ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΜΣ ΗΛΕΚΤΡΟΝΙΚΗ ΜΑΘΗΣΗ

Ανάλυση Σχεδιασμός και Ανάπτυξη ενός e-Service

ΑΛΥΣΑΝΔΡΑΤΟΥ ΕΛΕΥΘΕΡΙΑ

ΜΕ/08003

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΜΑΡΙΝΟΣ ΘΕΜΙΣΤΟΚΛΕΟΥΣ

Ευχαριστώ τον επιβλέποντα καθηγητή μου κ.Μαρίνο Θεμιστοκλέους για την βοήθεια, τη στήριξη και την υπομονή του.

Περιεχόμενα

Ευρετήριο Εικόνων

Ευρετήριο Πινάκων

Κεφάλαιο 1. Εισαγωγή

1.1 Ορισμός Προβλήματος

Σαν γενική περιγραφή του προβλήματος μπορούμε να θεωρήσουμε την προσπάθεια συγχρονισμού δύο βάσεων δεδομένων. Ειδικότερα, η παρούσα εργασία στο ένα σκέλος της αφορά στον συγχρονισμό της βάσης δεδομένων ενός συστήματος παραγγελιοληψίας μιας αλυσίδας καταστημάτων εστίασης με αυτήν του e-shop της ίδιας αλυσίδας. Η επιχείρηση διατηρεί δύο βάσεις δεδομένων, μια για το e-shop και μια για τα ≪φυσικά≫ καταστήματα. Κάτω από αυτές τις συνθήκες όταν εκτελείται μια παραγγελία μέσω Internet γίνεται χρήση της βάσης δεδομένων του ηλεκτρονικού καταστήματος ενώ αντίστοιχα όταν ο πελάτης επισκέπτεται τα ≪φυσικά≫ κατάστηματα η αγορά γίνεται μέσω της βάσης δεδομένων του καταστήματος. Γίνεται, επομένως, αντιληπτό ότι αν δεν ενημερώνονται και οι δύο βάσεις σε κάθε δοσοληψία, είτε γίνεται μέσω καταστήματος είτε γίνεται μέσω Internet, το αποτέλεσμα θα είναι η κάθε μια να έχει τα δικά της δεδομένα που σε καμία περίπτωση δεν θα αντιστοιχούν στα πραγματικά. Με τη κατασκευή του ενός web service που περιγράφεται στη συγκεκριμένη εργασία επιτυγχάνεται ο συγχρονισμός των δυο βάσεων ώστε να υπάρχει ενοποίηση των πωλήσεων αλλά και δυνατότητα μοναδικής ταυτοποίησης και εξυπηρέτησης του εκάστοτε πελάτη.

Στο άλλο σκέλος της η εργασία αυτή καταπιάνεται με το πώς μια εταιρία εστίασης στα κεντρικά της μπορεί να ενημερώνεται για τις κινήσεις και τη λειτουργία των υποκαταστημάτων της. Η δημιουργία ενός web service που θα καλείται για να δώσει πληροφορίες όπως οι τζιροι των υποκαταστημάτων, οι ωριαίες πωλήσεις τους, ποια καταστήματα και ποια σημεία πωλησης δουλεύουν ανα πάσα στιγμή καθώς επίσης να ορίζεται από τα κεντρικά η αλλαγή καταλόγων και τιμών για καλύτερο έλεγχο και διαχείριση των υποκαταστημάτων.

Και τα δύο προαναφερθέντα προβλήματα υλοποιούνται με web services διότι υπάρχει ασυμβατότητα μεταξύ των εφαρμογών που πρόκειται να επικοινωνήσουν.

1.2 Αντικείμενο Εργασίας

Η εργασία αυτή χωρίζεται σε δύο διαφορετικά κομμάτια, το θεωρητικό και το πρακτικό. Το θεωρητικό κομμάτι έχει σαν σκοπό να μας εισάγει:

- στην σημασία των Web Services (Διαδικτυακές Υπηρεσίες)
- στα πλεονεκτήματα τους
- στην αρχιτεκτονική τους
- στις τεχνολογίες που χρησιμοποιούν.

Επίσης, στο θεωρητικό κομμάτι αναλύονται τα εργαλεία που απαιτούνται για την υλοποίηση ενός Web Service.

Στο πρακτικό κομμάτι γίνεται υλοποίηση δύο Web Services. Το πρώτο ονομάζεται InternetDeliveryService και εξυπηρετεί τη διασύνδεση διαφορετικών συστημάτων με ένα κεντρικό σύστημα παραγγελιοληψίας. Σκοπός αυτής της διασύνδεσης είναι τα προαναφερθέντα συστήματα να μπορούν να:

- να καταχωρούν παραγγελίες στην κεντρική βάση δεδομένων του συστήματος
- να έχουν πρόσβαση στα στοιχεία καταχωρημένων πελατών όπως κατάστημα εξυπηρέτησης πελάτη, τιμοκατάλογος πελάτη , στοιχεία διεύθυνσης και λοιπά στοιχεία
- να δέχονται παραγγελίες μόνο για τα ανοιχτά καταστήματα και να μπορούν να ενημερώσουν τον πελάτη τους για τον εκάστοτε χρόνο εξυπηρέτησης του
- να ενημερώνονται για την εκάστοτε διαθεσιμότητα προϊόντων
- να γνωστοποιούν στον πελάτη όταν αυτός το επιθυμεί το στάδιο που βρίσκεται η παραγγελία του

Όλες οι παραπάνω λειτουργίες γίνονται On demand σε πραγματικό χρόνο και αποτελούν διαφορετικές κλήσεις του Web Service InternetDeliveryService.

Το δεύτερο web service (Storemanager) μεσολαβεί αναμεσα στο διαχειριστικό εργαλείο μιας εταιρίας και στα επιμέρους συστήματα παραγγελιοληψίας των υποκαταστημάτων της. Σκοπός αυτής της διαμεσολάβησης είναι:

- να ενημερώνονται τα κεντρικά για τις πωλήσεις των υποκαταστημάτων είτε συνολικά όλης της μέρας είτε ανα ώρα
- να μπορούν μόνο τα κεντρικά να υλοποιήσουν αλλαγές στη δομή του καταλόγου αλλά και στις τιμές των προϊόντων των υποκαταστημάτων
- να γνωρίζουν τα κεντρικά ανά πάσα στιγμή πόσα και ποια υποκαταστήματα είναι ανοικτά

Όλες οι παραπάνω λειτουργίες γίνονται On demand σε πραγματικό χρόνο και αποτελούν διαφορετικές κλήσεις του Web Service Storemanager.

1.3 Ανάλυση Τόμου

Ο τόμος της παρούσας εργασίας έχει οργανωθεί ως εξής στα παρακάτω κεφάλαια:

- Το Κεφάλαιο 1 είναι καθαρά εισαγωγικό και πραγματεύεται το σκοπό της παρούσας εργασίας καθώς και δομείται
- Το Κεφάλαιο 2 παρουσιάζει την έννοια των Διαδικτυακών Υπηρεσιών , την αρχιτεκτονική τους και τα πλεονεκτήματα τους, τις τεχνολογίες που χρησιμοποιούν οι Διαδικτυακές Υπηρεσίες.
- Το Κεφάλαιο 3 αναλύει τα εργαλεία που θα χρησιμοποιηθούν στο επόμενο κεφάλαιο για να υλοποιηθεί το Web Service InternetDeliveryService.
- Το Κεφάλαιο 4 περιέχει το σχεδιασμό, την ανάλυση και την υλοποίηση των Web Services. Συγκεκριμένα περιλαμβάνει τη δομή και την αρχιτεκτονική της βάσης δεδομένων, τις κλήσεις που μπορούμε να πραγματοποιήσουμε σε αυτά τα Web Services και σχετικά παραδείγματα.
- Το Κεφάλαιο 5 αποτελεί τον επίλογο της εργασίας καθώς αναφέρονται και πιθανές μελλοντικές επεκτάσεις της εργασίας αυτής.
- Στο Παράρτημα φαίνονται οι οθόνες εξωτερικών προγραμμάτων που καλούν τα Web Services.

