίνακας 44: WSDL VirtueMart_Orders_Srv <types> - LoginParams

Πεδίο	Τύπος
User	string
Password	string



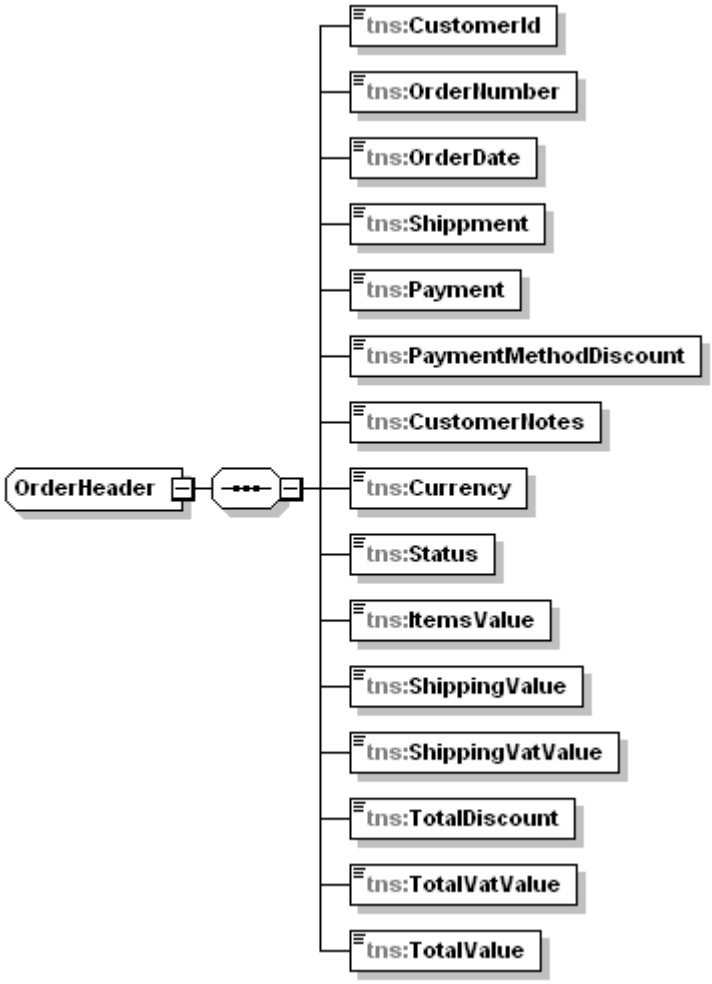
Σχήμα 54: LoginParams

OrderStatus: βλέπε δομή δεδομένων "**Κατάσταση Παραγγελίας** "



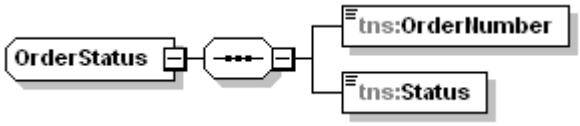
Σχήμα 55: OrderStatus

OrderHeader: βλέπε δομή δεδομένων "**Παραγγελίες – Βασικά στοιχεία**"



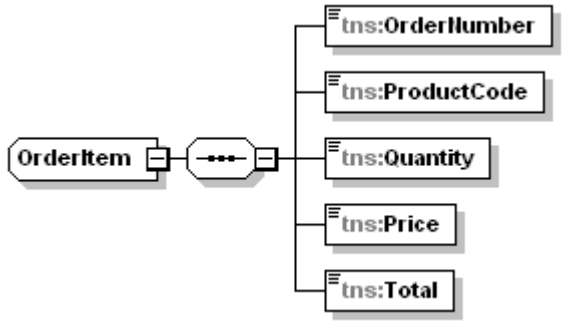
Σχήμα 56: OrderHeader

OrderStatus: βλέπε δομή δεδομένων "**Κατάσταση Παραγγελίας**"



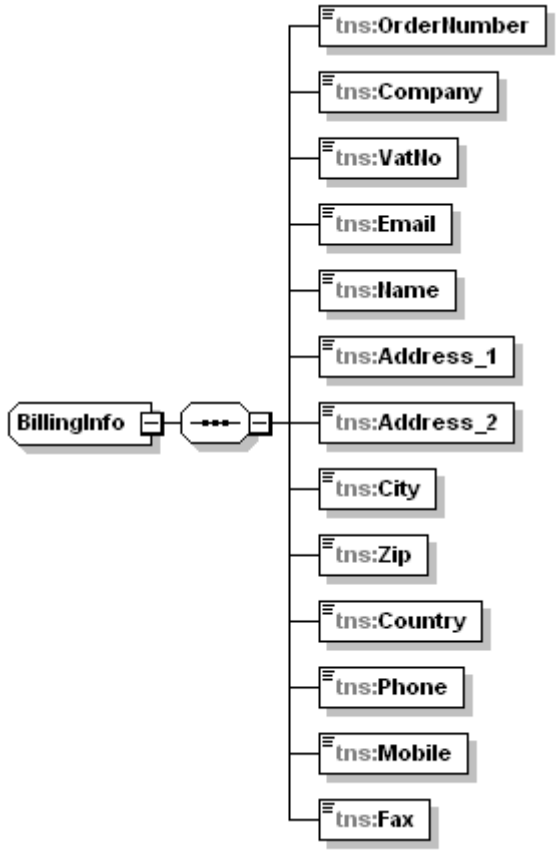
Σχήμα 57: OrderStatus

OrderItem: βλέπε δομή δεδομένων "**Παραγγελίες – Γραμμές ειδών**"



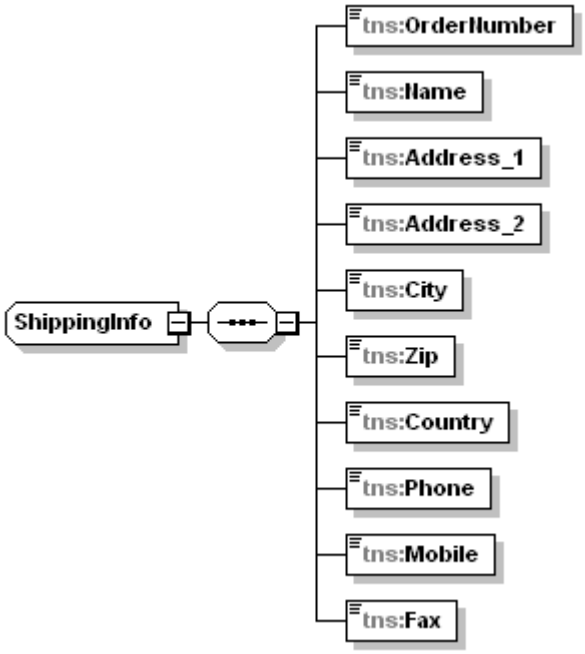
Σχήμα 58: OrderItem

BillingInfo: βλέπε δομή δεδομένων "**Παραγγελίες – Στοιχεία τιμολόγησης**"



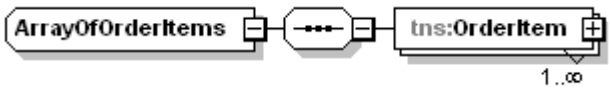
Σχήμα 59: BillingInfo

ShippingInfo: βλέπε δομή δεδομένων "**Παραγγελίες – Στοιχεία αποστολής**"



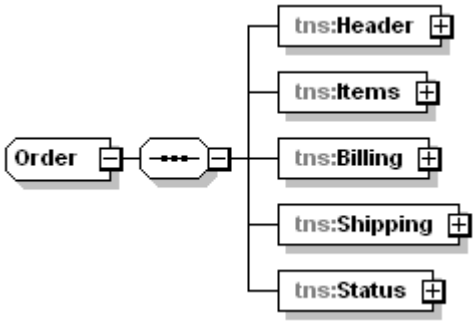
Σχήμα 60: ShippingInfo

ArrayOfOrderItems: Πίνακας γραμμών ειδών παραγγελίας



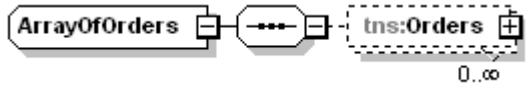
Σχήμα 61: ArrayOfOrderItems

Order: Δομή που περιέχει όλες τις δομές που συνθέτουν μια παραγγελία.



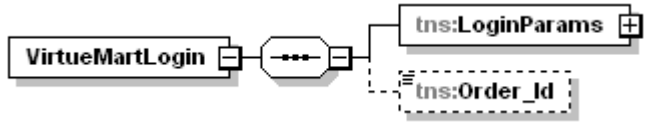
Σχήμα 62: Order

ArrayOfOrders: Πίνακας παραγγελιών



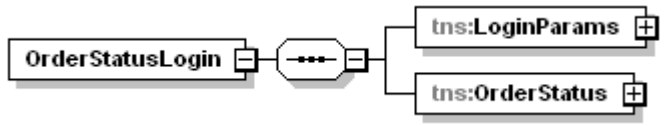
Σχήμα 63: ArrayOfOrders

VirtueMartLogin: Δομή που εμπεριέχει τις παραμέτρους σύνδεσης χρήστη και προαιρετικά έναν αριθμό παραγγελίας, για την ανάκτηση των παραγγελιών.



Σχήμα 64: VirtueMartLogin

OrderStatusLogin: Δομή που εμπεριέχει τις παραμέτρους σύνδεσης χρήστη για την ενημέρωση των καταστάσεων των παραγγελιών.

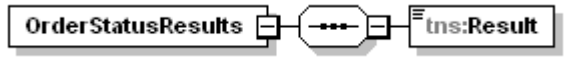


Σχήμα 65: OrderStatusLogin

OrdersResults: Δομή που εμπεριέχει τις ανακτηθέντες παραγγελίες.

Σχήμα 66: OrdersResults

OrderStatusResults: Δομή που εμπεριέχει τα το αποτέλεσμα της ενημέρωσης της κατάστασης παραγγελίας.



Σχήμα 67: OrderStatusResults

Μηνύματα <message>

Πίνακας 45: WSDL VirtueMart_Orders_Srv <message> - OrdersRequest

OrdersRequest	
Parts Name	Parts element
parameters	tns:VirtueMartLogin

Πίνακας 46: WSDL VirtueMart_Orders_Srv <message> - OrdersResponse

OrdersResponse	
Parts Name	Parts element
results	tns:OrdersResults

Πίνακας 47: WSDL VirtueMart_Orders_Srv <message> - OrderStatusRequest

OrderStatusRequest	
Parts Name	Parts element
parameters	tns:OrderStatusLogin

Πίνακας 48: WSDL VirtueMart_Orders_Srv <message> - OrderStatusResponse

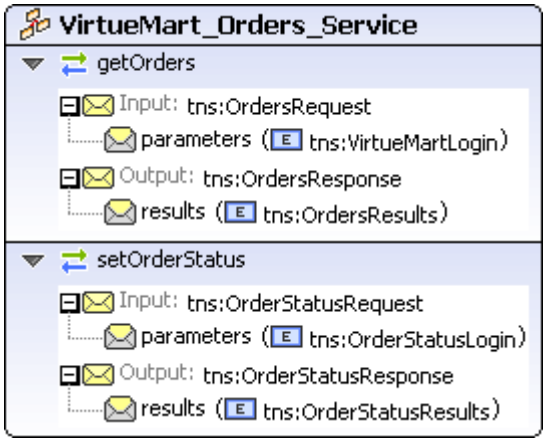
OrderStatusResponse	
Parts Name	Parts element
results	tns:OrderStatusResults

Port types <portType>

Πίνακας 49: WSDL VirtueMart_Orders_Srv <portType> - VirtueMart_Orders_Service

VirtueMart_Orders_Service	
operation	getOrders
input	tns:OrdersRequest
output	tns:OrdersResponse
operation	setOrderStatus
input	tns:OrderStatusRequest

output	tns:OrderStatusResponse



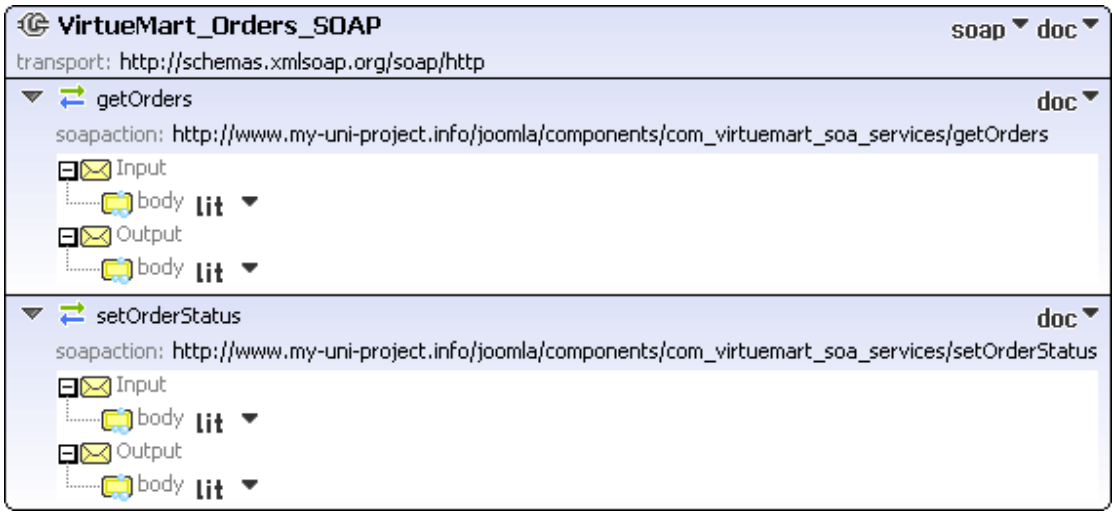
Σχήμα 68: VirtueMart_Orders_Service

Bindings <binding>

Πίνακας 50: WSDL VirtueMart_Orders_Srv <binding> - VirtueMart_Orders_SOAP

VirtueMart_Orders_SOAP	
type	tns:VirtueMart_Orders_Service
transport	"http://schemas.xmlsoap.org/soap/http"
operation	getOrders
soapAction	http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/getOrders
style	document
input	literal
output	literal
operation	setOrderStatus
soapAction	http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/setOrderStatus
style	document

input	literal
output	literal

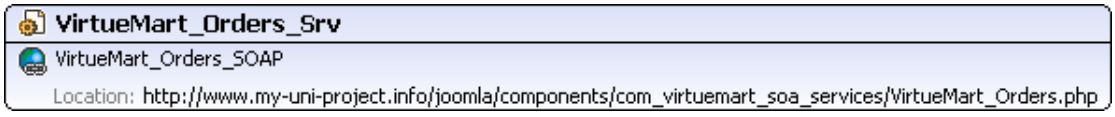


Σχήμα 69: VirtueMart_Orders_SOAP

Υπηρεσίες <service>

Πίνακας 51: WSDL VirtueMart_Orders_Srv <service> - VirtueMart_Orders_Srv

VirtueMart_Orders_Srv	
Port name	VirtueMart_Orders_SOAP
Port binding	tns:VirtueMart_Orders_SOAP
location	http://www.my-uni-project.info/joomla/components/com_virtuemart_soa_services/VirtueMart_Orders.php



Σχήμα 70: VirtueMart_Orders_Srv

ΕΠΙΛΟΓΟΣ

Σε αυτό το κεφάλαιο παρουσιάστηκε η ομαδοποίηση των λειτουργιών των υπηρεσιών ιστού και η δομή των WSDL αρχείων τους. Το επόμενο κεφάλαιο θα αναλύσει τα βήματα που πάρθηκαν για την δημιουργία των SOAP Servers των υπηρεσιών σε PHP επεκτείνοντας την λειτουργικότητα του VirtueMart και χρησιμοποιώντας εσωτερικές κλάσεις του και εσωτερικές κλάσης του Joomla CMS.