Κεφάλαιο 2. Βιβλιογραφική Αναφορά

2.1 Εισαγωγή

Στη παρούσα ενότητα της εργασίας γίνεται μια βιβλιογραφική αναφορά στα Web Services και αναλυτική παρουσίαση των πλεονεκτημάτων τους. Παρουσιάζονται επίσης εργαλεία για τη δημιουργία Web Services, πρωτόκολλα και αρχιτεκτονικές που προκύπτουν.

2.2 Τι είναι Web Services

Τα Web Services (Υπηρεσίες Διαδικτύου) είναι μια καινοτομική αρχιτεκτονική με την οποία παρέχεται η δυνατότητα δημιουργίας και χρήσης ηλεκτρονικών υπηρεσιών στο διαδίκτυο με απλό και οικονομικό τρόπο. Μέχρι πρόσφατα η δημιουργία και η παροχή υπηρεσιών από επιχειρήσεις στο Internet γίνονταν με ακαθόριστο τρόπο ο οποίος διέφερε από επιχείρηση σε επιχείρηση. Έτσι, ενώ υπήρχε ένα αρκετά μεγάλο σύνολο από παρεχόμενες υπηρεσίες στο Internet, για να μπορούσε κάποιος να τις χρησιμοποιήσει θα έπρεπε για κάθε μία υπηρεσία να μελετήσει τον τρόπο με τον οποίο θα την καλέσει, να ελέγξει αν χρησιμοποιούν το ίδιο πρωτόκολλο επικοινωνίας (TCP/IP, Http, κλπ) και γενικά να προσαρμόσει όλο το σύστημά του έτσι ώστε να γίνει συμβατό με αυτό του παροχέα της υπηρεσίας.

Για παράδειγμα ας υποθέσουμε ότι κάποια επιχείρηση ενδιαφερόταν να χρησιμοποιήσει μία υποτιθέμενη υπηρεσία που παρείχε το Εθνικό Κέντρο Βιβλίου και η οποία παρουσίαζε όλες τις συνοδευτικές πληροφορίες (τίτλο, εκδοτικό οίκο, τιμή κλπ) για κάποιο βιβλίο δοθέντος του κωδικού του (ISBN).

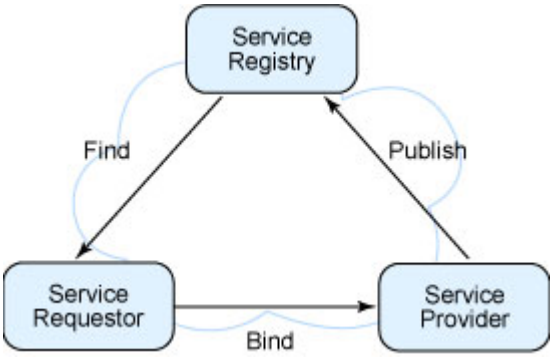
Σε αυτή την περίπτωση ο προγραμματιστής της επιχείρησης θα έπρεπε στην ουσία να δημιουργήσει ένα σύστημα συμβατό με αυτό του Εθνικού Κέντρου Βιβλίου και ως προς το πρωτόκολλο επικοινωνίας αλλά και ως προς τον τρόπο κλήσης των ερωτημάτων και κατόπιν να το προσαρμόσει στις ανάγκες του συστήματος της επιχείρησης. Πολλές φορές αυτό ήταν πολύ δύσκολο – αν όχι ακατόρθωτο – και ακόμα περισσότερες φορές οι επιχειρήσεις σχεδίαζαν τα συστήματά τους έτσι ώστε να αποφεύγουν τέτοιου είδους συνεργασίες με ξένες πηγές για λόγους πολυπλοκότητας και γενικότερα για λόγους κόστους.

Τα πράγματα όμως τα τελευταία χρόνια φαίνεται να παίρνουν διαφορετική τροπή αφού πλέον σχεδόν όλες οι επιχειρήσεις που δημιουργούν υπηρεσίες στο Internet βασίζονται σε μία κοινή αρχιτεκτονική ανάπτυξης, δημοσίευσης και εκμετάλλευσης των υπηρεσιών τους, όπως αυτή καθορίζεται από το W3C και που ορίζεται ως η αρχιτεκτονική των web services.

Μια διαδικτυακή υπηρεσία αποτελείται από πολλές συσχετιζόμενες τεχνολογίες που τοποθετούνται σε διαφορετικό επίπεδο. Ξεκινώντας από κάτω προς τα πάνω, αναφέρουμε τα πρότυπα που χρησιμοποιούνται και τα αναλύουμε εκτενέστερα μαζί με τα θέματα ασφάλειας και διαχείρισης στις επόμενες ενότητες. Αρχικά, απαιτείται ένα πρωτόκολλο για μεταφορά πληροφοριών μέσω δικτύου, όπως το http (HyperText Transfer Protocol), το SMTP (Simple Mail Transport Protocol) ή το FTP (File Transfer Protocol). Κάθε κλήση και απόκριση της υπηρεσίας θα πρέπει να «συσκευαστεί» σε ένα μήνυμα SOAP (Simple Object Access Protocol), το οποίο μπορεί να υφίσταται επεξεργασία από κάποιες επεκτάσεις SOAP (SOAP Extensions) πριν αποσταλεί από τον αιτούντα υπηρεσία (request agent) και παραδοθεί στον πάροχο υπηρεσίας (provider agent) και αντίστροφα. Τα μηνύματα που ανταλλάσσονται και ο τρόπος που γίνεται η ανταλλαγή περιγράφονται λεπτομερώς στο αρχείο WSDL (Web Services Description Language).

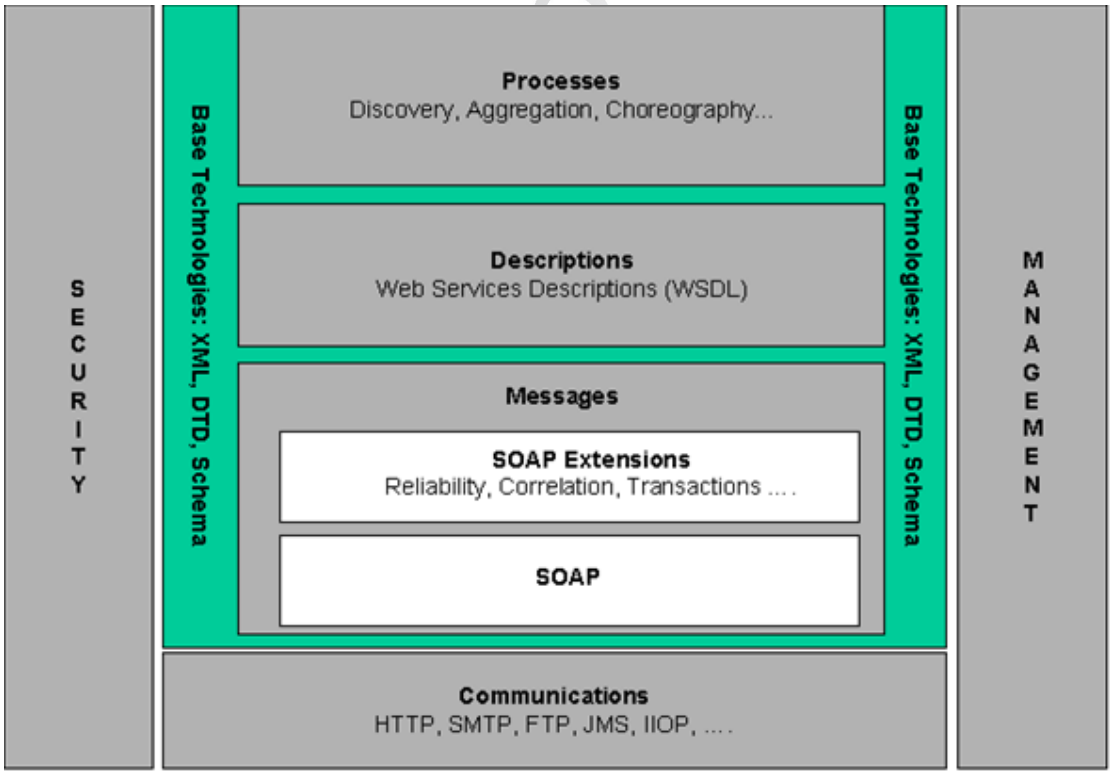
Επόμενο βήμα είναι η ανακάλυψη των υπηρεσιών, για την οποία υπάρχουν οι παρακάτω προσεγγίσεις :

- Η ύπαρξη ενός καταλόγου καταγραφής και δημοσίευσης των υπηρεσιών με τη μορφή υπηρεσίας καταγραφής (registry), όπως το UDDI (Universal Description, Definition and Integration) και το DISCO (Microsoft Discovery), που αποτελεί την πιο διαδεδομένη λύση
- Η ύπαρξη ιστοσελίδων παραπομπής σε υπηρεσίες, τύπου http://www.amazon.com και πρότυπα SOAP, WSDL, UDDI και DISCO έχουν γραφεί σε γλώσσα επισήμανσης XML (eXtensible Markup Language) και ενδεχομένως υπακούουν σε κάποιο έγγραφο DTD (Document Type Definition) ή XML Schema.



Εικόνα 1: Δομή των Web Services

Η τεχνολογία των Web Services αποτελεί την πιο εξελίξιμη και νεωτεριστική τεχνολογία του μέλλοντος, παγκοσμίως. Ο προσδιορισμός της αποτελεί προϊόν συνεργασίας κολοσσών της πληροφορικής, όπως η Microsoft, η Sun Microsystems και η IBM και αποτελεί την ιδανική λύση για διεπιχειρησιακές εφαρμογές και συνεργασίες (Business-to-Business, B2B).



Εικόνα 2:Αρχιτεκτονική των Web Services

Τα Web Services στηρίζονται σε ευρέως διαδεδομένα πρότυπα της βιομηχανίας της πληροφορικής και δυνητικά συνιστούν τον βέλτιστο τρόπο για την διασύνδεση των επιχειρησιακών εφαρμογών τόσο μέσα στο εταιρικό δίκτυο (intranet), αλλά και στο διαδίκτυο. Με τα Web Services μπορούν να διασυνδεθούν και να επικοινωνήσουν εσωτερικές εφαρμογές όπως π.χ. το Λογιστικό πακέτο με το πακέτο Διαχείρισης Πελατειακών Σχέσεων μέσω του intranet. Επίσης, μπορούν να χρησιμοποιηθούν για την διασύνδεση, την συνεργασία και την ανταλλαγή πληροφοριών μεταξύ ενδοεπιχειρησιακών εφαρμογών και εφαρμογών τρίτων, όπως π.χ. των συνεργατών και προμηθευτών μιας εταιρίας μέσω του διαδικτύου. Στηρίζονται στην XML και άλλες καινοτόμες τεχνολογίες και προσφέρουν ένα απλούστερο τρόπο για την επίτευξη κατανεμημένων εφαρμογών (distributed computing), διευκολύνοντας τα διασυνδεδεμένα συστήματα να ανταλλάξουν πληροφορίες και να συνδιαλλαγούν μεταξύ τους. Η Forward e-business αξιοποιώντας την εφαρμοσμένη εμπειρία της σε Java και XML εισέρχεται δυναμικά στον χώρο των Web Services, προσφέροντάς οικονομικές, ποιοτικές και ευέλικτες λύσεις, που θα οδηγήσουν οποιαδήποτε επιχείρηση στον δρόμο των παγκόσμιων εξελίξεων.

2.3 Πλεονεκτήματα των Web Services

Η αρχιτεκτονική των web services παρέχει αρκετά πλεονεκτήματα μερικά από τα οποία αναφέρονται παρακάτω:

- **Διαλειτουργικότητα:** Ένα web service παρέχει ανεξαρτησία τόσο από λειτουργικό σύστημα όσο και από το hardware. Οποιοδήποτε πρόγραμμα που συμβαδίζει με αυτή τη τεχνολογία μπορεί πολύ εύκολα να προσπελάσει μία τέτοια υπηρεσία.
- **Ενσωμάτωση:** Σε ένα υπάρχον λογισμικό σύστημα που λειτουργεί μέσα στο Internet η δημιουργία ενός web service δεν απαιτεί αλλαγές στον μηχανισμό του συστήματος.
- **Διαθεσιμότητα και δημοσίευση:** Οι πληροφορίες για τα web services δημοσιεύονται οπότε η εύρεση και η χρήση τους μπορεί να είναι ταχύτατες.
- **Επέκταση:** Ένα έτοιμο web service είναι δυνατό να ανανεωθεί με εύκολο τρόπο παρέχοντας έτσι επιπρόσθετες υπηρεσίες στους χρήστες του.
- **Μικρό κόστος δημιουργίας και χρήσης:** Εφόσον σε ένα λογισμικό σύστημα υπάρχει ήδη κάποια διαδικασία που χρειάζεται να επεκταθεί σε on-line υπηρεσία,

η δημιουργία του web service κοστίζει ελάχιστα. Επίσης το κόστος ενσωμάτωσης ενός web service σε κάποιο website ή σε δικτυακή εφαρμογή είναι πάρα πολύ μικρό. Ακόμα και στις περιπτώσεις που η χρήση κάποιου web service γίνεται με ενοικίαση σίγουρα το συνολικό κόστος της χρήσης είναι αρκετά πιο μικρό από το κόστος δημιουργίας της υπηρεσίας αυτής.

- **Χρήση λογισμικών συστημάτων:** Όλα τα λογισμικά συστήματα και ειδικότερα τα websites που χρησιμοποιούν έτοιμες υπηρεσίες γίνονται πιο λειτουργικά και πιο φιλικά αφού παρέχουν περισσότερες υπηρεσίες στους χρήστες.

2.4 Δημιουργία των Web Services

Υπάρχουν αρκετές διαφορετικές πλατφόρμες στις οποίες μπορεί να βασιστεί κανείς για τη δημιουργία ενός web service. Από τη μεριά της Microsoft, οι έτοιμες λύσεις που δίνει το περιβάλλον Visual Studio.net έχουν προσελκύσει πολλούς προγραμματιστές για να δημιουργούν τέτοιες υπηρεσίες.

Επίσης άλλες μεγάλες εταιρίες όπως η IBM και η ORACLE χρησιμοποιούν τα δικά τους προγραμματιστικά εργαλεία. Τέλος ακόμα και οι περισσότερες γλώσσες προγραμματισμού έχουν ενσωματώσει στις δυνατότητές τους την αυτόματη δημιουργία SOAP servers και την υποστήριξη των web services.

Μία δωρεάν λύση προσφέρεται και στους προγραμματιστές δυναμικών ιστοσελίδων που χρησιμοποιούν την γλώσσα PHP. Υπάρχουν έτοιμες βιβλιοθήκες (π.χ. nusoap) που μπορούν να χρησιμοποιηθούν από οποιοδήποτε για να δημιουργήσει απλά προσθέτοντας μόνο 5 γραμμές εντολών ένα web service.

2.5 Υπηρεσιοκεντρική Αρχιτεκτονική (Service Oriented Architecture – SOA)

Το μοντέλο που χρησιμοποιείται στις μέρες μας για την περιγραφή των «Υπηρεσιών Διαδικτύου» βασίζεται κυρίως στο κλασικό επιχειρησιακό μοντέλο, το οποίο έχουν υιοθετήσει οι διάφοροι ιδιωτικοί οργανισμοί και απεικονίζεται στο παρακάτω σχήμα

Εικόνα 3: Υπηρεσιοκεντρική Αρχιτεκτονική

Από το παραπάνω σχήμα μπορούμε να διακρίνουμε τις οντότητες που απαρτίζουν την τρέχουσα αρχιτεκτονική των υπηρεσιών διαδικτύου. Αυτές είναι η οντότητα που ζητάει την υπηρεσία (Service Requestor), η οντότητα που παρέχει την υπηρεσία (Service Provider) και τέλος η οντότητα του καταλόγου υπηρεσιών (Service Registry).