ΚΕΦΑΛΑΙΟ 6

ΔΗΜΙΟΥΡΓΙΑ ΤΩΝ ΥΠΗΡΕΣΙΩΝ ΙΣΤΟΥ ΓΙΑ ΤΟ VirtueMart

ΕΙΣΑΓΩΓΗ

Στο παρόν κεφάλαιο θα παρουσιαστούν οι PHP SOAP Servers που δημιουργήθηκαν, το πως επεκτείνεται η λειτουργικότητα του VirtueMart και ο τρόπος ελέγχου χρήστη στο Joomla.

6.1 Επεκτείνοντας την λειτουργικότητα του VirtueMart

Καθώς πρόκειται να επεκταθεί η λειτουργικότητα του VirtueMart θα χρησιμοποιηθούν κλάσεις που προσφέρει το VirtueMart και το Joomla.

Αρχικά αποτρέπεται η απευθείας πρόσβαση στο κώδικα, δηλαδή δεν μπορεί ο χρήστης να τρέξει τον κώδικα τοποθετώντας την url διεύθυνση του php αρχείου στον φυλλομετρητή ιστιοσελίδων του (browser)

```
define( '_VALID_MOS', 1 );
define( '_JEXEC', 1 );
```

Στην συνέχεια ορίζεται μια global μεταβλητή για την αποθήκευση της απόλυτης διαδρομής για το φάκελο ρίζα του joomla

```
global $mosConfig_absolute_path;
```

Ανάκτηση της απόλυτης διαδρομή του φακέλου ρίζα του joomla

```
$mosConfig_absolute_path= realpath( dirname(__FILE__).'/../..' );
```

Η _FILE_ επιστρέφει το όνομα του αρχείου που εκτελείτε, η dirname επιστρέφει το όνομα του φακέλου του αρχείου της παραμέτρου και η realpath μετατρέπει όλα τα σύμβολα '/./', '/../' και τα επιπλέον / και επιστρέφει την διαδρομή που προκύπτει.

Συμπεριλαμβάνεται το αρχείο configuration.php που έχει τις ρυθμίσεις του joomla, αφού πρώτα ελεγχθεί αν υπάρχει στην τρέχουσα διαδρομή.

```
if( file_exists(dirname(__FILE__).'/configuration.php' )) {
     require_once( dirname(__FILE__).'/configuration.php' );
} else {
require_once( $mosConfig_absolute_path.'/configuration.php');
}
```

Επειδή η υπηρεσία αναπτύσσεται ως μέρος του virtuemart και χρησιμοποιούνται functions και αντικείμενά του πρέπει να φορτωθεί ο virtuemart_parser.php

```
if(file_exists(dirname(__FILE__).'/../../components/com_virtuemart
/virtuemart_parser.php' )) {

require_once( dirname(__FILE__).'/../../components/com_virtuemart/
virtuemart_parser.php' );
          $mosConfig_absolute_path =
realpath( dirname(__FILE__).'/../..' );
     } else {
require_once( dirname(__FILE__).'/../../components/com_virtuemart/
virtuemart_parser.php');
     }
```

Για την ενημέρωση δεδομένων στο VirtueMart θα χρησιμοποιηθούν εσωτερικές κλάσεις του. Στις υπηρεσίες που δημιουργήθηκαν συμπεριλήφθηκαν οι παρακάτω κλάσης:

- Για την διαχείριση των χρηστών του ηλεκτρονικού καταστήματος, και κατ' επέκταση των πελατών

```
require_once(CLASSPATH.'ps_user.php');
```

- Για την διαχείριση των παραγγελιών του ηλεκτρονικού καταστήματος

```
require_once(CLASSPATH.'ps_order.php');
```

- Για την διαχείριση των κατασκευαστών των ειδών

```
require_once(CLASSPATH.'ps_manufacturer.php');
```

- Για την διαχείριση των κατηγοριών των ειδών

```
require_once(CLASSPATH.'ps_product_category.php');
```

- Για την διαχείριση των ειδών

```
require_once(CLASSPATH.'ps_product.php');
require_once(CLASSPATH.'ps_product_attribute.php');
```

Μια πλήρη λίστα των αντικειμένων και των μεθόδων που είναι διαθέσιμες στο VirtueMart βρίσκεται στον πίνακα “jos_vm_function” στην βάση του VirtueMart. Αναλυτική τεκμηρίωση είναι διαθέσιμη διεύθυνση http://www.codes-libres.org/apps/VirtueMart_1.1.2_eCommerce_Bundle_Joomla_1.5.9/nav.html?administrator/components/com_virtuemart/classes/ps_product.php.source.html.

6.2 Έλεγχος χρήστη στο Joomla

Είναι απαραίτητο να γίνει έλεγχος του ονόματος και του κωδικού του χρήστη που συμπεριλαμβάνονται στις παραμέτρους κατά την κλήση των υπηρεσιών, για αυτόν το σκοπό δημιουργήθηκε η function onAdminAuthenticate που δέχεται ως παραμέτρους το όνομα και τον κωδικό πρόσβασης του χρήστη και θα επιστρέφει το αποτέλεσμα του ελέγχου με ένα μήνυμα τύπου string (How are passwords encypted 2007).

Δήλωση της function

```
function onAdminAuthenticate($login,$passwd){
```

Εισαγωγή της κλάσης διαχείρισης χρηστών του joomla

```
jimport('joomla.user.helper');
```

Αρχικοποίηση της μεταβλητής $response που θα αποθηκεύσει το μήνυμα του αποτελέσματος.

```
$response="false";
```

Δημιουργία του ερωτήματος για την αναζήτηση του χρήστη με όνομα την παράμετρο $login

```
$list  = "SELECT id, username, password, usertype
          FROM `#__users` ";
$list .= "WHERE username='".$login."' ";
$response=$list;
```

Δημιουργία ενός νέου αντικειμένου σύνδεσης με τη βάση δεδομένων χρησιμοποιώντας την κλάση του VirtueMart.

```
$db = new ps_DB;
```

Εκτέλεση του ερωτήματος χρησιμοποιώντας την μέθοδο query() της ps_DB.

```
$db->query($list);
```

Έλεγχος για το αν το ερώτημα επιστρέφει κάποια εγγραφή

```
if($db->next_record()){
```

Ο κωδικός χρήστη αποθηκεύεται στην βάση με την μορφή “κρυπτογραφημένος κωδικός : συμβολοσειρά κωδικοποίησης”. Με την χρήση της explode() και ορίζοντας τον διαχωριστικό χαρακτήρα “:”, ανακτώνται τα δύο μέρη της τιμής του πεδίου password και αποθηκεύονται στον πίνακα $parts.

```
$parts     = explode( ':', $db->f("password") );
```

Η πρώτη θέση του πίνακα, δηλαδή ο κρυπτογραφημένος κωδικός του χρήστη, αποθηκεύεται στην μεταβλητή $crypt.

```
$crypt     = $parts[0];
```

Η δεύτερη θέση του πίνακα, δηλαδή η συμβολοσειρά που χρησιμοποιήθηκε για την κρυπτογράφηση του κωδικό χρήστη, αποθηκεύεται στην μεταβλητή `$salt`.

```
$salt= $parts[1];
```

Στην συνέχεια κρυπτογραφείται η τιμή της παραμέτρου $passwd χρησιμοποιώντας την ίδια συμβολοσειρά κρυπτογράφησης, με την βοήθεια της getCryptedPassword() του Joomla

```
$testcrypt = JUserHelper::getCryptedPassword($passwd, $salt);
```

Έπειτα γίνεται έλεγχος για το αν οι δύο κρυπτογραφημένοι κωδικοί είναι ίδιοι

```
if ($crypt == $testcrypt ) {
```

Μεταβολή της τιμής της μεταβλητής $response σε “no_admin”.

```
$response= "no_admin";
```

Ελέγχεται αν ο τύπος του χρήστη είναι "Super Administrator", από την τιμή του πεδίου “usertype” που επέστρεψε το ερώτημα στην βάση, αν είναι μεταβάλετε η τιμή της μεταβλητής $response σε “true”.

```
if ( $db->f("usertype") == "Super Administrator" ){
      $response= "true";
}
```

Αν οι δύο κρυπτογραφημένοι κωδικοί δεν είναι ίδιοι μεταβάλετε η τιμή της μεταβλητής $response σε “false”.

```
$response= "false";
```

Αν το ερώτημα στην βάση δεν επέστρεψε κάποια εγγραφή μεταβάλετε η τιμή της μεταβλητής $response σε “no_user”.

```
$response="no_user";
```

Τελικά επιστρέφεται η τιμή της μεταβλητής $response.

```
return $response;
```

6.3 Δήλωση κλάσεων για την υπηρεσία ιστού

Στη συνέχεια ορίζονται οι κλάσεις που θα χρησιμοποιηθούν, σύμφωνα με το σχεδιασμό του κεφαλαίου “Λειτουργικότητα, δομές δεδομένων και βασικός σχεδιασμός”.

6.3.1 Κλάσεις της VirtueMart_Store

Χώρα

```
class Country{
     public $Country;
     public $Name;

     function __construct($Country, $Name) {
          $this->Country = $Country;
          $this->Name = $Name;
     }
}
```

Νόμισμα

```
class Currency{
     public $Currency;
     public $Name;

     function __construct($Currency, $Name) {
          $this->Currency = $Currency;
          $this->Name = $Name;
     }
}
```

ΦΠΑ

```
class Vat{
	public $Vat;
	public $Name;

	function __construct($Vat, $Name ) {
		$this->Vat = $Vat;
		$this->Name = $Name;
	}
}
```

Κατάσταση παραγγελίας

```
class Status{
	public $Status;
	public $Name;

	function __construct($Status, $Name) {
		$this->Status = $Status;
		$this->Name = $Name;
	}
}
```

Τρόπος αποστολής

```
class Shippment{
	public $Shippment;
	public $Name;

	function __construct($Shippment, $Name) {
		$this->Shippment = $Shippment;
		$this->Name = $Name;
	}
```

```
}
```

Τρόπος πληρωμής

```
class Payment{
     public $Payment;
     public $Name;

     function __construct($Payment, $Name) {
          $this->Payment = $Payment;
          $this->Name = $Name;
     }
}
```

6.3.2 Κλάσεις της VirtueMart_Customers

Πελάτης

```
class Customer{
     public $CustomerId;
     public $VatNo;
     public $CompanyName;
     public $BillingName;
     public $Email;
     public $Address_1;
     public $Address_2;
     public $City;
     public $Zip;
     public $Country;
     public $Phone_1;
     public $Phone_2;
     public $Fax;
     public $Permissions;
```

```
     function __construct($CustomerId, $VatNo, $CompanyName,
$BillingName, $Email, $Address_1, $Address_2, $City, $Zip,
$Country, $Phone_1, $Phone_2, $Fax, $Permissions) {
          $this->CustomerId = $CustomerId;
          $this->VatNo = $VatNo;
          $this->CompanyName = $CompanyName;
          $this->BillingName = $BillingName;
          $this->Email = $Email;
          $this->Address_1 = $Address_1;
          $this->Address_2 = $Address_2;
          $this->City = $City;
          $this->Zip = $Zip;
          $this->Country = $Country;
          $this->Phone_1 = $Phone_1;
          $this->Phone_2 = $Phone_2;
          $this->Fax = $Fax;
          $this->Permissions = $Permissions;
     }
}
```

6.3.3 Κλάσεις της VirtueMart_Products

Κατασκευαστής ειδών

```
class Manufacturer{
     public $Manufacturer_Id;
     public $Name;

     function __construct($Manufacturer_Id, $Name) {
          $this->Manufacturer_Id = $Manufacturer_Id;
          $this->Name = $Name;
     }
```

```
}
```

Κατηγορία ειδών

```
class Category {
     public $Category_Id;
     public $Parent_Category;
     public $Name;
     public $Description;
     public $IsActive;
     public $Category_Thumb_Image;
     public $Category_Full_Image;
     public $Products_Per_Row;


     function __construct($Category_Id, $Parent_Category, $Name,
$Description, $IsActive, $Category_Thumb_Image,
$Category_Full_Image, $Products_Per_Row) {
          $this->Category_Id = $Category_Id;
          $this->Parent_Category = $Parent_Category;
          $this->Name = $Name;
          $this->Description = $Description;
          $this->IsActive = $IsActive;
          $this->Category_Thumb_Image = $Category_Thumb_Image;
          $this->Category_Full_Image = $Category_Full_Image;
          $this->Products_Per_Row = $Products_Per_Row;
     }
}
```

Είδος

```
class Item{
     public $Item_Id;
     public $Code;
     public $Name;
     public $Category;
```

```
     public $Manufacturer;
     public $IsActive;
     public $Currency;
     public $Vat;
     public $Unit;
     public $DescriptionShort;
     public $Description;
     public $Price;
     public $DiscPrc;
     public $DiscVal;
     public $Stock;
     public $Availability;
     public $Item_thumb_Image;
     public $Item_Full_Image;


     function __construct($Item_Id, $Code, $Name, $Category,
$Manufacturer, $IsActive, $Currency, $Vat, $Unit,
$DescriptionShort, $Description, $Price, $DiscPrc, $DiscVal,
$Stock, $Availability, $Item_thumb_Image,
            $Item_Full_Image) {
           $this->Item_Id = $Item_Id;
           $this->Code = $Code;
           $this->Name = $Name;
           $this->Category = $Category;
           $this->Manufacturer = $Manufacturer;
           $this->IsActive = $IsActive;
           $this->Currency = $Currency;
           $this->Vat = $Vat;
           $this->Unit = $Unit;
           $this->DescriptionShort = $DescriptionShort;
           $this->Description = $Description;
           $this->Price = $Price;
           $this->DiscPrc = $DiscPrc;
```

```
            $this->DiscVal = $DiscVal;
            $this->Stock = $Stock;
            $this->Availability = $Availability;
            $this->Item_thumb_Image = $Item_thumb_Image;
            $this->Item_Full_Image = $Item_Full_Image;
      }
}
```