Με περισσότερη λεπτομέρεια, κάθε μία οντότητα έχει τους ακόλουθους ρόλους:

- **Η οντότητα, η οποία ζητάει την υπηρεσία** (Service Requestor), είναι στην ουσία ο «αιτούμενος» (Client) της υπηρεσίας και πυροδοτεί την έναρξη της όλης διαδικασίας. Αρχικά επιφορτίζεται με το έργο της αναζήτησης της κατάλληλης περιγραφής μιας υπηρεσίας και τις καταχωρήσεις της υπηρεσίας καταλόγου. Στη συνέχεια, και ενώ έχει εντοπίσει τον επιθυμητό παροχέα, εγκαθιδρύει σύνδεση με αυτόν, καλεί την υπηρεσία και λαμβάνει τα αποτελέσματα.
- **Η οντότητα που παρέχει την υπηρεσία** (Service Provider), δημοσιοποιεί την περιγραφή της υπηρεσίας ιστού στην οντότητα του καταλόγου υπηρεσιών (Service Registry) και ουσιαστικά παρέχει την υπηρεσία στο διαδίκτυο. Επιπλέον, της ανατίθεται ο ρόλος της λήψης των μηνυμάτων κλήσης για την υπηρεσία από έναν ή περισσότερους αιτούμενους και της παροχής των απαραίτητων αποτελεσμάτων.
- Τέλος, **η οντότητα του καταλόγου υπηρεσιών** (Service Registry), περιέχει καταχωρήσεις με τις περιγραφές των ήδη δημοσιοποιηθέντων υπηρεσιών διαδικτύου. Για τις υπηρεσίες αυτές παρέχονται τρόποι αναζήτησης ανάμεσα στις διάφορες περιγραφές.

Οι λειτουργίες που παρέχονται μέσω αυτής της αρχιτεκτονικής είναι οι ακόλουθες:

- Η καταχώρηση των απαραίτητων πληροφοριών για μια υπηρεσία από τη μεριά του παρόχου (publishing), η οποία επιτυγχάνεται μέσω της χρήσης της γλώσσας περιγραφής WSDL.
- Η αναζήτηση και η εύρεση στους καταλόγους της κατάλληλης περιγραφής μιας «Υπηρεσίας Διαδικτύου» (finding). Η επικρατούσα τεχνολογία καταλόγου ονομάζεται UDDI.
- Τέλος, έχουμε την εγκαθίδρυση σύνδεσης μεταξύ αιτούμενου και παροχέα (binding), κλήση της κατάλληλης υπηρεσίας και αποστολή των αποτελεσμάτων. Όλα τα παραπάνω επιτυγχάνονται με την ανταλλαγή μηνυμάτων SOAP μεταξύ των τριών εμπλεκόμενων οντοτήτων, η οποία βασίζεται στη γλώσσα μεταδεδομένων XML.

Το κλειδί της παραπάνω αρχιτεκτονικής είναι η περιγραφή της «Υπηρεσίας Διαδικτύου». Η περιγραφή είναι εκείνη που δημοσιοποιείται από τον παροχέα της υπηρεσίας στον κατάλογο υπηρεσιών. Η περιγραφή είναι το αποτέλεσμα της αναζήτησης του αιτούμενου στον κατάλογο υπηρεσιών. Η περιγραφή είναι εκείνη που λέει στον αιτούμενο όσα ακριβώς πρέπει να γνωρίζει προκειμένου να κάνει κλήση της υπηρεσίας που τον ενδιαφέρει.

Τέλος, η περιγραφή της υπηρεσίας μπορεί να περιέχει πληροφορία για το τι είδους είναι το αποτέλεσμα που αναμένεται να επιστραφεί ύστερα από την κλήση της υπηρεσίας.

Με την παραπάνω αρχιτεκτονική ήδη υπάρχουσες εφαρμογές μπορούν εύκολα να μετασχηματισθούν σε υπηρεσίες που να εξυπηρετούν άλλες υπάρχουσες – ή και νέες εφαρμογές. Πολλές υπηρεσίες μπορούν να συνδυαστούν και να συνεργασθούν για την παραγωγή ενός αποτελέσματος με τρόπο διαφανή προς τον τελικό χρήστη, στον οποίο δίνεται η αίσθηση ότι κλήθηκε μία και μόνο υπηρεσία. Επιπρόσθετα, οργανισμοί και επιχειρήσεις μπορούν ευκολότερα να κατασκευάσουν λογισμικό, που να αλληλεπιδρά με επιχειρησιακές διαδικασίες και να ανταποκρίνεται γρήγορα στις αλλαγές του επιχειρησιακού περιβάλλοντος.

2.6 Τεχνολογίες που χρησιμοποιούν τα Web Services

Οι τεχνολογίες που χρησιμοποιούν τα Web Services συμπεριλαμβάνουν:

- XML, που περιλαμβάνει βασική XML, XML schemas και XML parsers.
- SOAP (Simple Object Access Protocol), που αποτελεί ένα πρωτόκολλο επικοινωνίας εφαρμογών βασισμένο σε XML.
- REST (Representational State Transfer), που είναι μια μέθοδος που χρησιμοποιεί το πρωτόκολλο HTTP για τη μεταφορά των XML μηνυμάτων μεταξύ των Web Services.
- WSDL (Web Services Description Languages), που είναι ένα XML schema για περιγραφή των μηνυμάτων, λειτουργιών και αντιστοιχήσεις πρωτοκόλλων των υπηρεσιών διαδικτύου.
- UDDI (Universal Description Discovery and Integration), που είναι ο χώρος αποθήκευσης για καταχώρηση και αναζήτηση περιγραφών υπηρεσιών διαδικτύου.

Η ανάγκη για χρήση των υπηρεσιών διαδικτύου ανακύπτει από την απαίτηση των χρηστών να μπορούν να έχουν εύκολη πρόσβαση σε πληροφορία που μπορεί να δημοσιευτεί σε οποιοδήποτε μέρος του διαδικτύου. Η υπάρχουσα τεχνολογική υποδομή του παγκοσμίου ιστού αν και έχει διευκολύνει τον κόσμο των επιχειρήσεων έχει μερικούς περιορισμούς. Δεν καλύπτει την ανάγκη αυτόματης αλληλεπίδρασης μεταξύ εφαρμογών. Σήμερα οι εφαρμογές πρέπει να εκτελεστούν «με το χέρι» χρησιμοποιώντας έναν φυλλομετρητή. Επίσης χρειάζεται ένας καλύτερος μηχανισμός για την αναζήτηση πληροφορίας στο διαδίκτυο από αυτόν που χρησιμοποιείται σήμερα και βασίζεται στην «σάρωση» HTML σελίδων προκειμένου να βρεθεί το ζητούμενο αλφαριθμητικό ή ομάδα αλφαριθμητικών. Οι υπηρεσίες διαδικτύου έρχονται να καλύψουν τέτοιου είδους κενά εκμεταλλευόμενες την κατανεμημένη μορφή του διαδικτύου και παρέχοντας ένα νέο μοντέλο ανταλλαγής της πληροφορίας.

2.6.1 HTTP (HyperText Transfer Protocol)

Το πρωτόκολλο Μεταφοράς Υπέρ-κειμένου HTTP (HyperText Transfer Protocol) είναι το στάνταρ πρωτόκολλο μεταφοράς στον Παγκόσμιο Ιστό. Κάθε ανταλλαγή πληροφορίας συνίσταται από μια αίτηση ASCII, ακολουθούμενη από μία απόκριση RFC 822 τύπου MIME, η οποία αποτελεί το de facto πρότυπο μορφής μηνύματος για το ηλεκτρονικό ταχυδρομείο.

Παρά το γεγονός ότι είναι σύνηθες να χρησιμοποιείται το πρωτόκολλο Ελέγχου Μεταφοράς TCP (Transmission Control Protocol) για τη σύνδεση μεταφοράς, δεν απαιτείται επίσημα από το πρότυπο.

2.6.2 Πρωτόκολλο XML (eXtensible Markup Language)

Η XML (eXtensible Markup Language) αναπτύχθηκε από το W3C's XML Working Group το 1996 και συνδυάζει την ισχύ και την επεκτασιμότητα της SGML (Standard Generalized Markup Language), από την οποία προέρχεται, με την απλότητα που απαιτεί η κοινότητα του Διαδικτύου. Είναι μια μεταφερτή, ευρέως υποστηριζόμενη, ανοικτή τεχνολογία για την περιγραφή δεδομένων.