6.3.4 VirtueMart_Orders

Παραγγελία

```
class Order{
      public $Header;
      public $Items;
      public $Billing;
      public $Shipping;
      public $Status;

      function __construct($Header, $Items, $Billing, $Shipping,
$Status) {
            $this->Header = $Header;
            $this->Items = $Items;
            $this->Billing = $Billing;
            $this->Shipping = $Shipping;
            $this->Status = $Status;
      }
}
```

Βασικά στοιχεία παραγγελίας

```
class OrderHeader{
      public $CustomerId;
      public $OrderNumber;
```

```
    public $OrderDate;
    public $Shippment;
    public $Payment;
    public $PaymentMethodDiscount;
    public $CustomerNotes;
    public $Currency;
    public $Status;
    public $ItemsValue;
    public $ShippingValue;
    public $ShippingVatValue;
    public $TotalDiscount;
    public $TotalVatValue;
    public $TotalValue;


    function __construct($CustomerId, $OrderNumber, $OrderDate,
$Shippment, $Payment, $PaymentMethodDiscount, $CustomerNotes,
$Currency, $Status, $ItemsValue, $ShippingValue,
$ShippingVatValue, $TotalDiscount, $TotalVatValue, $TotalValue) {
        $this->CustomerId = $CustomerId;
        $this->OrderNumber = $OrderNumber;
        $this->OrderDate = $OrderDate;
        $this->Shippment = $Shippment;
        $this->Payment = $Payment;
        $this->PaymentMethodDiscount = $PaymentMethodDiscount;
        $this->CustomerNotes = $CustomerNotes;
        $this->Currency = $Currency;
        $this->Status = $Status;
        $this->ItemsValue = $ItemsValue;
        $this->ShippingValue = $ShippingValue;
        $this->ShippingVatValue = $ShippingVatValue;
        $this->TotalDiscount = $TotalDiscount;
        $this->TotalVatValue = $TotalVatValue;
        $this->TotalValue = $TotalValue;
```

```
	}
}
```

Γραμμές ειδών παραγγελίας

```
class OrderItem{
	public $OrderNumber;
	public $ProductCode;
	public $Quantity;
	public $Price;
	public $Total;

	function __construct($OrderNumber, $ProductCode, $Quantity, $Price, $Total) {
		$this->OrderNumber = $OrderNumber;
		$this->ProductCode = $ProductCode;
		$this->Quantity = $Quantity;
		$this->Price = $Price;
		$this->Total = $Total;
	}
}
```

Στοιχεία τιμολόγησης παραγγελίας

```
class BillingInfo{
	public $OrderNumber;
	public $Company;
	public $VatNo;
	public $Email;
	public $Name;
	public $Address_1;
	public $Address_2;
	public $City;
	public $Zip;
	public $Country;
```

```
     public $Phone;
     public $Mobile;
     public $Fax;


     function __construct($OrderNumber, $Company, $VatNo, $Email,
$Name, $Address_1, $Address_2, $City, $Zip, $Country, $Phone,
$Mobile, $Fax) {
          $this->OrderNumber = $OrderNumber;
          $this->Company = $Company;
          $this->VatNo = $VatNo;
          $this->Email = $Email;
          $this->Name = $Name;
          $this->Address_1 = $Address_1;
          $this->Address_2 = $Address_2;
          $this->City = $City;
          $this->Zip = $Zip;
          $this->Country = $Country;
          $this->Phone = $Phone;
          $this->Mobile = $Mobile;
          $this->Fax = $Fax;
     }
}
```

Στοιχεία αποστολής παραγγελίας

```
class ShippingInfo{
     public $OrderNumber;
     public $Name;
     public $Address_1;
     public $Address_2;
     public $City;
     public $Zip;
     public $Country;
     public $Phone;
```

```
	public $Mobile;
	public $Fax;


	function __construct($OrderNumber, $Name, $Address_1,
$Address_2, $City, $Zip, $Country, $Phone, $Mobile, $Fax) {
		$this->OrderNumber = $OrderNumber;
		$this->Name = $Name;
		$this->Address_1 = $Address_1;
		$this->Address_2 = $Address_2;
		$this->City = $City;
		$this->Zip = $Zip;
		$this->Country = $Country;
		$this->Phone = $Phone;
		$this->Mobile = $Mobile;
		$this->Fax = $Fax;
	}
}
```

Κατάσταση παραγγελίας

```
class OrderStatus{
	public $OrderNumber;
	public $Status;


	function __construct($OrderNumber, $Status) {
		$this->OrderNumber = $OrderNumber;
		$this->Status = $Status;
	}
}
```

6.4 Δημιουργία των υπηρεσιών

Σε αυτή την παράγραφο παρουσιάζεται ο τρόπος προσπέλασης των παραμέτρων που δέχεται μια υπηρεσία κατά την κλήση της, ο τρόπο δόμησης και επιστροφής των αποτελεσμάτων της, ο τρόπος υλοποίησης μεθόδων ανάκτησης δεδομένων από το VirtueMart, ο τρόπος υλοποίησης μεθόδων εισαγωγής και μεταβολής δεδομένων στο VirtueMart και το πως ορίζεται ο SOAP Server και οι μέθοδοί του.

6.4.1 Παράμετροι

Για κάθε μέθοδο που υλοποιεί η υπηρεσία πρέπει να δημιουργηθεί μια function. Οι παράμετροί της πρέπει να έχουν την δομή του μηνύματος που ορίστηκε στο στοιχείο <wsdl:input> του στοιχείου <wsdl:operation> που εσωκλείεται από το <wsdl:port> στοιχείο στο αρχείο WSDL της υπηρεσίας.

Χρησιμοποιώντας ως παράδειγμα την “getStoreBasicParameters” της “VirtueMart_Store_Srv” όπως ορίστηκε στο κεφάλαιο “Σχεδιασμός των WSDL αρχείων” οι παράμετροι που λαμβάνονται κατά την κλήση της υπηρεσίας έχουν την δομή

Σχήμα 71: Παράμετροι εισαγωγής για την getStoreBasicParameters της VirtueMart_Store_Srv

με την “tns:LoginParams” να είναι

Πίνακας 52: Παράμετροι εισαγωγής - tns:LoginParams

Πεδίο	Τύπος
User	string
Password	string

Ορισμός της function με παραμέτρους την $parameters

```
function getStoreBasicParameters($parameters)
```

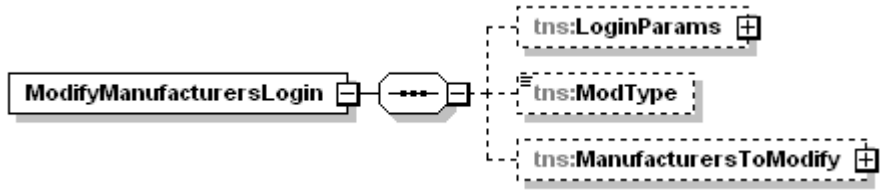
προσπέλαση του ονόματος του χρήστη ακολουθώντας την δομή

```
$parameters->LoginParams->User
```

και αντίστοιχα για τον κωδικό πρόσβασης

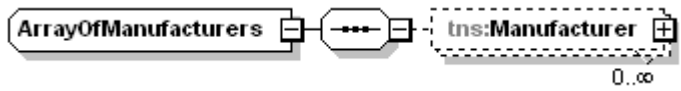
```
$parameters->LoginParams->Password.
```

Σε ποιο πολύπλοκες δομές όπου στις παραμέτρους περιλαμβάνονται πίνακες ακολουθείται ακριβώς η ίδια τεχνική. Για παράδειγμα στην περίπτωση της “modifyManufacturers” της “VirtueMart_Products_Srv”



Σχήμα 72: Παράμετροι εισαγωγής για την modifyManufacturers της VirtueMart_Products_Srv

με την “tns:LoginParams” να είναι ίδια με την παραπάνω, την “tns:ModType” να είναι “string” και την “ ManufacturersToModify” να είναι



Σχήμα 73: ManufacturersToModify της ModifyManufacturersLogin

και την “tns:Manufacturer” να είναι

Πίνακας 53: Παράμετροι εισαγωγής - tns:Manufacturer

Πεδίο	Τύπος
Manufacturer_Id	integer
Name	string

Ορισμός της function με παραμέτρους την $parameters

```
function modifyManufacturers($parameters)
```

Για την προσπέλαση του ονόματος και του κωδικού χρήστη ακολουθείται δομή όπως παρουσιάστηκε παραπάνω

```
$parameters->LoginParams->User
$parameters->LoginParams->Password.
```

Για την παράμετρο “ModType” παρόμοια

```
$parameters->ModType
```

Για τον πίνακα των “Manufacturer” όμως πρώτα πρέπει να αποθηκευτεί κάθε εγγραφή “Manufacturer” του πίνακα σε μια μεταβλητή και στην συνέχεια να προσπελαστούν οι τιμές του. Για να προσπελαστούν όλες τις εγγραφές του πίνακα γίνεται χρήση της foreach()

```
foreach($parameters->ManufacturersToModify->Manufacturer as
$myManufacturer)
```

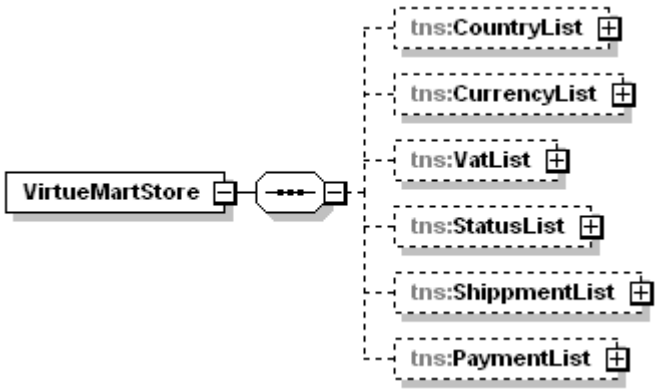
και στην συνέχεια ανακτώνται οι τιμές της $myManufacturer

```
$myManufacturer->Manufacturer_Id
$myManufacturer->Name
```

6.4.2 Αποτελέσματα

Τα αποτελέσματα της μεθόδου της υπηρεσίας πρέπει να έχουν την δομή του μηνύματος που ορίζεται στο στοιχείο <wsdl:output> του στοιχείου <wsdl:operation> που εσωκλείεται από το <wsdl:port> στοιχείο.

Λαμβάνοντας ως παράδειγμα την “getStoreBasicParameters” της “VirtueMart_Store_Srv” όπως ορίστηκε στο κεφάλαιο “Σχεδιασμός των WSDL αρχείων” τότε οι παράμετροι που θα επιστρέψει η υπηρεσία πρέπει να έχουν την δομή



Σχήμα 74: Δομή αποτελεσμάτων για την getStoreBasicParameters της VirtueMart_Store_Srv

με “tns:CountryList”, “tns:CurrencyList”, “tns:VatList”, “tns:StatusList”, “tns:ShippmentList”, “tns:PaymentList” να είναι πίνακες των Χωρών, των Νιμισμάτων, των ΦΠΑ, των Καταστάσεων Παραγγελίας, των Τρόπων Αποστολής και των Τρόπων Πληρωμής αντίστοιχα.

Για να το επιτευχθεί αυτό δημιουργούνται τα επιμέρους αντικείμενα

```
$myCountry= new Country($Id, $Name);
$myCurrency= new Currency($Id, $Name);
$myVat = new Vat($Id, $Name);
```

```
$myStatus= new Status($Id, $Name);
$myShippment= new Shippment($Id, $Name);
$myPayment= new Payment($Id, $Name);
```

τα οποία αποθηκεύουμε σε αντίστοιχους πίνακες

```
$Countries[] = $myCountry;
$Currencies[] = $myCurrency;
$Vats[] = $myVat;
$Statuses[] = $myStatus;
$Shippments[] = $myShippment;
$Payments[] = $myPayment;
```

που στην συνέχεια συνθέτουν την βασική δομή

```
$service_return['CountryList'] = $Countries;
$service_return['CurrencyList'] = $Currencies;
$service_return['VatList'] = $Vats;
$service_return['StatusList'] = $Statuses;
$service_return['ShippmentList'] = $Shippments;
$service_return['PaymentList'] = $Payments;
```

που επιστρέφεται

```
return $service_return;
```

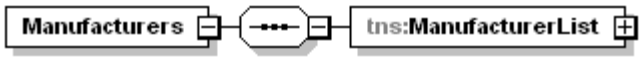
6.4.3 Μέθοδοι ανάκτησης δεδομένων

Έχοντας υπόψιν όλα όσα καλύφθηκαν μέχρι τώρα είναι εύκολο να δημιουργηθεί μια μέθοδος υπηρεσίας για ανάκτηση δεδομένων. Για αυτή την παράγραφο θα χρησιμοποιηθεί η “getManufacturers” της “VirtueMart_Products_Srv”, η οποία χρησιμοποιήθηκε ως παράδειγμα στο κεφάλαιο “Εισαγωγή στην WSDL”.

Η μέθοδος λαμβάνει παραμέτρους το όνομα χρήστη και τον κωδικό πρόσβασης

Σχήμα 75: Παράμετροι εισαγωγής της getManufacturers της VirtueMart_Products_Srv

και επιστρέφει έναν πίνακα με του “Κατασκευαστές”.



Σχήμα 76: Δομή αποτελεσμάτων της getManufacturers της VirtueMart_Products_Srv

Ο ορισμός της function

```
function getManufacturers($parameters) {
```

έλεγχος για το αν ο χρήστης είναι ο Administrator του ηλεκτρονικού καταστήματος

```
$result = onAdminAuthenticate($parameters->LoginParams->User,
$parameters->LoginParams->Password);
```

Αν είναι δημιουργείται ένα αντικείμενο σύνδεσης με τη βάση δεδομένων χρησιμοποιώντας την κλάση του VirtueMart

```
if ($result == "true"){
     $db = new ps_DB;
```

Το ερώτημα ανάκτησης των κατασκευαστών από την βάση

```
     $sql_query =   "SELECT manufacturer_id, mf_name
                      FROM #__{vm}_manufacturer";
```

εκτέλεση του ερωτήματος, τα αποτελέσματά του αποθηκεύονται στην $db

```
     $db->query( $sql_query );
```

Προσπέλαση όλων των εγγραφών που επεστράφησαν

```
     while ($db->next_record()) {
```

αποθήκευση των τιμών των πεδίων της εγγραφής σε μεταβλητές

```
            $Id    = $db->f("manufacturer_id" );
            $Name = $db->f("mf_name" );
```

Δημιουργία ενός αντικειμένου “Κατασκευαστή”

```
            $myManufacturer = new Manufacturer($Id, $Name);
```

αποθήκευση του αντικείμενου σε πίνακα αντικειμένων “Κατασκευαστή”

```
            $Manufacturers[] = $myManufacturer;
      }
```

Αποθήκευση του πίνακα των αντικειμένων Κατασκευαστή στην θέση "ManufacturerList" του πίνακα $service_return

```
      $service_return['ManufacturerList'] = $Manufacturers;
```

Επιστροφή της δομής της μεθόδου

```
      return $service_return;
```

Αν όμως ο χρήστης δεν είναι ο Administrator

```
}else if ($result== "false"){
```

Σε περίπτωση που το password του χρήστη είναι λάθος επιστρέφεται ανάλογο μήνυμα λάθους. Ανατρέχοντας στον ορισμό του WSDL αρχείου και στα μηνύματα της μεθόδου παρατηρείται ότι δεν έχει ορίστει κάποιο στοιχείο <wsdl:fault> με όνομα “name="JoomlaServerAuthFault"”, έτσι παρόλο που η μέθοδος επιστρέφει ένα μήνυμα λάθους το μήνυμα δεν θα μπορέσει να φτάσει ποτέ στον καταναλωτή, ανταυτού θα υπάρξει ένα σφάλμα “timeout”.

```
      return new SoapFault("JoomlaServerAuthFault", "Autification
KO for : ".$params->login);
}else if ($result == "no_admin"){
```

σε περίπτωση που ο χρήστης δεν είναι ο Administrator

```
      return new SoapFault("JoomlaServerAuthFault", "User is not a
Super Administrator : ".$params->login);
}else{
```

σε περίπτωση που ο χρήστης δεν υπάρχει

```
     return new SoapFault("JoomlaServerAuthFault", "User does not
exist : ".$params->login);
}
}
```

Παρόμοια δημιουργήθηκαν όλες οι μέθοδοι ανάκτησης δεδομένων, ακολουθεί ο κατάλογός τους και τα ερωτήματα SQL που χρησιμοποιούνται σε κάθε μια από αυτές.