Η XML περιγράφει μια κατηγορία αντικειμένων δεδομένων που ονομάζονται XML έγγραφα και εν μέρει τη συμπεριφορά των προγραμμάτων υπολογιστών που τα επεξεργάζονται. Τα έγγραφα XML αποτελούνται από μονάδες αποθήκευσης, τις οντότητες (entities), οι οποίες περιέχουν αναλυμένα λεκτικά (parsed) ή μη αναλυμένα λεκτικά (unparsed) δεδομένα. Τα αναλυμένα λεκτικά δεδομένα (parsed) συνίστανται από χαρακτήρες, ορισμένοι από τους οποίους αποτελούν δεδομένα χαρακτήρων και κάποιοι άλλοι δημιουργούν markup, δηλαδή κωδικοποιούν μια περιγραφή της διάταξης και της λογικής δομής του εγγράφου. Για τα μη αναλυμένα λεκτικά δεδομένα (unparsed) δεν είναι γνωστό εάν είναι κείμενο ή όχι, αλλά ακόμη κι αν είναι κείμενο, ενδέχεται να μην είναι XML, οπότε αγνοούνται κατά τη διαδικασία της ανάλυσης λεκτικών δεδομένων (parsing), στην οποία θα αναφερθούμε εκτενέστερα στη συνέχεια, και διοχετεύονται όπου προβλέπει η εφαρμογή.

Η XML αποτελεί σήμερα το πρότυπο για την αποθήκευση δεδομένων που ανταλλάσσονται μεταξύ των εφαρμογών χάρη στα ακόλουθα χαρακτηριστικά που παρουσιάζει :

- Υποστηρίζει ανεξαρτησία από τα δεδομένα και διαχωρίζει τα περιεχόμενα από τον τρόπο εμφάνισής τους και τον χειρισμό τους, οπότε διευκολύνεται η λεκτική ανάλυσή τους (parsing).
- Διατίθενται έτοιμοι τρόποι σύνδεσης των κειμένων XML με τα πλέον σύγχρονα προγραμματιστικά περιβάλλοντα, όπως το Document Object Model (DOM) και το Simple API for XML (SAX).

- Είναι επεκτάσιμη και ανεξάρτητη από πλατφόρμες, γεγονός που την καθιστά απρόσβλητη σε τεχνολογικές αλλαγές.
- Τα έγγραφα XML είναι αναγνώσιμα από ανθρώπους και μηχανές και παρότι δεν προορίζονται για ανάγνωση προσφέρουν αυτή τη δυνατότητα στο χρήστη εάν κριθεί αναγκαίο.
- Είναι πλήρως συμβατή με Unicode, οπότε μπορεί να χειριστεί την πληροφορία που έχει γραφεί σε οποιαδήποτε ανθρώπινη γλώσσα. Παράλληλα, υποστηρίζει διεθνείς και τοπικές προσαρμογές.
- Χρησιμοποιώντας την XML, οι δημιουργοί των εγγράφων μπορούν να αναπαραστήσουν πολύπλοκες δομές δεδομένων, όπως λίστες, εγγραφές και δένδρα και να περιγράψουν οποιονδήποτε τύπο δεδομένων, συμπεριλαμβάνοντας μαθηματικούς τύπους, οδηγίες διαμόρφωσης λογισμικού, μουσική, αποδείξεις και οικονομικές αναφορές.

Ενδεικτικά αναφέρουμε ότι η XML αποτελεί τη βάση του RDF (Resource Description Framework), του Σημασιολογικού Ιστού (Semantic Web), της WML (Wireless Markup Language), της MathML και της GML (Geography Markup Language).

Για να είναι ένα έγγραφο XML **σωστό** θα πρέπει να είναι συγχρόνως :

- Καλοσχηματισμένο (well-formed), δηλαδή να υπακούει σε όλους τους συντακτικούς κανόνες της XML. Εάν ένα έγγραφο δεν είναι συντακτικά σωστό, τότε δεν θεωρείται έγγραφο XML, η ανάλυση (parsing) σταματά και ο αναλυτής (parser) δίνει ένα σφάλμα.
- Έγκυρο (valid), δηλαδή να ικανοποιεί ένα σύνολο κανόνων που έχει ορίσει ο χρήστης και περιλαμβάνεται σε προαιρετικά έγγραφα XML, όπως τους Ορισμούς Τύπων Εγγράφων (Document Type Definitions – DTDs) και τα Σχήματα XML (XML Schemas).

2.6.2.1 DTD

Η XML χρησιμοποιεί ένα μοντέλο για την περιγραφή των εγγράφων XML. Ένα τέτοιο μοντέλο ονομάζεται Document Type Definition (DTD). Η XML το χρησιμοποιεί για να περιγράψει τα

δεδομένα. Το DTD είναι ένας μηχανισμός που κληρονομήθηκε από την γλώσσα SGML και χρησιμοποιείται για την περιγραφή κάθε αντικειμένου σε ένα έγγραφο XML, όπως είναι τα στοιχεία και τα χαρακτηριστικά. Αποτελεί ένα σύνολο κανόνων που αφορούν στα tags της γλώσσας XML. Πιο συγκεκριμένα ορίζει κανόνες για το ποια είναι τα επιτρεπτά ονόματα στοιχείων και τα επιτρεπτά υποστοιχεία (subelements) και χαρακτηριστικά (attributes) για κάθε συγκεκριμένο στοιχείο. Ακολουθεί ένα έγγραφο DTD.

Να κάνουμε τις εξής παρατηρήσεις σχετικά με τη δομή και τη σύνταξη των εγγράφων DTD:

- Όλα τα ονόματα των αντικειμένων ξεκινούν με γράμμα και δεν επιτρέπεται ο χαρακτήρας κενού διαστήματος. Στην θέση του χρησιμοποιείται το ενωτικό.
- Η XML υποστηρίζει πλήρως το πρότυπο κωδικοποίησης χαρακτήρων Unicode, συνεπώς μπορούμε ελεύθερα να χρησιμοποιήσουμε ελληνικούς χαρακτήρες ώς έγκυρους χαρακτήρες ονομάτων για τα στοιχεία του εγγράφου μας.
- Οι χαρακτήρες *, ?, και + ονομάζονται χαρακτήρες ύπαρξης και υποδεικνύουν αν και πώς τα στοιχεία στην λίστα των πεδίων επαναλαμβάνονται. Αν δεν υπάρχει δείκτης ύπαρξης σε ένα στοιχείο αυτό σημαίνει ότι το συγκεκριμένο στοιχείο εμφανίζεται μια φορά. Στοιχείο με τον χαρακτήρα + πρέπει να εμφανίζεται μια τουλάχιστον φορά. Με τον χαρακτήρα * καμία ή περισσότερες φορές. Με τον χαρακτήρα ? σημαίνει ότι το στοιχείο μπορεί να εμφανιστεί το πολύ μια φορά.
- Οι χαρακτήρες , και | ονομάζονται συζευκτήρες. Βοηθούν στον διαχωρισμό των παιδιών ενός στοιχείου μέσα στο μοντέλο περιεχομένου. Ο χαρακτήρας , σημαίνει ότι τα δύο στοιχεία αριστερά και δεξιά του κόμματος πρέπει να εμφανίζονται με την ίδια σειρά μέσα στο έγγραφο. Ενώ ο χαρακτήρας | σημαίνει ότι μόνο το ένα από τα στοιχεία πρέπει να εμφανίζεται στο έγγραφο.

Ένα έγγραφο XML μπορεί να ανήκει σε μια από τις εξής δύο βασικές κατηγορίες:

- Έγκυρο (valid) , δηλαδή να διαθέτει ένα DTD που το συνοδεύει
- Σωστά μορφοποιημένο (Well-formed) που δεν διαθέτει κάποιο συνοδευτικό DTD, απλά υποχρεούνται να εφαρμόζει όλους τους κανόνες σύνταξης της XML.

Κάθε διαφορετικός τύπος έγκυρου εγγράφου XML χρησιμοποιεί το δικό του DTD. Για να είναι εφικτή η επαλήθευση της εγκυρότητας ενός XML για συμμόρφωση με τους κανόνες που ορίζει το συνοδευτικό DTD. Με τα ίδια προγράμματα πρέπει να ελέγχονται και τα ίδια τα έγγραφα DTD για συνέπεια με τους κανόνες που ορίζονται στο πρότυπο DTD. Τα γνωστότερα προγράμματα validators είναι τα εξής:

- Γραμμής εντολών, όπως για παράδειγμα το XSV
- Αυτά που διαθέτουν API για τους προγραμματιστές όπως είναι τα Apache Xerces IBM Schema Quality Checker Microsoft MSXML4.0
- Validators με γραφικό περιβάλλον XML Spy Turbo XML

2.6.2.2 XML Schemas

Τα XML Schemas αποτελούν σύσταση του οργανισμού W3C και ουσιαστικά είναι έγγραφα XML που καθορίζουν πώς πρέπει να δομούνται κάποια άλλα έγγραφα XML. Το πλεονέκτημά τους σε σχέση με τα DTDs (Document Type Definitions) είναι ότι μπορούν να υποστούν χειρισμούς, παραδείγματος χάρη μπορούν να προστεθούν ή να διαγραφούν στοιχεία, όπως σε οποιοδήποτε άλλο έγγραφο XML.

Τα XML Schemas εγγυώνται ότι ο σωστός τύπος δεδομένων αποθηκεύεται σε κάθε στοιχείο, για παράδειγμα το περιεχόμενο του στοιχείου *Date_of_birth* θα είναι τύπου date και όχι ακέραιος ή κάποια ακολουθία χαρακτήρων (string). Επίσης, επιτρέπουν στον χρήστη να ορίσει δικούς του τύπους δεδομένων, επιπλέον αυτών που ήδη υπάρχουν στο σύστημα τύπων XML Schema Definition (XSD). Διευκολύνουν τη μετάβαση ανάμεσα στις πλατφόρμες διότι το σύστημα τύπων XML Schema Definition (XSD) είναι ανεξάρτητο από πλατφόρμες, οπότε δεν δημιουργείται πρόβλημα εάν ένας .NET εξυπηρετητής στείλει μια τιμή ακεραίου (integer) μήκους 32 bit σε έναν Visual Basic 6.0 πελάτη, για τον οποίο ο τύπος ακέραιος (integer) έχει μήκος 16 bit.

Τα DTD είναι ένα χαρακτηριστικό της XML που κληρονομήθηκε από τον πρόγονο της γλώσσας, την SMGML. Για τον λόγο αυτό τα DTD δεν υποστηρίζουν ενδογενώς τους χώρους ονοματοδοσίας. Εξαιτίας του παραπάνω γεγονότος αλλά και πολλών περιορισμών που θέτει το DTD λόγω της συσχέτισης του με την SMGL, το W3C εξετάζει την αντικατάσταση των DTD με τη νέα σύσταση των XML Schemas. Ένα XML Schema είναι ένα XML έγγραφο με επέκταση αρχείου .xsd το οποίο εκφράζει τους κανόνες που πρέπει να ακολουθεί ένα XML έγγραφο. Ανάλογα με την λειτουργία των DTD, κάθε καλά έγκυρο έγγραφο XML μπορεί να συνοδεύεται εναλλακτικά από ένα XML Schema. Λόγω των πολλών πλεονεκτημάτων του XML Schema έναντι του DTD, πολλές φορές είναι χρήσιμο να μετατρέπουμε τα διαθέσιμα .dtd αρχεία .xsd. Τα βασικότερα πλεονεκτήματα που προσφέρουν τα XML Schemas έναντι των DTD είναι τα εξής:

- Η γλώσσα που χρησιμοποιείτε για την σύνταξη των .xsd αρχείων ακολουθεί σύνταξη ανάλογη με την σύνταξη των .xml εγγράφων. Εν αντιθέσει τα έγγραφα dtd χρησιμοποιούν σύνταξη ασύμβατη με αυτή των .xml εγγράφων.
- Τα XML Schemas υποστηρίζουν τους χώρους ονοματοδοσίας (namespaces). Ένα namespaces ορίζει το σύνολο στο οποίο ανήκουν τα ονόματα των στοιχείων σε ένα έγγραφο XML.
- Υποστηρίζουν ενδογενώς περισσότερους τύπους δεδομένων από ότι τα DTD. Ενώ τα DTD υποστηρίζουν 10 τύπος δεδομένων, τα Schemas υποστηρίζουν τουλάχιστον 44 τύπους. Επίσης τα XML Schemas επιτρέπουν τον ορισμό νέων σύνθετων τύπων και έτσι αυξάνονται κατά πολύ οι δυνατότητες περιγραφής δεδομένων και κανόνων που τα διέπουν.
- Υποστηρίζουν κανονικές εκφράσεις (regular expressions). Ειδικά κατά τον ορισμό νέων σύνθετων τύπων η δυνατότητα αυτή είναι πάρα πολύ χρήσιμη.
- Τα Schemas προσφέρουν δυνατότητες συμπύκνωσης των tags (inlined element declarations), με αποτέλεσμα την μείωση της πολυπλοκότητας και την ευκολότερη σάρωση και ανάγνωση του εγγράφου κανόνων .xsd.
- Στα Schemas μπορούν να οριστούν στοιχεία με μηδενικό (null) περιεχόμενο ή ακόμα και δύο ή περισσότερα στοιχεία που έχουν το ίδιο όνομα , αλλά διαφορετικό περιεχόμενο.
- Μπορούν να χρησιμοποιηθούν σε ένα έγγραφο περισσότερα του ενός Schemas , αρκεί όλα να ανήκουν στο ίδιο namespace.
- Ένα Schema με το στοιχείο <include> μπορεί να χρησιμοποιεί στοιχεία που είναι δηλωμένα σε ένα άλλο Schema και με το στοιχείο <redefine> μπορεί επίσης να επανακαθορίζει τα στοιχεία αυτά.
- Η επεκτασιμότητα ενός Schema αυξάνεται αν ενσωματωθούν στο αρχείο .xsd οι τεχνολογίες XSLT/Xpath και Schematron [XFR02].
- Επιτρέπεται στα XML έγγραφα να χρησιμοποιούν πρόσθετα στοιχεία, πέραν αυτών που ορίζονται στο XML Schema, με βάση τις ανάγκες του δημιουργού περιεχομένου. Στην περίπτωση αυτή δηλώνονται τα στοιχεία <any> και <anyAttribute> στο .xsd αρχείο για να δηλώσουν την επεκτασιμότητα που περιγράφτηκε.

2.6.2.3. XML (XML Parsers)

Αν και τα έγγραφα XML είναι αρχεία κειμένου, η ανάκτηση δεδομένων από αυτά μέσω των τεχνικών σειριακής προσπέλασης αρχείων δεν είναι ούτε πρακτική, ούτε αποτελεσματική, ειδικά σε περιπτώσεις όπου τα δεδομένα πρέπει να προστεθούν ή να αφαιρεθούν δυναμικά. Το κενό αυτό έρχεται να καλύψει η διαδικασία της λεκτικής ανάλυσης (parsing), η οποία αναλύει το κείμενο στα επιμέρους στοιχεία του, στα οποία συμπεριλαμβάνονται αρχή ετικέτας, τέλος ετικέτας, κείμενο, ιδιότητες.

Στην διαδικασία της λεκτικής ανάλυσης εμπλέκεται πάντα ένας λεκτικός αναλυτής (parser). Ο λεκτικός αναλυτής κατέχει κεντρικό ρόλο στην αντιστοίχηση και αλληλοσύνδεση της δενδρικής δομής του κειμένου XML με την αντίστοιχη (συνήθως αντικειμενοστραφή) δομή στην χρησιμοποιούμενη γλώσσα, δεδομένου ότι το κείμενο XML είναι ένας συρμός χαρακτήρων που ανταλλάσσεται μέσω μίας επικοινωνιακής σύνδεσης, ενός αρχείου στον δίσκο, κλπ. Οι πιο γνωστοί λεκτικοί αναλυτές είναι ο DOM (Document Object Model) και ο SAX (Simple API for XML).

Το Μοντέλο Αντικειμένου Εγγράφου (Document Object Model - DOM) αποθηκεύει τα δεδομένα εγγράφου ως δομές δένδρων στη μνήμη με αυτόματο τρόπο, χωρίς την παρέμβαση του χρήστη κατά την εξέλιξή της ανάλυσης. Βοηθά τον χρήστη να θεωρήσει την δενδρική μορφή του XML κειμένου με αντικειμενοστραφή τρόπο, καθώς το δένδρο DOM αναπαριστά κάθε στοιχείο του εγγράφου XML (στοιχείο, ιδιότητα, κλπ.) ως ένα κόμβο στο δένδρο, και να επέμβει σε αυτό με συγκεκριμένες διεπαφές. Με το XML DOM παρέχεται, λοιπόν, η δυνατότητα στον χρήστη να δημιουργήσει ένα XML κείμενο, να πλοηγηθεί μέσα σε αυτό και να προσθέσει, να μεταβάλει και να αφαιρέσειστοιχεία του.