VirtueMart_Store_Srv

getStoreBasicParameters

Χώρες:

```
SELECT country_id, country_name FROM #__{vm}_country
```

Νομίσματα:

```
SELECT currency_id, currency_name FROM #__{vm}_currency
```

ΦΠΑ:

```
SELECT tax_rate_id, CONCAT(tax_country,' ',tax_rate * 100) as
vat_name
FROM #__{vm}_tax_rate
```

Καταστάσεις Παραγγελίας:

```
SELECT order_status_id, order_status_name FROM
#__{vm}_order_status
```

Τρόποι Αποστολής

```
SELECT shipping_carrier_id, shipping_carrier_name FROM
#__{vm}_shipping_carrier
```

Τρόποι Πληρωμής

```
SELECT payment_method_id, payment_method_name
FROM #__{vm}_payment_method WHERE payment_enabled = 'Y'
```

VirtueMart_Customers_Srv

getCustomers

Πελάτες

```
SELECT
     user_id as CustomerId
     , extra_field_1 as VatNo
     , company as CompanyName
     , concat(ifnull(first_name,''), ' ', ifnull(middle_name,'')
     , ' ', ifnull(last_name,'')) as BillingName
     , user_email as Email
     , Address_1
     , Address_2
     , City
     , Zip
     , ifnull(country_id,country) as Country
     , Phone_1
     , Phone_2
     , Fax
     , perms as Permissions
FROM #__{vm}_user_info cust
LEFT JOIN #__{vm}_country country ON cust.country =
country.country_3_code
WHERE address_type='BT'
```

VirtueMart_Products_Srv

getCategories

Κατηγορίες

```
SELECT
     category_id as Category
     , category_parent_id as Parent_Category
     , category_name as Name
     , category_description as Description
     , category_publish as IsActive
     , Category_Thumb_Image
     , Category_Full_Image
     , Products_Per_Row
FROM #__{vm}_category cat
LEFT JOIN #__{vm}_category_xref catref     ON
cat.category_id=catref.category_child_id
```

getManufacturers

Κατασκευαστές

```
SELECT manufacturer_id, mf_name FROM #__{vm}_manufacturer
```

getItems

Είδη

```
SELECT
     prod.product_id as Item_Id
     , product_sku as Code
     , product_name as Name
     , category_id as Category
     , manufacturer_id as Manufacturer
     , product_publish as IsActive
     , product_currency as Currency
     , product_tax_id as Vat
```

```
     , product_unit as Unit
     , product_s_desc as DescriptionShort
     , product_desc as Description
     , product_price as Price
     , amount * is_percent as DiscPrc
     , amount * ( is_percent -1 ) * ( -1 ) as DiscVal
     , product_in_stock as Stock
     , product_availability as Availability
     , product_thumb_image as Item_thumb_Image
     , product_full_image as Item_Full_Image
FROM #__{vm}_product prod
LEFT JOIN #__{vm}_product_category_xref prodcat
     ON prod.product_id = prodcat.product_id
LEFT JOIN #__{vm}_product_mf_xref prodmanf
     ON prod.product_id = prodmanf.product_id
LEFT JOIN #__{vm}_product_price prodprice
     ON prod.product_id = prodprice.product_id
LEFT JOIN #__{vm}_product_discount proddisc
     ON prod.product_discount_id = proddisc.discount_id
WHERE product_parent_id =0
```

VirtueMart_Orders_Srv

getOrders

Κατάσταση Παραγγελίας (OrderStatus)

```
SELECT
     order_status as Status
     , order_id as OrderNumber
FROM #__{vm}_orders
WHERE order_id='".$OrderNumber."'"
```

Παραγγελίες – Βασικά στοιχεία (OrderHeader)

```
SELECT
     orders.user_id as CustomerId
     , orders.order_id as OrderNumber
     , FROM_UNIXTIME(cdate) as OrderDate
     , shipping_carrier_id as Shippment
     , payment.payment_method_id as Payment
     , payment_method_discount as PaymentMethodDiscount
     , customer_note as CustomerNotes
     , currency_id as Currency
     , order_status_id as Status
     , ifnull(order_subtotal,0) as ItemsValue
     , ifnull(order_shipping,0) as ShippingValue
     , ifnull(order_shipping_tax,0) as ShippingVatValue
     , ifnull(order_discount,0) as TotalDiscount
     , ifnull(order_tax,0) + ifnull(order_shipping_tax,0)
          as TotalVatValue
     , ifnull(order_total,0) as TotalValue
FROM #__{vm}_orders orders
LEFT JOIN #__{vm}_order_payment payment
     ON orders.order_id=payment.order_id
LEFT JOIN #__{vm}_shopper_vendor_xref shopper
     ON orders.user_id=shopper.user_id
     AND orders.vendor_id=shopper.vendor_id
LEFT JOIN #__{vm}_payment_method pmt_method
     ON shopper.shopper_group_id=pmt_method.shopper_group_id
     AND shopper.vendor_id=pmt_method.vendor_id
     AND payment.payment_method_id=pmt_method.payment_method_id
LEFT JOIN #__{vm}_currency currency
     ON orders.order_currency = currency.currency_code
LEFT JOIN #__{vm}_shipping_carrier carrier
```

```
     ON SUBSTRING(SUBSTRING(`ship_method_id`,
     INSTR(ship_method_id, '|') +1,CHAR_LENGTH(ship_method_id)),1,
     INSTR(SUBSTRING(`ship_method_id`,
     INSTR(ship_method_id, '|')+1,
     CHAR_LENGTH(ship_method_id)),'|')-1)
     = carrier.shipping_carrier_name
LEFT JOIN #__{vm}_order_status status
     ON orders.order_status = status.order_status_code
```

Αν υπάρχει παράμετρος “Order_Id” προσθέτουμε

```
WHERE orders.order_id > '".$Order_Id."' "
```

Παραγγελίες – Γραμμές ειδών (OrderItem)

```
SELECT
     order_id as OrderNumber
     , product_sku as ProductCode
     , ifnull(product_quantity,0) as Quantity
     , ifnull(product_final_price,0) as Price
     , ifnull(product_quantity,0) * ifnull(product_final_price,0)
          as Total
FROM #__{vm}_order_item orderItem
INNER JOIN #__{vm}_product product
     ON orderItem.product_id = product.product_id
WHERE order_id = '".$OrderNumber."'"
```

Παραγγελίες – Στοιχεία τιμολόγησης (BillingInfo)

```
SELECT
     order_id as OrderNumber
     , company as Company
     , extra_field_1 as VatNo
     , user_email as Email
     , concat(ifnull(title,''),' ',ifnull(first_name,'')
          ,' ',ifnull(middle_name,''),' ',ifnull(last_name,''))
```

```
          as Name
     , address_1 as Address_1
     , address_2 as Address_2
     , city as City
     , zip as Zip
     , country as Country
     , phone_1 as Phone
     , phone_2 as Mobile
     , fax as Fax
FROM #__{vm}_order_user_info
WHERE order_id = '".$OrderNumber."' AND address_type = 'BT'"
```

Παραγγελίες – Στοιχεία αποστολής (ShippingInfo)

```
SELECT
     order_id as OrderNumber
     , concat(ifnull(title,''),' '
          ,ifnull(first_name,''),' '
          ,ifnull(middle_name,''),' ',ifnull(last_name,''))
          as Name
     , address_1 as Address_1
     , address_2 as Address_2
     , city as City
     , zip as Zip
     , country as Country
     , phone_1 as Phone
     , phone_2 as Mobile
     , fax as Fax
FROM #__{vm}_order_user_info
WHERE order_id = '".$OrderNumber."' AND address_type = 'ST'
```

6.4.4 Μέθοδοι εισαγωγής και μεταβολής δεδομένων

Για την εισαγωγή και μεταβολή δεδομένων στο VirtueMart, όπως ήδη αναφέρθηκε, θα χρησιμοποιηθούν κυρίως κλάσεις που παρέχονται από το ίδιο το VirtueMart. Χρησιμοποιώντας το παράδειγμα της εισαγωγής των ειδών (θα γίνει μόνο μια περιληπτική παρουσίαση) θα αναλυθεί η τεχνική που ακολουθήθηκε.

Αρχικά συμπεριλαμβάνεται η κλάση του VirtueMart για την διαχείριση των κατηγοριών των ειδών.

```
require_once(CLASSPATH.'ps_product_category.php');
```

έπειτα δημιουργείται ένα αντικείμενο της κλάσης

```
$ps_product = new ps_product;
```

Στην συνέχεια αποδίδονται στο αντικείμενο $ps_product τιμές. Υπάρχουν τρεις τρόποι για την απόδοση τιμών

- $_SESSION

  Μέσω μεταβλητών της τρέχουσας “Συνεδρίας”, για παράδειγμα ο τύπος του χρήστη

  ```
  $_SESSION['auth']= "admin";
  ```

  ή το ID του καταστήματος

  ```
  $_SESSION['ps_vendor_id'] = 1;
  ```

- $_REQUEST

  Μέσω τιμών της $_REQUEST, για παράδειγμα

  ```
  $_REQUEST['product_tax_id'] = $myItems->Vat;
  $_REQUEST['product_s_desc'] = $myItems->DescriptionShort;
  ```

- Τιμές πίνακα

Με την δημιουργία ενός πίνακα που έχει θέσεις με ονόματα αντίστοιχα με τα ονόματα των μεταβλητών του αντικειμένου που θα χρησιμοποιηθεί για την εισαγωγή/μεταβολή των δεδομένων, για παράδειγμα

```
$item['product_sku'] = "Κωδικός";
$item['product_name'] = "Όνομα"
```

Για να καθοριστεί ποιος από τους παραπάνω τρεις τρόπους θα χρησιμοποιηθεί, καθώς και τα ονόματα μεταβλητών πρέπει να εξεταστεί η τεκμηρίωση της κλάσης του VirtueMart που θα χρησιμοποιηθεί και στην συνέχεια στην ανάλογη μέθοδό της. Για την περίπτωση της εισαγωγής ειδών η τεκμηρίωση των κλάσεων του VirtueMart βρίσκεται στην διαδρομή:

http://www.codes-libres.org/apps/VirtueMart_1.1.2_eCommerce_Bundle_Joomla_1.5.9/

nav.html?administrator/components/com_virtuemart/classes

Επιλέγοντας από την αριστερή λίστα την "ps_product.php" και από την λίστα των μεθόδων της την "add()" (δεύτερο κλικ στον ορισμό της) θα εμφανίσει τον κώδικα της μεθόδου (add() κώδικας). Στη γραμμή 256 φαίνεται για την απόδοση της τιμής του ID του καταστήματος χρησιμοποιείται η $_SESSION,

```
$ps_vendor_id = $_SESSION["ps_vendor_id"];
```

στην γραμμή 278 φαίνεται ότι για την σύντομη περιγραφή του είδους χρησιμοποιείται η $_REQUEST,

```
'product_s_desc' => vmRequest::getVar('product_s_desc', '',
'default', '', VMREQUEST_ALLOWHTML)
```

και στις γραμμές 275 και 276 φαίνεται ότι για την ανάθεση της τιμής του κωδικού και του ονόματος του είδους χρησιμοποιείται πίνακας τιμών

```
'product_sku' => vmGet($d,'product_sku'),
'product_name' => vmGet($d,'product_name'),
```

Αφού αποδοθούν οι τιμές καλείται η επιθυμητή μέθοδος του αντικειμένου και αποθηκεύουμε το αποτέλεσμά της στην μεταβλητή $result.

```
$result = $ps_product->add($item);
```

Υπάρχουν περιπτώσεις που είναι απαραίτητο να ενημερωθούν δεδομένα χρησιμοποιώντας SQL απευθείας στους πίνακες του VirtueMart, σε αυτή την περίπτωση απλά εκτελείται το κατάλληλο SQL ερώτημα

```
$db = new ps_DB;
$sql_query = "UPDATE  #__{vm}_product
           SET product_full_image='".$myItems->Item_Full_Image."',
           product_thumb_image='".$myItems->Item_thumb_Image."'
           WHERE product_id=".$product_id;
$result = $db->query($sql_query);
```

Ακολουθεί ο κατάλογος των μεθόδων εισαγωγής/μεταβολής δεδομένων που υλοποιήθηκαν:

VirtueMart_Products_Srv

- Εισαγωγή / μεταβολή Κατασκευαστών Ειδών

  modifyManufacturers

- Εισαγωγή / μεταβολή Κατηγοριών Ειδών

  modifyCategories

- Εισαγωγή / μεταβολή Ειδών

  modifyItems

VirtueMart_Orders_Srv

- Μεταβολή Κατάστασης Παραγγελίας

  setOrderStatus

6.4.5 Ορισμός του SOAP Server και των μεθόδων του

Το τελευταίο βήμα είναι να ορισθεί ο SOAP Server και οι μέθοδοί του. Δημιουργείται έναν νέο αντικείμενο SOAP Server και δηλώνεται ποιο αρχείο WSDL θα χρησιμοποιηθεί,

```
$server = new SoapServer("VirtueMart_Orders.wsdl");
```

προστίθενται οι functions που έχουν οριστεί,

```
$server->addFunction("getOrders");
$server->addFunction("setOrderStatus");
```

και ενεργοποιείται ο SOAP Server ο οποίος θα χειρίζεται τις κλήσεις SOAP καλώντας τις απαραίτητες functions και επιστρέφοντας τα αποτελέσματα

```
$server->handle();
```

ΕΠΙΛΟΓΟΣ

Σε αυτό το κεφάλαιο αναλύθηκε ο τρόπος δημιουργίας των SOAP Servers σε PHP. Στο επόμενο και τελευταίο κεφάλαιο παρουσιάζεται η εφαρμογή ενδιάμεσου λογισμικού που κάνει χρήση των υπηρεσιών ιστού που δημιουργήθηκαν για να ανακτήσει και να καταχωρήσει δεδομένα στο ηλεκτρονικό κατάστημα.