Το Simple API for XML (SAX) μοιάζει με σειριακό αναγνώστη (SAX reader) του κειμένου XML, χαρακτήρα προς χαρακτήρα, ο οποίος αναγνωρίζοντας τα διάφορα μέρη του αρχείου, δημιουργεί αντίστοιχα συμβάντα (events), στα οποία το πρόγραμμα του χρήστη οφείλει να ανταποκριθεί σύμφωνα με τις επιθυμίες του. Μόλις ο χειριστής συμβάντων (event handler) που έχει δημιουργήσει ο χρήστης αντιμετωπίσει το συμβάν, ο λεκτικός αναλυτής συνεχίζει μέχρι το επόμενο σημείο. Με τον λεκτικό αναλυτή SAX, λοιπόν, ο χρήστης βρίσκεται στο κατώτερο προγραμματιστικά περιβάλλον που του δίνει πλήρη πρόσβαση στο έγγραφο XML. Η ένταξη των επιθυμιών του (μέσω μιας εφαρμογής) είναι πιο επίπονη σε σχέση με το DOM, αλλά οι διαθέσιμες δυνατότητες και η ταχύτητα υπερτερούν. Ο χρήστης δεν περιμένει να δημιουργηθεί και να του προσφερθεί η πλήρης δομή του XML σύμφωνα με το DOM μοντέλο, αλλά ειδοποιείται και μπορεί να ανταποκριθεί στα σχετικά συμβάντα “on the fly”, καθώς διαβάζεται το XML κείμενο και χωρίς να περιμένει το τέλος αυτής της ανάγνωσης.

2.6.2.4 XSLT / XSFO

Για το στυλ και την μορφοποίηση των εγγράφων XML, δηλαδή τον τρόπο παρουσίασης του εγγράφου μέσα από μια εφαρμογή, όπως ένας browser, ή ένας επεξεργαστής κειμένου, υπάρχουν τεχνολογίες που καθιστούν τα έγγραφα XML πιο εύκολα παραμετροποιήσιμα σε ότι έχει να κάνει με την εμφάνισή τους.

Ένας τρόπος ελέγχου της μορφοποίησης των XML εγγράφων είναι να ενσωματώσουμε στο έγγραφο XML κάποια HTML tags, που συνεισφέρουν στην μορφοποίηση, με στόχο την επιθυμητή παρουσίαση του εγγράφου μέσα από έναν web browser. Ειδικότερα αν καταφέρουμε να έχουμε μια αυστηρή έκδοση της HTML που να υπακούει σε όλους τους κανόνες δομής της XML, τότε μπορούμε να δημιουργήσουμε την μορφοποίηση που θέλουμε, διατηρώντας όμως την απαραίτητη αυστηρή δομή της XML. Μια τέτοια προσπάθεια εκφράζεται από την εξελιγμένη γλώσσα σήμανσης, την XHTML.

Για την επεξεργασία των εγγράφων XML χρησιμοποιούνται και οι τεχνολογίες XSLT και XSLFO (XSL Formating Objects), οι οποίες μαζί με την σύσταση xpath (XML Path Language) υπάγονται στην σύσταση XSL (XML Stylesheet Language) του W3C. Η σύσταση XSL αφορά στο στυλ των εγγράφων XML και μας δίνει την δυνατότητα να μετατρέπουμε έγγραφα XML σε άλλα έγγραφα XML με διαφορετικό DTD ή XMK Shema, όπως επίσης και σε έγγραφα διαφορετικών μορφών (HTML, text και άλλα). Ακόμη, η XSL βοηθά όταν θέλουμε να δημοσιεύσουμε έναν μεγάλο όγκο από έγγραφα είτε να αναδιοργανώσουμε κάποια έγγραφα XML ώστε να δημιουργήσουμε πίνακες περιεχομένων ή άλλες χρηστικές δομές αναπαράστασης των πληροφοριών. Η σύσταση XSLT αφορά στον μετασχηματισμό ενός εγγράφου XML, ενώ η σύσταση XSLFO στην μορφοποίησή του για παρουσίαση. Την ίδια λειτουργία με την XSLFO επιτελεί και η τεχνολογία CSS (Cascading Style Sheets) ωστόσο η CSS είναι πολύ δημοφιλέστερη και μονοπωλεί το ενδιαφέρον των κατασκευαστών λογισμικού και παροχών περιεχομένου. Ακολουθεί αμέσως μια ανάλυση των τριών βασικών τεχνολογιών με τις οποίες επιτυγχάνεται ο μετασχηματισμός και η μορφοποίηση των εγγράφων XML.

CSS

Η τεχνολογία CSS (Cascading Style Sheets) είναι ένας μηχανισμός κανόνων που χρησιμοποιείται για να προσδώσει συγκεκριμένο στυλ στα στοιχεία ενός εγγράφου. Με την

χρήση του ενός αρχείου κανόνων CSS παρέχονται εντολές στον browser πώς να εμφανιστεί XML ή HTML. Έτσι αποσυνδέονται γενικά η εμφάνιση μιας σελίδας από το περιεχόμενό της. Το CSS συστάθηκε από W3C και έχει δύο εκδόσεις: την έκδοση CSS 1.0 και την έκδοση CSS 2.0 που υποστηρίζει τη XML σε μεγαλύτερο βαθμό. Με βάση το CSS δημιουργήθηκε και σύσταση XSLFO από το W3C, όπως θα αναφέρουμε παρακάτω.

XSLT

Τα αρχεία XSLT ονομάζονται και XSLT stylesheets και έχουν σύνταξη παρόμοια με αυτή της XML. Κάθε XSLT stylesheet είναι ένα καλά δομημένο (well formed) XML έγγραφο. Τα αρχεία XSL stylesheets βοηθούν πολύ σε κάθε προσπάθεια δημοσίευσης περιεχομένου που υποστηρίζει XML. Ένα χαρακτηριστικό παράδειγμα είναι η δημιουργία της μορφοποίησης ενός website από περιεχόμενο μιας βάσης δεδομένων βασισμένης σε XML. Στην περίπτωση αυτή ένα αρχείο με οδηγίες μορφοποίησης XSLT βοηθά στην μετατροπή του εγγράφου .xml σε ένα έγγραφο .html. Η συνεισφορά του XSLT επεκτείνεται όταν θέλουμε να έχουμε πολλές διαφορετικές εκδόσεις του περιεχομένου ενός website όπως για παράδειγμα αρχεία .html, αρχεία απλού κειμένου txt (plain text), αρχεία για εμφάνιση από φορητές συσκευές μέσω WAP (WML) κ .α.

Για την αναγνώριση των XSLT stylesheets και την εκτέλεση των XML μετασχηματισμών που αυτά ορίζουν απαιτείται κάποιο πρόγραμμα XSLT processor.

Τέτοια προγράμματα είναι ενσωματωμένα σε μεγαλύτερα πακέτα επεξεργασίας εγγράφων XML, όπως είναι το XML Spy. XSLT processors επίσης ενσωματώνονται και σε browsers, όπως για παράδειγμα ο Internet Explorer 6, ο οποίος διαθέτει το πρόγραμμα MSXML parser. Ωστόσο υπάρχουν και ανεξάρτητα προγράμματα XSLT processors, τα οποία μπορούν να ανακτηθούν από το Internet.

Το δομικό στοιχείο σε ένα XSLT stylesheet είναι το element <xsl:stylesheet>. Διαθέτει σαν υποστοιχεία μια σειρά από elements <xsl:template>….<xsl:template>. Μέσα σε αυτά τα <xsl:template> elements εκφράζονται οι μετασχηματισμοί που πρέπει να γίνουν στο έγγραφο XML. Κάθε template περιέχει ένα attribute με όνομα match. Το match είναι μια διαδρομή για ο στοιχείο του XML εγγράφου στο οποίο αναφέρεται το template. Παράλληλα με το match, κάθε template υπάρχει μια δήλωση <xsl:apply-templates/> η οποία είναι μια κλήση που πρέπει να γίνει στο stylesheet κατά την ιεραρχική σάρωση των στοιχείων του πηγαίου XML εγγράφου από τον αναλυτή.