ΚΕΦΑΛΑΙΟ 7

ΥΛΟΠΟΙΗΣΗ ΕΝΔΙΑΜΕΣΟΥ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΣΥΝΔΕΣΗ ΤΟΥ VirtueMart ΜΕ ΤΟ Soft1 ERP

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο παρουσιάζεται η εφαρμογή ενδιάμεσου λογισμικού που δημιουργήθηκε για διασύνδεση του ERP με το ηλεκτρονικό κατάστημα. Αρχικά παρουσιάζεται η διεπαφή χρήστη, στην συνέχεια καλύπτεται ο τρόπος σύνδεσης σε υπηρεσίες ιστού μέσα από το Visual Studio 2008 και την C#, και τέλος υπάρχει μια επισκόπηση των κλάσεων και των μεθόδων της εφαρμογής.

7.1 Διεπαφή χρήστη

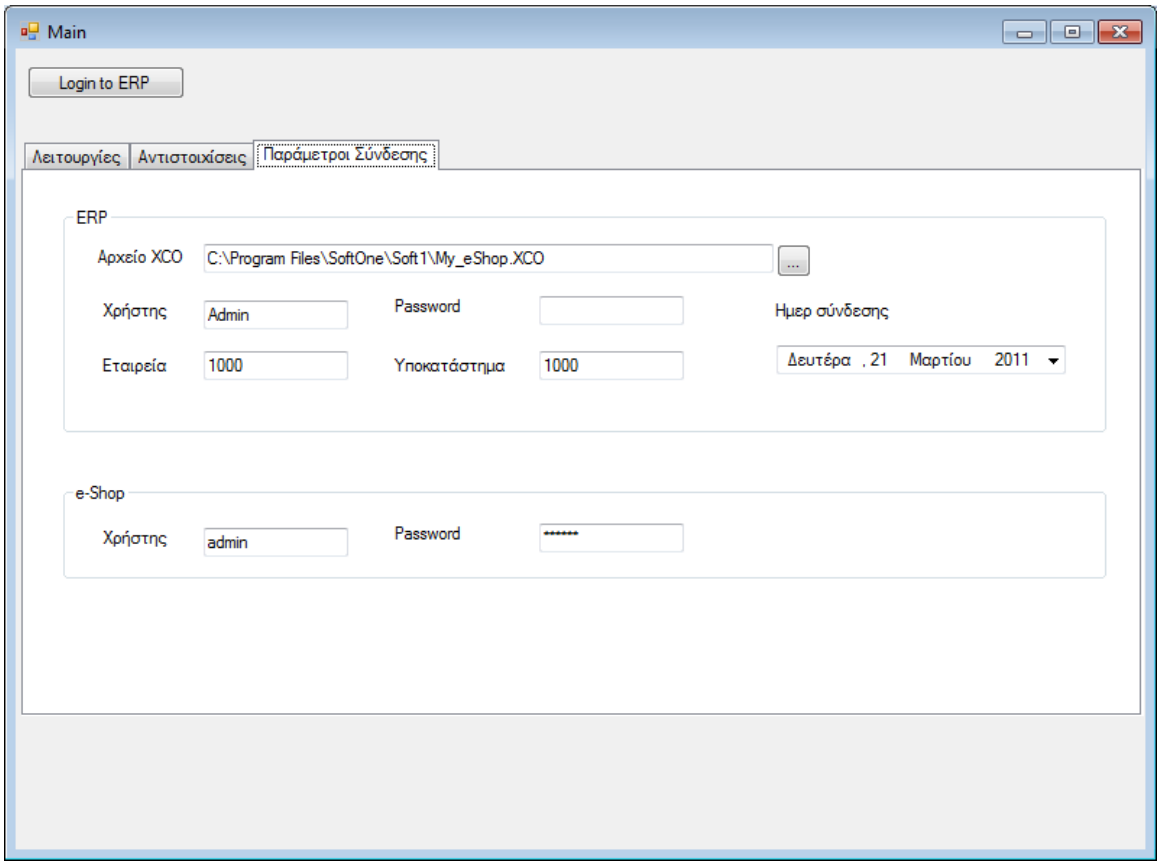
Σύμφωνα με τον βασικό σχεδιασμό του ενδιάμεσου λογισμικού (κεφάλαιο “Λειτουργικότητα, δομές δεδομένων και βασικός σχεδιασμός”) η υλοποίηση της διεπαφής χωρίζεται σε τρεις λειτουργικές ενότητες:

- Παράμετροι διασύνδεσης με το ERP και το ηλεκτρονικό κατάστημα
- Αντιστοιχίσεις βασικών αρχείων ERP και ηλεκτρονικού καταστήματος
- Λειτουργίες ενημέρωσης μεταξύ ERP και ηλεκτρονικού καταστήματος

7.1.1 Παράμετροι διασύνδεσης με το ERP και το ηλεκτρονικό κατάστημα

Η καρτέλα των παραμέτρων σύνδεσης χωρίζεται σε δύο ενότητες με την χρήση δύο GroupBox. Η πρώτη αφορά το ERP και περιέχει το αρχείο XCO (αρχείο σύνδεσης για το Sof1 ERP), το όνομα του χρήστη, τον κωδικό του χρήστη, καθώς και την ημερομηνία σύνδεσης, την εταιρεία και το υποκατάστημα της εταιρείας.

Στην ενότητα του e-Shop υπάρχουν το όνομα του χρήστη και ο κωδικός πρόσβασης.



Σχήμα 77: Διεπαφή χρήστη - Παράμετροι Σύνδεσης

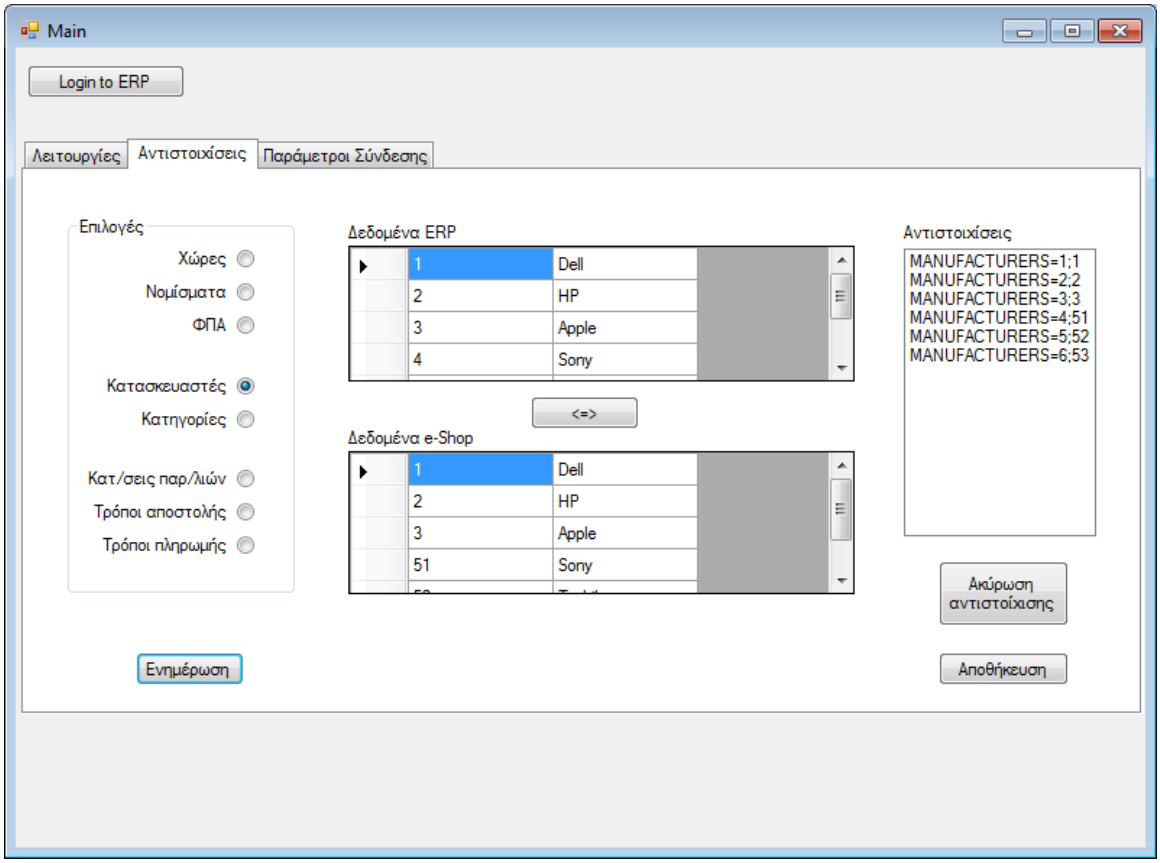
7.1.2 Αντιστοιχίσεις βασικών αρχείων ERP και ηλεκτρονικού καταστήματος

Η καρτέλα για την αντιστοίχιση των βασικών αρχείων έχει χωριστεί σε τρεις κάθετες ενότητες. Στα αριστερά υπάρχουν οι επιλογές των αρχείων για αντιστοίχιση. Οι επιλογές έχουν ομαδοποιηθεί με την χρήση ενός GroupBox και αναπαριστώνται με οχτώ RadioButtons, χωρισμένα με κενά σε τρεις επιμέρους ενότητες. Πρώτα τα βασικά αρχεία της εταιρείας, έπειτα τα βασικά αρχεία των ειδών και τέλος τα βασικά αρχεία των παραγγελιών. Στο κάτω μέρος της αριστερής ενότητας υπάρχει το Button “Ενημέρωση” που ανάλογα με την επιλογή

του χρήστη ανακτά τα αρχεία από το ηλεκτρονικό κατάστημα, το ERP και τις αποθηκευμένες αντιστοιχίσεις από το αρχείο παραμέτρων της εφαρμογής (ERP_to_VM.ini).

Στην κεντρική ενότητα υπάρχουν δύο DataGridViews και ανάμεσά τους ένα Button με την ένδειξη “<=>”. Το πάνω DataGridView εμφανίζει τα δεδομένα του ERP, ενώ το κάτω τα δεδομένα από το ηλεκτρονικό κατάστημα. Το Button αντιστοιχεί τις επιλεγμένες τιμές των δύο DataGridViews.

Στην δεξιά πλευρά υπάρχει ένα ListBox και δύο Buttons. Στο ListBox εμφανίζονται οι αντιστοιχίσεις που έχουν ήδη γίνει (από το αρχείο παραμέτρων) και οι αντιστοιχίσεις που μόλις έχει πραγματοποιήσει ο χρήστης. Με το Button “Ακύρωση αντιστοίχισης” μπορεί να ακυρωθεί η επιλεγμένη αντιστοίχηση (η αντιστοίχηση προς ακύρωση επιλέγεται στο ListBox), ενώ με το Button “Αποθήκευση” οι τιμές του ListBox αποθηκεύονται στο αρχείο παραμέτρων της εφαρμογής.



Σχήμα 78: Διεπαφή χρήστη - Αντιστοιχίσεις

7.1.3 Λειτουργίες ενημέρωσης μεταξύ ERP και ηλεκτρονικού καταστήματος

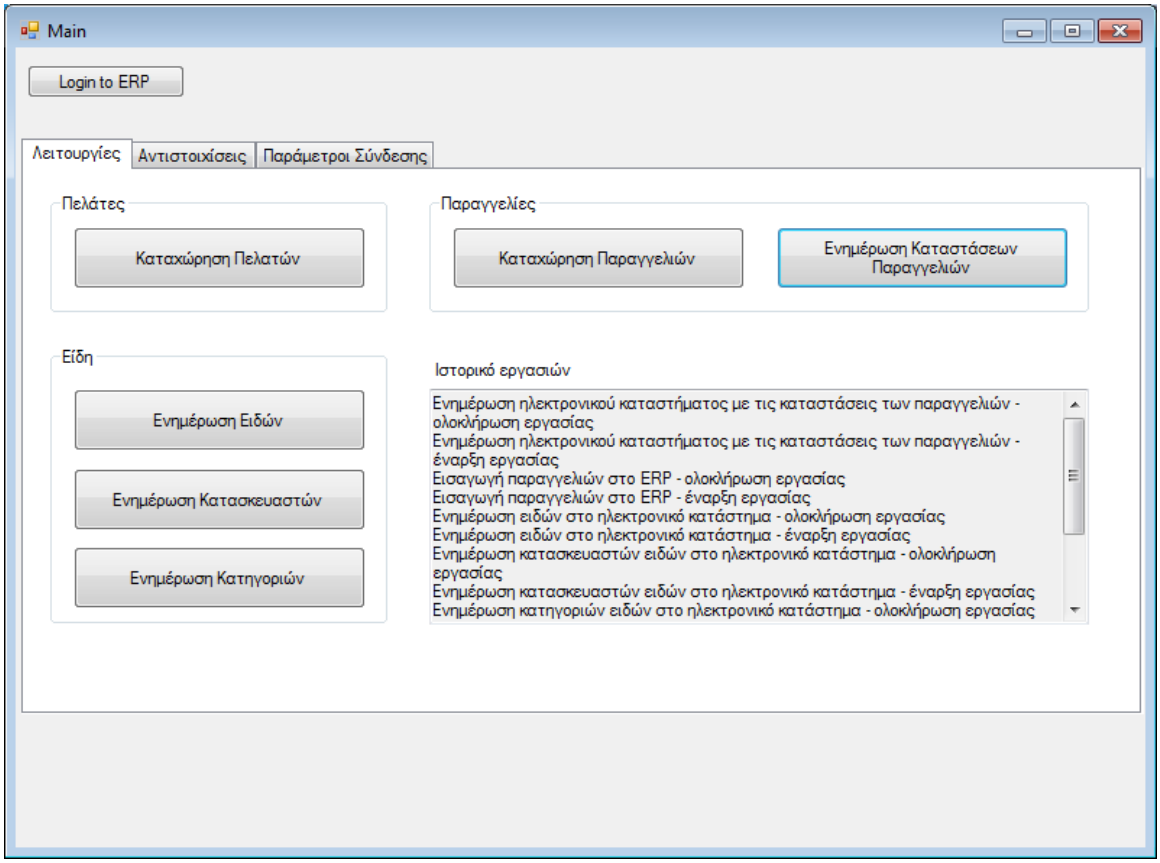
Η καρτέλα “Λειτουργίες” έχει χωριστεί σε τέσσερα τμήματα. Για το διαχωρισμό τους χρησιμοποιήθηκαν τρία GroupBox, ενώ η τέταρτη ενότητα είναι ένα TextBox με ενεργοποιημένη την ιδιότητα “Multiline”.

Πάνω αριστερά βρίσκεται η ενότητα “Πελάτες” η οποία περιέχει μόνο το Button “Καταχώριση Πελατών”. Η “Καταχώριση Πελατών” μεταφέρει τους νέους πελάτες από το ηλεκτρονικό κατάστημα στο ERP, καθώς επίσης ενημερώνει τα στοιχεία των πελατών του ERP που ήδη υπάρχουν.

Κάτω αριστερά βρίσκεται η ενότητα “Είδη” η οποία περιέχει τρία Buttons για τις λειτουργίες ενημέρωση αρχείων σχετικών με την διαχείριση ειδών. Η λειτουργία “Ενημέρωση Ειδών” καταχωρεί νέα είδη και ενημερώνει τα στοιχεία των προϋπαρχόντων ειδών του ηλεκτρονικού καταστήματος, συμπεριλαμβανομένου και του αποθέματός τους. Η λειτουργία “Ενημέρωση Κατασκευαστών” ενημερώνει τους κατασκευαστές ειδών, ενώ η λειτουργία “Ενημέρωση Κατηγοριών” ενημερώνει τις κατηγορίες ειδών του ηλεκτρονικού καταστήματος.

Πάνω δεξιά υπάρχει η ενότητα “Παραγγελίες” με δύο Buttons σε οριζόντια παράθεση. Με το “Καταχώρηση Παραγγελιών” καταχωρούνται οι παραγγελίες του ηλεκτρονικού καταστήματος στο ERP, ενώ με το “Ενημέρωση Καταστάσεων Παραγγελιών” ενημερώνονται οι καταστάσεις των παραγγελιών του ηλεκτρονικού καταστήματος.

Τέλος κάτω δεξιά υπάρχει το TextBox “Ιστορικό εργασιών” όπου εμφανίζονται μηνύματα σχετικά με τις λειτουργίες που πραγματοποιούνται από την εφαρμογή.



Σχήμα 79: Διεπαφή χρήστη - Λειτουργίες

7.1.4 Περαιτέρω σχεδιαστικές παράμετροι της διεπαφής χρήστη

Καθώς η πρώτη λειτουργία που πρέπει να πραγματοποιήσει ο χρήστης είναι να συνδεθεί στο ERP το Button “Login to ERP” επιλέχθηκε να είναι έξω από της καρτέλες στο πάνω αριστερό μέρος της εφαρμογής. Επίσης κατά την έναρξη της εφαρμογή όλα τα Buttons είναι ανενεργά και ενεργοποιούνται μόνο έπειτα από επιτυχή σύνδεση με το ERP.

Οι καρτέλες των λειτουργιών έχουν παρατεθεί με σειρά συχνότητας χρήσης. Έτσι πρώτη βρίσκεται η καρτέλα “Λειτουργίες” που περιέχει τις βασικές λειτουργίες της εφαρμογής. Έπειτα υπάρχει η καρτέλα “Αντιστοιχίσεις” οι λειτουργίες της οποίας

θα χρησιμοποιηθούν κυρίως κατά την αρχική παραμετροποίηση της διασύνδεσης του ηλεκτρονικού καταστήματος και του ERP. Ενώ τελευταία βρίσκεται η καρτέλα “Παράμετροι Σύνδεση” τα στοιχεία της οποίας θα ενημερωθούν στην πρώτη χρήση της εφαρμογής και μετέπειτα θα παραμείνουν σταθερά.

Στην καρτέλα “Λειτουργίες” επιλέχθηκε τα Buttons να έχουν μεγάλο μέγεθος και η διάταξη των στοιχείων να είναι απλή και λιτή για την διευκόλυνση του χρήστη.

Οι λέξη “Ενημέρωση” αναφέρεται σε λειτουργίες που έχουν να κάνουν με μεταφορά δεδομένων προς το ηλεκτρονικό κατάστημα, ενώ η λέξη “Καταχώρηση” αναφέρεται σε λειτουργίες μεταφοράς δεδομένων προς το ERP. Ο λόγος αυτής της διάκρισης είναι επειδή ο χρήστης στην καθημερινή εργασία του “καταχωρεί” στοιχεία στο ERP σύστημα της εταιρείας, και αντίστοιχα “ενημερώνει” το ηλεκτρονικό κατάστημα με αλλαγές που έχουν πραγματοποιηθεί στο ERP σύστημα.

Στο “Ιστορικό ενεργειών” οι νέες ενέργειες εμφανίζονται στην κορυφή “απωθώντας” τις προγενέστερες ενέργειες προς το κάτω μέρος του στοιχείου. Έτσι ο χρήστης συνεχώς μπορεί να δει άμεσα τις πιο πρόσφατες ενέργειες που έχει πραγματοποιήσει, ενώ επίσης κάνοντας χρήση του scrollbar που εμφανίζεται να δει παλαιότερες.

7.2 Αρχείο παραμέτρων εφαρμογής

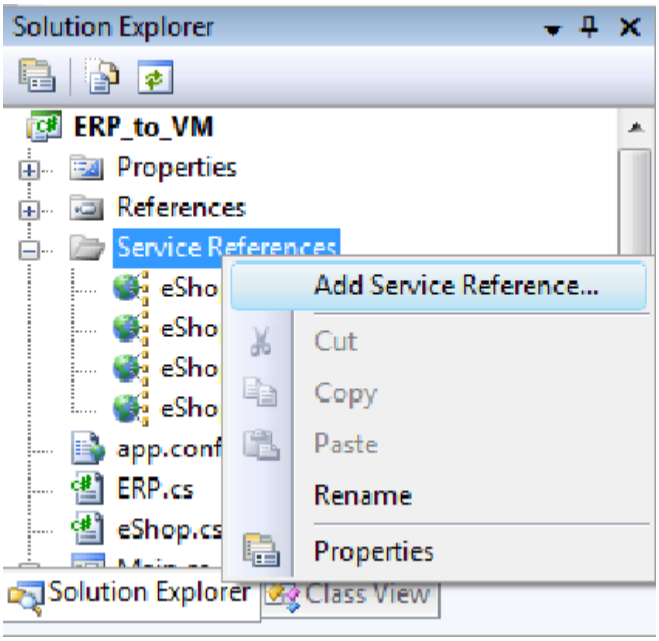
Η εφαρμογή δημιουργεί και ενημερώνει ένα αρχείο παραμέτρων, το “ERP_to_VM.ini”. Το αρχείο ενημερώνεται κάθε φορά που γίνεται σύνδεση με το ERP (λειτουργία “Login to ERP”), και κάθε φορά που εκτελείται η λειτουργία “Αποθήκευση” από την καρτέλα “Αντιστοιχίσεις”. Οι παράμετροι που

αποθηκεύονται είναι δύο μορφών, οι αντιστοιχήσεις και οι παράμετροι σύνδεσης. Οι αντιστοιχήσεις έχουν την σύνταξη “ΠΑΡΑΜΕΤΡΟΣ = Τιμή ERP;Τιμή ηλεκτρονικού καταστήματος”, ενώ οι παράμετροι σύνδεσης “ΠΑΡΑΜΕΤΡΟΣ=Τιμή”. Παρακάτω παρατίθεται ένα παράδειγμα των περιεχομένων του αρχείου παραμέτρων.

```
COUNTRY=1000;84
CURRENCY=100;47
VATS=1310;4
VATS=1210;3
MANUFACTURERS=1;1
MANUFACTURERS=6;53
SHIPMENT=1;1
PAYMENT=1001;3
PAYMENT=1003;1
PAYMENT=1006;2
CATEGORIES=1;62
STATUS=2;2
STATUS=3;3
STATUS=4;4
PARAM_COMPANY=1000
PARAM_BRANCH=1000
PARAM_USER=Admin
PARAM_PASSWORD=
PARAM_XCO=C:\Program Files\SoftOne\Soft1\My_eShop.XCO
PARAM_ESHOP_USER=admin
PARAM_ESHOP_PASSWORD=
```

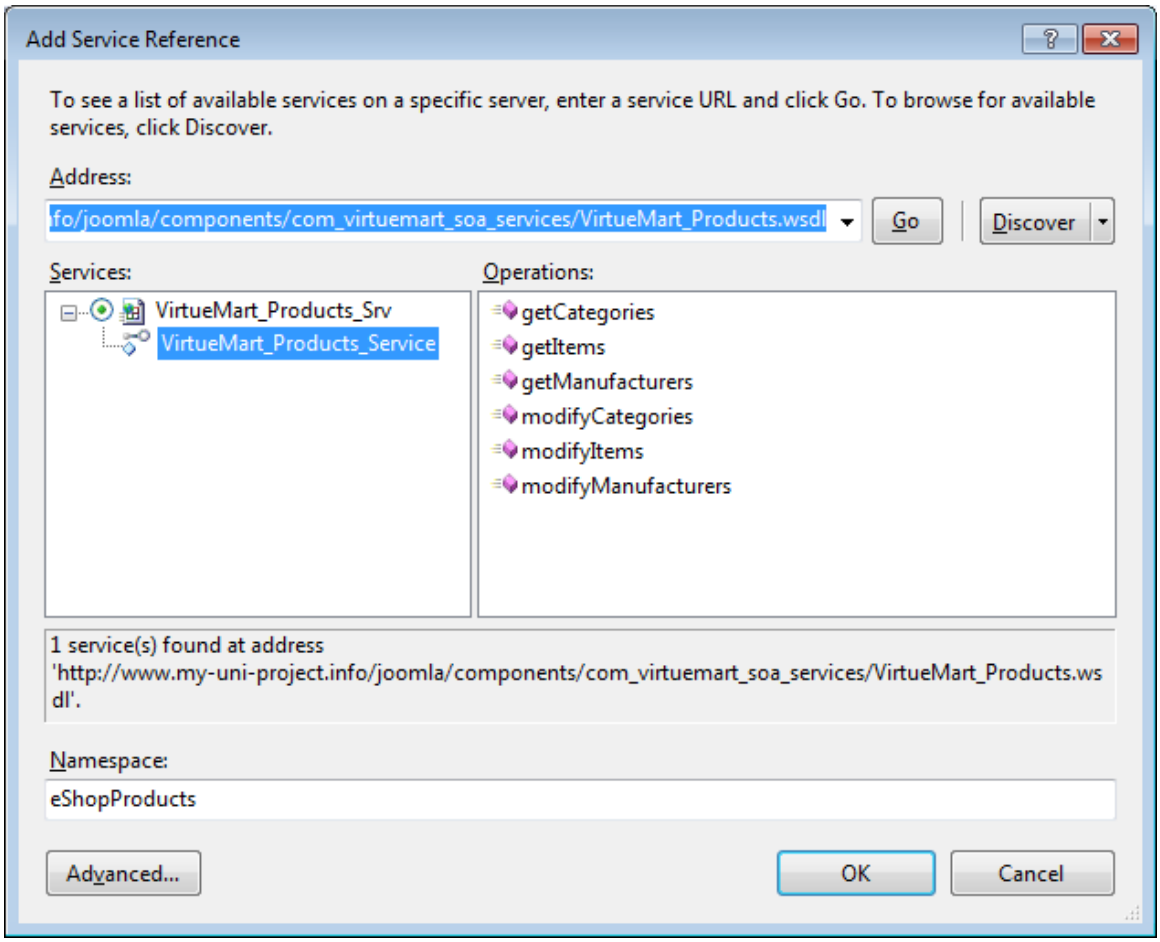
7.3 Σύνδεση σε υπηρεσίες ιστού με το Visual Studio 2008

Για να χρησιμοποιηθεί μια υπηρεσία ιστού στο Visual Studio 2008 πρέπει πρώτα να δημιουργηθεί μια αναφορά σε αυτή (service reference). Πρώτα στον “Solution Explorer” και στο “Service References” με δεξί κλικ επιλέγεται η ένδειξη “Add Service Reference...”.



Σχήμα 80: Service Reference - Add Service Reference 1

Στον οδηγό που εμφανίζεται και στο πεδίο “Address” μπαίνει η διεύθυνση που υπάρχει το WSDL αρχείο της υπηρεσίας και έπειτα επιλέγεται το κουμπί “Go”.



Σχήμα 81: Service Reference - Add Service Reference 2

Στο αριστερό μέρος εμφανίζεται το όνομα της υπηρεσίας, αν αναπτυχθεί η δομή εμφανίζονται όλα τα στοιχεία <wsdl:portType> του WSDL αρχείου και αν επιλεχθεί ένα από αυτά, στα δεξιά εμφανίζονται όλα τα στοιχεία <wsdl:operation>.

Στο κάτω μέρος του οδηγού υπάρχει το πεδίο “Namespace” όπου ορίζεται το όνομα της περιοχής ονομάτων που αναφέρεται σε αυτή την υπηρεσία ιστού.

Για την εφαρμογή δημιουργήθηκαν τέσσερις αναφορές υπηρεσία ιστού που αντιστοιχούν στις τέσσερις υπηρεσίες που δημιουργήθηκαν για το ηλεκτρονικό

κατάστημα. Η eShopStore αντιστοιχεί στην VirtueMart_Store, η eShopCustomers στην VirtueMart_Customers, η eShopProducts στην VirtueMart_Products και η eShopOrders στην VirtueMart_Orders.

7.4 Κλήση των υπηρεσιών ιστού με C#

Αφού έχουν δημιουργηθεί η αναφορά σε μια υπηρεσία ιστού, το επόμενο βήμα είναι να δημιουργηθεί μια μέθοδο για την κλήση της υπηρεσίας. Παρακάτω θα χρησιμοποιηθεί ως παράδειγμα η μέθοδο getEShopStore() της κλάσης “eShop” της εφαρμογής.

Αρχικά δηλώνεται η μέθοδος που επιστρέφει ένα αντικείμενο τύπου eShopStore.getStoreBasicParametersResponse, και λαμβάνει ως παραμέτρους το όνομα του χρήστη και τον κωδικό πρόσβασής του.

```
public eShopStore.getStoreBasicParametersResponse
getEShopStore(string strUser, string strPass)  {
```

Έπειτα δημιουργείται ένα αντικείμενο τύπου eShopStore.getStoreBasicParametersRequest που θα χρησιμοποιηθεί για την αποστολή των παραμέτρων κλήσης της υπηρεσίας,

```
eShopStore.getStoreBasicParametersRequest myRequest = new
eShopStore.getStoreBasicParametersRequest();
```

ένα αντικείμενο τύπου eShopStore.LoginParams που θα χρησιμοποιηθεί για την αποθήκευση των παραμέτρων σύνδεσης του χρήστη,

```
eShopStore.LoginParams myLoginParams = new
eShopStore.LoginParams();
```

αποδίδονται τιμές στις μεταβλητές του myLoginParams,

```
myLoginParams.User = strUser;

myLoginParams.Password = strPass;
```

δημιουργείται ένα αντικείμενο τύπου eShopStore.VirtueMartLogin και αποδίδεται στην μεταβλητή του LoginParams το αντικείμενο που έχει δημιουργηθεί και ενημερωθεί παραπάνω, το myLoginParams

```
eShopStore.VirtueMartLogin myLogin = new
eShopStore.VirtueMartLogin();

myLogin.LoginParams = myLoginParams;
```

Στη συνέχεια ενημερώνεται η μεταβλητή VirtueMartLogin του myRequest με το αντικείμενο myLogin,

```
myRequest.VirtueMartLogin = myLogin;
```

δημιουργείται ένα αντικείμενο πελάτη (client) της υπηρεσίας,

```
eShopStore.VirtueMart_Store_Service srv_Store = new
eShopStore.VirtueMart_Store_ServiceClient("VirtueMart_Store_SOAP")
;
```

δημιουργείται ένα αντικείμενο τύπου eShopStore.getStoreBasicParametersResponse στο οποίο θα αποθηκευτεί η απάντηση της υπηρεσίας.

```
eShopStore.getStoreBasicParametersResponse myResponse = new
eShopStore.getStoreBasicParametersResponse();
```

Τέλος καλείται η μέθοδος getStoreBasicParameters της υπηρεσίας με παραμέτρους την myRequest και αποθηκεύεται το αποτέλεσμά της στην myResponse

```
myResponse = srv_Store.getStoreBasicParameters(myRequest);
```

και επιστρέφεται το αποτέλεσμα.

```
return myResponse;        }
```

7.5 Κλάσεις και μέθοδοι

Σύμφωνα με τον βασικό σχεδιασμό του ενδιάμεσου λογισμικού (κεφάλαιο “Λειτουργικότητα, δομές δεδομένων και βασικός σχεδιασμός”) η λειτουργικότητα χωρίζεται σε τρεις ενότητες:

- Λειτουργίες σύνδεσης και μεταφοράς δεδομένων από και προς το ηλεκτρονικό κατάστημα
- Λειτουργίες σύνδεσης και μεταφοράς δεδομένων από και προς το ERP
- Λειτουργίες διασύνδεση, αντιστοίχιση και γενικές εσωτερικές λειτουργίες του ενδιάμεσου λογισμικού.

Οι τρεις αυτές ενότητες αποτελούν και τις τρεις κλάσεις της εφαρμογής.

7.5.1 Κλάση “eShop”

Η κλάση “eShop” παρέχει μεθόδους για διασύνδεση και μεταφορά δεδομένων από και προς το ηλεκτρονικό κατάστημα. Ακολουθεί η λίστα με τις μεθόδους που υλοποιούνται.

getEShopStore

Ανακτά τα δεδομένα των βασικών πινάκων του ηλεκτρονικού καταστήματος. Τις χώρες, τα νομίσματα, τις κατηγορίες ΦΠΑ, τις καταστάσεις παραγγελίας, τους τρόπους αποστολής και τους τρόπους πληρωμής.

Πίνακας 54: Μέθοδος getEShopStore

Παράμετροι	
Όνομα	Τύπος
User	string

Password	string
Επιστρέφει eShopStore.getStoreBasicParametersResponse	

getEShopCustomers

Ανακτά τους πελάτες του ηλεκτρονικού καταστήματος.

Πίνακας 55: Μέθοδος getEShopCustomers

Παράμετροι	
Όνομα	Τύπος
User	string
Password	string
Επιστρέφει eShopCustomers.getCustomersResponse	

getEShopManufacturers

Ανακτά τους κατασκευαστές ειδών του ηλεκτρονικού καταστήματος

Πίνακας 56: Μέθοδος getEShopManufacturers

Παράμετροι	
Όνομα	Τύπος
User	string
Password	string
Επιστρέφει eShopProducts.getManufacturersResponse	

setEShopManufacturers

Ενημερώνει του κατασκευαστές ειδών του ηλεκτρονικού καταστήματος

Πίνακας 57: Μέθοδος setEShopManufacturers

Παράμετροι	
Όνομα	Τύπος
User	string
Password	string
strModType	string
ManufacturersToModify	eShopProducts.Manufacturer[]
Επιστρέφει	
eShopProducts.modifyManufacturersResponse	

getEShopCategories

Ανακτά τις κατηγορίες ειδών του ηλεκτρονικού καταστήματος

Πίνακας 58: Μέθοδος getEShopCategories

Παράμετροι	
Όνομα	Τύπος
User	string
Password	string
Επιστρέφει	
eShopProducts.getCategoriesResponse	

setEShopCategories

Ενημερώνει τις κατηγορίες ειδών του ηλεκτρονικού καταστήματος

Πίνακας 59: Μέθοδος setEShopCategories

Παράμετροι	
Όνομα	Τύπος
User	string
Password	string
strModType	string
CategoriesToModify	eShopProducts.Category []
Επιστρέφει	
eShopProducts.modifyManufacturersResponse	

getEShopItems

Ανακτά τα είδη του ηλεκτρονικού καταστήματος

Πίνακας 60: Μέθοδος getEShopItems

Παράμετροι	
Όνομα	Τύπος
User	string
Password	string
Επιστρέφει	
eShopProducts.getItemsResponse	

setEShopItems

Ενημερώνει τα είδη του ηλεκτρονικού καταστήματος

Πίνακας 61: Μέθοδος setEShopItems

Παράμετροι	
Όνομα	Τύπος
User	string

Password	string
strModType	string
ItemsToModify	eShopProducts. Item
Επιστρέφει	
eShopProducts.modifyItemsResponse	

setEShopOrderStatus

Ενημερώνει την κατάσταση παραγγελιών του ηλεκτρονικού καταστήματος

Πίνακας 62: Μέθοδος setEShopOrderStatus

Παράμετροι	
Όνομα	Τύπος
User	string
Password	string
OrderStatus	eShopOrders.OrderStatus
Επιστρέφει	
eShopProducts.modifyItemsResponse	

getEShopOrders

Ανακτά τις παραγγελίες του ηλεκτρονικού καταστήματος

Πίνακας 63: Μέθοδος getEShopOrders

Παράμετροι	
Όνομα	Τύπος
User	string
Password	string
Επιστρέφει	
eShopOrders.getOrdersResponse	

getEShopOrders

Ανακτά τις παραγγελίες του ηλεκτρονικού καταστήματος που είναι μεταγενέστερες από την παραγγελία της οποίας το id δίνεται στην παράμετρο strOrder_Id.

Πίνακας 64: Μέθοδος getEShopOrders

Παράμετροι	
Όνομα	Τύπος
User	string
Password	string
strOrder_Id	string
Επιστρέφει eShopOrders.getOrdersResponse	

7.5.2 Κλάση "ERP"

Η κλάση "EPR" παρέχει μεθόδους για διασύνδεση και μεταφορά δεδομένων από και προς το Soft1 ERP. Για να επιτευχθεί επικοινωνία με το Soft1 ERP χρησιμοποιούμε τις βιβλιοθήκες που παρέχονται από την εταιρεία Softone. Για να μπορέσουν να χρησιμοποιηθούν οι εξωτερικής βιβλιοθήκες στο Visual Studio 2008 πρέπει να προστεθούν οι αντίστοιχες αναφορές (references) σε αυτές (στον Solution Explorer), οι οποίες στην συνέχεια συμπεριλαμβάνονται με την χρήση της εντολής "using".

```
using SoftOne.Interop;
using SoftOne.System;
```

Ακολουθεί η λίστα με τις μεθόδους που υλοποιούνται.

getTableId

Χρησιμοποιείται για την εύρεση της τιμής του κύριου κλειδιού ενός πίνακα, λαμβάνει ως παραμέτρους τα επιμέρους τμήματα μια SQL εντολής και επιστρέφει το αποτέλεσμα.

Πίνακας 65: Μέθοδος getTableId

Παράμετροι	
Όνομα	Τύπος
myErp	S1Prg
strSelect	string
strFrom	string
strWhere	string
Επιστρέφει	
int	

getCountries

Ανακτά τις χώρες από το ERP.

Πίνακας 66: Μέθοδος getCountries

Παράμετροι	
Όνομα	Τύπος
myErp	S1Prg
Επιστρέφει	
DataTable	

getCurrencies

Ανακτά τα νομίσματα από το ERP.

Πίνακας 67: Μέθοδος getCurrencies

Παράμετροι	
Όνομα	Τύπος
myErp	S1Prg
Επιστρέφει	
DataTable	

getVats

Ανακτά τις κατηγορίες ΦΠΑ από το ERP.

Πίνακας 68: Μέθοδος getVats

Παράμετροι	
Όνομα	Τύπος
myErp	S1Prg
Επιστρέφει	
DataTable	

getStatus

Ανακτά τις καταστάσεις παραγγελιών από το ERP.

Πίνακας 69: Μέθοδος getStatus

Παράμετροι	
Όνομα	Τύπος
myErp	S1Prg
intCompany	int
Επιστρέφει	
DataTable	

getShipment

Ανακτά τους τρόπους αποστολής από το ERP.

Πίνακας 70: Μέθοδος getShipment

Παράμετροι	
Όνομα	Τύπος
myErp	S1Prg
intCompany	int
Επιστρέφει	
DataTable	

getPayments

Ανακτά τους τρόπους πληρωμής από το ERP.

Πίνακας 71: Μέθοδος getPayments

Παράμετροι	
Όνομα	Τύπος
myErp	S1Prg
intCompany	int
Επιστρέφει	
DataTable	

getCustomers

Ανακτά τους πελάτες από το ERP που έχουν γίνει εισαγωγή από το ηλεκτρονικό κατάστημα.

Πίνακας 72: Μέθοδος getCustomers

Παράμετροι

Όνομα	Τύπος
myErp	S1Prg
intCompany	int
Επιστρέφει	
DataTable	

setCustomers

Καταχωρεί στο ERP τους νέους πελάτες του ηλεκτρονικού καταστήματος και ενημερώνει τα στοιχεία των πελατών που έχουν ήδη καταχωρηθεί με τυχόν αλλαγές.

Πίνακας 73: Μέθοδος setCustomers

Παράμετροι	
Όνομα	Τύπος
myErp	S1Prg
intCompany	int
dtbCustomers	DataTable
strType	string
Επιστρέφει	
bool	

getManufacturers

Ανακτά από το ERP τους κατασκευαστές ειδών που θα ενημερώσουν το ηλεκτρονικό κατάστημα.

Πίνακας 74: Μέθοδος getManufacturers

Παράμετροι	
Όνομα	Τύπος

myErp	S1Prg
intCompany	int
Επιστρέφει	
DataTable	

getCategories

Ανακτά από το ERP τις κατηγορίες ειδών που θα ενημερώσουν το ηλεκτρονικό κατάστημα.

Πίνακας 75: Μέθοδος getCategories

Παράμετροι	
Όνομα	Τύπος
myErp	S1Prg
intCompany	int
Επιστρέφει	
DataTable	

getItems

Ανακτά από το ERP τα είδη που θα ενημερώσουν το ηλεκτρονικό κατάστημα.

Πίνακας 76: Μέθοδος getItems

Παράμετροι	
Όνομα	Τύπος
myErp	S1Prg
intCompany	int
Επιστρέφει	
DataTable	

setItems

Ενημερώνει τα είδη του ERP με Item_Id που έχουν λάβει στο ηλεκτρονικό κατάστημα.

Πίνακας 77: Μέθοδος setItems

Παράμετροι	
Όνομα	Τύπος
myErp	S1Prg
intCompany	int
Item_Code	string
Item_Id	string
Επιστρέφει	
bool	

getOrderStatus

Ανακτά από το ERP τις καταστάσεις των καταχωρημένων παραγγελιών που έχουν γίνει εισαγωγή από το ηλεκτρονικό κατάστημα

Παράμετροι	
Όνομα	Τύπος
myErp	S1Prg
intCompany	int
Επιστρέφει	
DataTable	

getLastOrder

Ανακτά από το ERP το id της τελευταίας παραγγελίας του ηλεκτρονικού καταστήματος που έχει καταχωρηθεί στο σύστημα.

Πίνακας 78: Μέθοδος getLastOrder

Παράμετροι	
Όνομα	Τύπος
myErp	S1Prg
intCompany	int
Επιστρέφει	
string	

setOrders

Καταχωρεί στο ERP τις νέες παραγγελίες του ηλεκτρονικού καταστήματος

Πίνακας 79: Μέθοδος setOrders

Παράμετροι	
Όνομα	Τύπος
myErp	S1Prg
intCompany	int
dtbHeader	DataTable
dtbItems	DataTable
dtbBilling	DataTable
dtbShipping	DataTable
dtbStatus	DataTable
Επιστρέφει	
bool	

7.5.3 Κλάση "Main" - Μέθοδοι διαχείρισης του αρχείου παραμέτρων

Η κλάση "Main" υλοποιεί μεθόδους για την διαχείριση του αρχείου παραμέτρων της εφαρμογής, και υλοποιεί λειτουργίες διασύνδεσης μεταξύ του ERP και του

ηλεκτρονικού καταστήματος καλώντας τις κατάλληλες μεθόδους από τις κλάσεις “ERP” και “eShop” ανάλογα με τα συμβάντα (events) που προέρχονται από την διεπαφή χρήστη. Ακολουθεί η λίστα των μεθόδων της χωρισμένες σε ενότητες.

getIniValue

Επιστρέφει την/τις τιμή/ες της παραμέτρου strValue που βρίσκονται στο αρχείο που έχει ανοιχθεί με τον mysr. Υπάρχουν τρεις τύποι αναζήτησης που καθορίζονται από την τιμή της παραμέτρου strType, ο “WHOLE_INI” που επιστρέφει όλες τις τιμές της παραμέτρου διαχωρισμένες με τον χαρακτήρα “#”, ο “EXCEPT_LINE” που επιστρέφει όλες τις γραμμές του αρχείου εκτός μιας συγκεκριμένης γραμμής που καθορίζεται από την τιμή της παραμέτρου strValue, και ο “=” που επιστρέφει την τιμή μιας συγκεκριμένης παραμέτρου.

Πίνακας 80: Μέθοδος getIniValue

Παράμετροι	
Όνομα	Τύπος
mysr	StreamReader
strType	string
strValue	string
Επιστρέφει	
string	

getIniMappingValues

Ενημερώνει τον DataTable myMappings με τις αντιστοιχισμένες τιμές της παραμέτρου strIniValuesType.

Πίνακας 81: Μέθοδος getIniMappingValues

Παράμετροι	
Όνομα	Τύπος

strIniValuesType	string

readFromIni

Διαβάζει το αρχείο παραμέτρων της εφαρμογής καλώντας την getIniValue() και ενημερώνει τα πεδία παραμέτρων της καρτέλας “Παράμετροι Σύνδεσης” με τις αποθηκευμένες τιμές.

writeToIni

Προσθέτει στο αρχείο παραμέτρων της εφαρμογής μια γραμμή με την τιμή της strValue.

Πίνακας 82: Μέθοδος writeToIni

Παράμετροι	
Όνομα	Τύπος
strValue	string

deleteFromIni

Διαγράφει από το αρχείο παραμέτρων της εφαρμογής μια γραμμή με την τιμή της strValue.

Πίνακας 83: Μέθοδος deleteFromIni

Παράμετροι	
Όνομα	Τύπος
strValue	string

updateMappingsDataTable

Ενημερώνει τον DataTable myMappings με όλες τις αντιστοιχισμένες τιμές καλώντας πολλές φορές την getIniMappingValues().

checkForMapping

Αναζητεί στον DataTable myMappings την παράμετρο που καθορίζεται από την τιμή της strIniValueType από το ERP ή το ηλεκτρονικό κατάστημα, ανάλογα με την τιμή της strFrom, με τιμή ίση της strValue και επιστρέφει την αντιστοιχισμένη τιμή.

Πίνακας 84: Μέθοδος checkForMapping

Παράμετροι	
Όνομα	Τύπος
strIniValueType	string
strFrom	string
strValue	string
Επιστρέφει	
string	

displayMappings

Αναζητεί στον DataTable myMappings τις αντιστοιχίσεις της τρέχουσας επιλογής (την τιμή της παραμέτρου strMappingType) και τις εμφανίζει στο ListBox των αντιστοιχίσεων.

Πίνακας 85: Μέθοδος displayMappings

Παράμετροι	
Όνομα	Τύπος
strMappingType	string

7.5.4 Κλάση "Main" - Μέθοδοι συμβάντων σύνδεσης στο ERP και της καρτέλας "Παράμετροι Σύνδεσης"

private void btnLoginToErp_Click(object sender, EventArgs e)

Χειρίζεται το συμβάν του πατήματος του Button "Login to ERP".

Υλοποιεί την σύνδεση με το ERP και ενημερώνει το αρχείο παραμέτρων της εφαρμογής με τις τιμές των παραμέτρων σύνδεσης

private void btnBrowseXCO_Click(object sender, EventArgs e)

Χειρίζεται το συμβάν του πατήματος του Button "..." δεξιά από το πεδίο "Αρχείο XCO".

Εμφανίζει ένα παράθυρο διαλόγου για την επιλογή του αρχείο XCO, και την αποθήκευση της διαδρομής στο πεδίο "Αρχείο XCO".

7.5.5 Κλάση "Main" - Μέθοδοι συμβάντων καρτέλας "Αντιστοιχίσεις"

private void btnGetData_Click(object sender, EventArgs e)

Χειρίζεται το συμβάν του πατήματος του Button "Ενημέρωση" που βρίσκεται στο κάτω αριστερό μέρος της καρτέλας.

- Καλεί την myEShop.getEShopStore() και εμφανίζει τις τιμές στο κάτω DataGridView ανάλογα με την "Επιλογή" που είναι ενεργεί.
- Καλεί μια από τις myERP.getCountries(), myERP.getCurrencies(), myERP.getVats(), myERP.getManufacturers(), myERP.getCategories(), myERP.getStatus(), myERP.getShipment(), myERP.getPayments() ανάλογα με την "Επιλογή" που είναι ενεργεί και εμφανίζει τις τιμές στο πάνω DataGridView.

- Τέλος καλεί την displayMappings() για την ενημέρωση του ListBox των αντιστοιχίσεων στο δεξιό μέρος της καρτέλας.

private void btnMap_Click(object sender, EventArgs e)

Χειρίζεται το συμβάν του πατήματος του Button “<=>” που βρίσκεται ανάμεσα στα δύο DataGridViews.

Προσθέτει στο ListBox των αντιστοιχίσεων τις επιλεγμένες τιμές.

private void btnUnmap_Click(object sender, EventArgs e)

Χειρίζεται το συμβάν του πατήματος του Button “Ακύρωση αντιστοίχισης”.

Αφαιρεί από το ListBox των αντιστοιχίσεων την επιλεγμένη αντιστοίχηση.

private void btnSaveMappings_Click(object sender, EventArgs e)

Χειρίζεται το συμβάν του πατήματος του Button “Αποθήκευση”.

- Καλεί την deleteFromIni() για να διαγραφούν από το αρχείο παραμέτρων όλες οι τιμές αντιστοίχησης της τρέχουσας επιλογής.
- Καλεί την writeToIni() για να γράψει στο αρχείο παραμέτρων τις νέες αντιστοιχήσεις.
- Καλεί την updateMappingsDataTable() για να ενημερωθεί ο πίνακας των αντιστοιχίσεων.
- Εμφανίζει μήνυμα στο χρήστη ότι η εργασία ολοκληρώθηκε.

7.5.6 Κλάση “Main” - Μέθοδοι συμβάντων καρτέλας “Λειτουργίες”

private void btnGetCustomers_Click(object sender, EventArgs e)

Χειρίζεται το συμβάν του πατήματος του Button “Καταχώρηση Πελατών”.

- Καλεί την myERP.getCustomers() για να ανακτήσει τους πελάτες που είναι ήδη καταχωρημένοι στο ERP.
- Καλεί την myEShop.getEShopCustomers() για να ανακτήσει τους πελάτες του ηλεκτρονικού καταστήματος.
- Πραγματοποιεί συγκρίσεις για τον καθορισμό των νέων πελατών και πραγματοποιεί έλεγχο των τιμών αντιστοιχίσεων για την χώρα του πελάτη με την κλήση της checkForMapping().
- Εν τέλη καλεί δύο φορές την myERP.setCustomers() μια φορά για εισαγωγή των νέων πελατών και μια φορά για ενημέρωση των ήδη υπαρχόντων.

private void btnSetManufacturers_Click(object sender, EventArgs e)

Χειρίζεται το συμβάν του πατήματος του Button “Ενημέρωση Κατασκευαστών”.

- Καλεί την myERP.getManufacturers() για να ανακτήσει τους κατασκευαστές από το ERP.
- Καλεί την myEShop.getEShopManufacturers() για να ανακτήσει τους κατασκευαστές του ηλεκτρονικού καταστήματος.
- Πραγματοποιεί συγκρίσεις για τον καθορισμό των νέων κατασκευαστών.
- Εν τέλη καλεί δύο φορές την myERP.setEShopManufacturers() μια φορά για εισαγωγή των νέων κατασκευαστών και μια φορά για ενημέρωση των ήδη υπαρχόντων.

private void btnSetCategories_Click(object sender, EventArgs e)

- Χειρίζεται το συμβάν του πατήματος του Button “Ενημέρωση Κατηγοριών”.
- Καλεί την myERP.getCategories() για να ανακτήσει τις κατηγορίες από το ERP.

- Καλεί την myEShop.getEShopCategories() για να ανακτήσει τις κατηγορίες του ηλεκτρονικού καταστήματος.
- Πραγματοποιεί συγκρίσεις για τον καθορισμό των νέων κατηγοριών.
- Εν τέλη καλεί δύο φορές την myERP.setEShopCategories() μια φορά για εισαγωγή των νέων κατηγοριών και μια φορά για ενημέρωση των ήδη υπαρχόντων.

private void btnSetItems_Click(object sender, EventArgs e)

Χειρίζεται το συμβάν του πατήματος του Button “Ενημέρωση Ειδών”.

- Καλεί την myERP.getItems() για να ανακτήσει των ειδών από το ERP.
- Καλεί την myEShop.getEShopItems() για να ανακτήσει των ειδών του ηλεκτρονικού καταστήματος.
- Πραγματοποιεί συγκρίσεις για τον καθορισμό των νέων ειδών.
- Πραγματοποιεί έλεγχο των τιμών αντιστοιχίσεων για το νόμισμα, την κατηγορία ΦΠΑ, την κατηγορία και τον κατασκευαστή του είδους με επαναλαμβανόμενη κλήση της checkForMapping().
- Έπειτα καλεί δύο φορές την myERP.setEShopItems() μια φορά για εισαγωγή των νέων ειδών και μια φορά για ενημέρωση των ήδη υπαρχόντων.
- Για τα νέα είδη καλεί επιπλέον την myERP.setItems() για να ενημερωθούν τα είδη του ERP με το Product_Id των ειδών του ηλεκτρονικού καταστήματος.

private void btnSetOrderStatus_Click(object sender, EventArgs e)

Χειρίζεται το συμβάν του πατήματος του Button “Ενημέρωση Καταστάσεων Παραγγελιών”.

- Καλεί την myERP.getOrderStatus() για να ανακτήσει τις καταστάσεις των καταχωρημένων παραγγελιών από το ERP.
- Για κάθε παραγγελία πραγματοποιεί έλεγχο της τιμής αντιστοίχησης της κατάστασης καλώντας την checkForMapping(), και ενημερώνει το ηλεκτρονικό κατάστημα με κλήση της myEShop.setEShopOrderStatus().

private void btnGetOrders_Click(object sender, EventArgs e)

Χειρίζεται το συμβάν του πατήματος του Button “Καταχώρηση Παραγγελιών”.

- Καλεί την myERP.getLastOrder() για την ανάκτηση της τελευταίας παραγγελίας που έχει καταχωρηθεί στο ERP.
- Καλεί την myEShop.getEShopOrders() με παράμετρο το αποτέλεσμα της myERP.getLastOrder() για ανάκτηση των μεταγενέστερων παραγγελιών
- Πραγματοποιεί έλεγχο των τιμών αντιστοίχησης για το νόμισμα, τον τρόπο αποστολής, τον τρόπο πληρωμής και την κατάσταση παραγγελίας καλώντας επαναλαμβανόμενα την checkForMapping().
- Καλεί την myERP.setOrders() για καταχώρηση των νέων παραγγελιών στο ERP.

ΕΠΙΛΟΓΟΣ

Σε αυτό το κεφάλαιο παρουσιάστηκε η διεπαφή χρήστη, οι κλάσεις και μέθοδοι της εφαρμογής ενδιάμεσου λογισμικού που δημιουργήθηκε. για διασύνδεση του ERP με το ηλεκτρονικό κατάστημα. Επιπλέον παρουσιάστηκε ο τρόπος σύνδεσης σε υπηρεσίες ιστού μέσα από το Visual Studio 2008 και την C#.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην παρούσα πτυχιακή εργασία δημιουργήθηκε μια εφαρμογή ενδιάμεσου λογισμικού (middleware) για διασύνδεση δύο ετερογενών συστημάτων χρησιμοποιώντας την μηνυματοστραφή προσέγγιση της τεχνολογίας των υπηρεσιών ιστού. Τα δύο συστήματα αποτελούνται από το ηλεκτρονικό κατάστημα VirtueMart που είναι υλοποιημένο σε PHP και χρησιμοποιεί ως DBMS την MySql, και το Soft1 ERP με χρήση .NET και DBMS τον Microsoft SQL Server. Παρουσιάστηκαν τα βήματα που ακολουθούνται για την κατασκευή υπηρεσιών ιστού σε PHP και ο τρόπος κατανάλωσής τους με C# .NET.

Ακόμη αναλύθηκαν θεωρητικά η υπηρεσιοστρεφής αρχιτεκτονική και οι υπηρεσίες ιστού. Παρουσιάστηκε ο τρόπος με τον οποίο ανταποκρίνονται στις προκλήσεις που προκύπτουν από την ανάγκη ανάπτυξης ευέλικτων και επεκτάσιμων πληροφοριακών συστημάτων στον σύγχρονο επιχειρησιακό περιβάλλον. Καθώς επίσης και οι λόγοι για τους οποίους θεωρούνται ως η εξέλιξη των υπαρχόντων μεθοδολογιών κατασκευής δομημένου λογισμικού.

Συμπερασματικά, σε έναν συνεχώς μεταβαλλόμενο και άκρως ανταγωνιστικό επιχειρησιακό περιβάλλον οι υπηρεσιοστρεφής αρχιτεκτονική και οι υπηρεσίες ιστού προσφέρουν ολόκληρο το αρχιτεκτονικό και τεχνολογικό υπόβαθρο που είναι αναγκαίο για την δημιουργία ευέλικτων και επεκτάσιμων πληροφοριακών συστημάτων. Παρέχοντας το συγκριτικό πλεονέκτημα έναντι άλλων τεχνολογιών, του μειωμένου κόστους και χρόνου παραγωγής και της εκμετάλλευσης υπαρχόντων πληροφοριακών συστημάτων.

ΑΝΑΦΟΡΕΣ

[How are passwords encypted 2007], http://forum.joomla.org/viewtopic.php?f=431&t=195712

[Joomla 2010] http://www.joomla.org/

[Papazoglou 2008], Michael Papazoglou (2008), Web Services: Principles and Technology, Pearson Education Limited, Essex, England

[Richards 2006], Robert Richards (2006), Pro PHP XML and Web Services, Apress

[SOA Principles, 2010] http://soaprinciples.com/

[Softone 2010] http://www2.softone.gr/

[Softone Developers Network 2010] https://groups.google.com/group/soft1?hl=el

[VirtueMart 2010] http://virtuemart.net/

[Web Services Description Language (WSDL)] http://www.w3.org/TR/wsdl

[Weerawarana 2005], Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, Fonald F. Ferguson (2005), Web Services Platform Architecture, Pearson Education Inc, Upper Saddle River, US

ΒΙΒΛΙΟΓΡΑΦΙΑ

Creating and Accessing Web Services (Visual C#) (2010), http://msdn.microsoft.com/en-us/library/ms228289(VS.80).aspx

How are passwords encypted (2007), http://forum.joomla.org/viewtopic.php?f=431&t=195712

VirtueMart_1.1.2_eCommerce_Bundle_Joomla_1.5.9 (2010), http://www.codes-libres.org/apps/VirtueMart_1.1.2_eCommerce_Bundle_Joomla_1.5.9/nav.html

How to access a Web service from a C# Desktop Application (2010),

http://andres.jaimes.net/2010/02/09/how-to-access-a-web-service-from-a-c-desktop-application/

Michael Papazoglou (2008), Web Services: Principles and Technology, Pearson Education Limited, Essex, England

Paul C. Brown (2008), Implementing SOA: Total Architecture in Practice, Addison Wesley Professional

PHP and SOAP (2010), http://php.net/manual/en/book.soap.php

PHP SOAP Extension (2010), http://devzone.zend.com/article/689

Robert Richards (2006), Pro PHP XML and Web Services, Apress

Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, Fonald F. Ferguson (2005), Web Services Platform Architecture, Pearson Education Inc, Upper Saddle River, US

Service-oriented architecture (2010), http://en.wikipedia.org/wiki/Service-oriented_architecture

Service-oriented architecture (SOA) (2010), http://www.service-architecture.com

SOAP Principles (2010), http://soaprinciples.com/

Softone Developers Network (2010), https://groups.google.com/group/soft1?hl=el

VirtueMart (2010), http://virtuemart.net/

VirtueMart 1.1 Developer Manual (2008),

http://virtuemart.net/documentation/Developer_Manual/

VirtueMart 1.1 User Manual (2009), http://virtuemart.net/documentation/User_Manual/

What is service-oriented architecture? (2010), http://www.javaworld.com/javaworld/jw-06-2005/jw-0613-soa.html

What is SOA (2010), http://www.whatissoa.com

Yu, Liyang.(2007), Introduction to Semantic Web and Semantic Web services, Chapman & Hall/CRC, Taylor & Francis Group, US