Τα βασικά σύμβολα που χρησιμοποιούνται για την σύνταξη των διαδρομών είναι τα σύμβολα / (στοιχείο ρίζα και άμεσος απόγονος, // (όλοι οι απόγονοι), * (όλοι οι άμεσοι απόγονοι) και | (επιλογή ενός εκ των δύο στοιχείων που βρίσκονται εκατέρωθεν του συμβόλου).

Οι διαδρομές αντιστοιχίζονται τόσο σε elements όσο και σε attributes του πηγαίου XML εγγράφου. Η διαδρομή που αναφέρεται σε ένα στοιχείο url με attribute το protocol που έχει τιμή mailto γράφεται ως match="url[@protocol='mailto']", δηλαδή το attribute στο οποίο γίνεται η αναφορά μπαίνει σε αγκύλες και πριν από το όνομα του τοποθετείται το σύμβολο.

Μπορούμε να χρησιμοποιήσουμε μια από τις έτοιμες συναρτήσεις της XSL, οι οποίες παίρνουν μηδέν ή περισσότερα ορίσματα και επιστρέφουν κάτι σχετικό με το πηγαίο έγγραφο. Για παράδειγμα η συνάρτηση text() επιστρέφει τα περιεχόμενα ενός στοιχείου. Παράλληλα με τις έτοιμες συναρτήσεις μπορούμε να ορίσουμε και δικές μας προσθέτοντας στο XSLT stylesheet ένα στοιχείο <xsl:fuctions > ως εξής:

< xsl:fuctions ns="myns" type="text/javascript">

fuction today()

{

return Date(), to String()

}

</ xsl:fuctions >

Το πρόγραμμα parser ξεκινά την σάρωση του πηγαίου XML εγγράφου από την ρίζα προς τους τελευταίους κόμβους ιεραρχικά, με βάση έναν αλγόριθμο DFS (Depht First Search). Σε κάθε βήμα της σάρωσης, ο αναλυτής προσπαθεί να αντιστοιχίσει τον τρέχοντα κόμβο ένα από τα πρότυπα που υπάρχουν στο XSLT stylesheet. Αν υπάρχει μια τέτοια αντιστοιχία , ο αναλυτής δημιουργεί τους κόμβους που του λέει το stylesheet στο παραγόμενο δέντρο (παραγόμενο XML έγγραφο). Όταν ο αναλυτής συναντήσει το στοιχείο <xsl:apply-templates>, τότε προχώρα στα παιδιά του τρέχοντος κόμβου και επαναλαμβάνει την διαδικασία, δηλαδή προσπαθεί να τα αντιστοιχίσει με κάποιο πρότυπο. Αν ο αναλυτής δεν βρει καμία αντιστοίχιση για ένα στοιχείο τότε εφαρμόζει το ενσωματωμένο πρότυπο του αναλυτή. Το ενσωματωμένο πρότυπο περιέχει συνήθως μόνο μια κλήση <xsl:apply-templates/> με χαρακτηριστικό match="* | /" που σημαίνει ότι ο αναλυτής οδηγείται στα παιδιά του

στοιχείου για το οποίο δεν είχε αρχικά βρει αντιστοιχία στο XSLT stylesheet. Έτσι η διαδικασία δεν τερματίζεται, παρά μόνο όταν ολοκληρωθεί η σάρωση όλου του πηγαίου XML εγγράφου. Επίσης κάποια στοιχεία έχουν σαν μοναδικό παιδί τους ένα κείμενο και για το κείμενο αυτό δεν έχει οριστεί κάποιο template στο XSLT. Τότε πάλι εκτελείται ένα ενσωματωμένο template του αναλυτή που υπαγορεύει την παρουσίαση του κειμένου στο τρέχον σημείο του παραγόμενου δέντρου.

Τα attributes που ορίζονται στο παραγόμενο XML δέντρο είναι είτε ρητά δηλωμένα στο XSLT είτε υπολογίζονται κατά την δημιουργία του παραγόμενου δέντρου.

Η δεύτερη περίπτωση γίνεται με την δήλωση <xsl:attribute name="….">…..</xsl:attribute> για τον υπολογισμό της τιμής του attribute name κατά την δημιουργία του παραγόμενου δέντρου.

Για την δημιουργία νέων αντικειμένων υπάρχουν μια σειρά από εντολές XSLT. Για παράδειγμα υπάρχουν οι εντολές <xsl:element> και <xsl:attribute> για την δημιουργία ενός στοιχείου ή ενός χαρακτηριστικού στο παραγόμενο δέντρο και υπολογισμό της τιμής του κατά την δημιουργία του δέντρου.

Όπως έχουμε πει υπάρχουν δύο τρόποι για να παράγουμε το τελικό επιθυμητό HTML έγγραφο. Είτε να εφαρμόσουμε εμείς εσωτερικά στον αναλυτή μας το XSLT και να παράγουμε το HTML είτε να δώσουμε στον τελικό χρήστη το πηγαίο XML έγγραφο και το XSLT και ο μετασχηματισμός να εκτελεστεί στην πλατφόρμα του χρήστη. Στην δεύτερη αυτή περίπτωση πρέπει να είμαστε προσεκτικοί με θέματα συμβατότητας ώστε το XSLT μας να καλύπτει τις προδιαγραφές του αναλυτή του χρήστη (π . χ ένας browser).

Σχετικά με την προτεραιότητα εφαρμογής των προτύπων του XSLT, ισχύουν τα εξής:

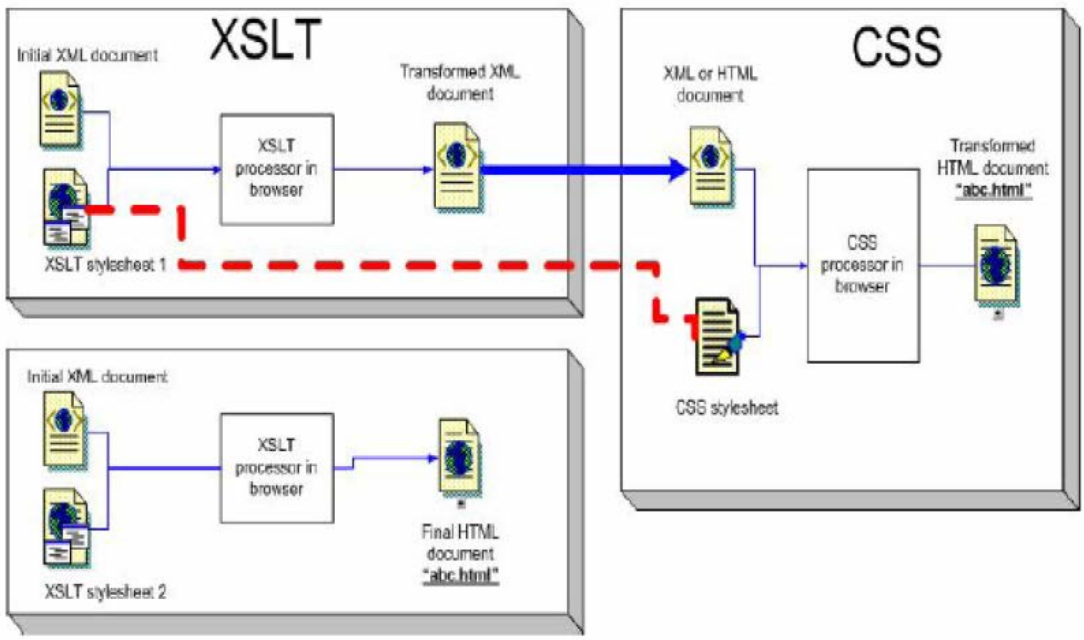
- Τα πρότυπα με τις πιο συγκεκριμένες διαδρομές προηγούνται αυτών που είναι λιγότερο συγκεκριμένα. Αν αυτό το κριτήριο δεν αρκεί για την επιλογή του επόμενου template, τότε
- Μεγαλύτερη προτεραιότητα έχει το template που βρίσκεται τελευταίο στο XSLT stylesheet.
- Όταν θέλουμε να επιβάλλουμε σε ένα XML έγγραφο με ένα αρχικό XML Schema να ακολουθεί ένα άλλο XML Schema, τότε πρέπει να δημιουργήσουμε ένα XSLT stylesheet για μετατροπή μεταξύ των δύο schemas και εν συνέχεια να εκτελέσουμε αυτό το XSLT stylesheet μέσα από έναν browser ή έναν XSLT αναλυτή. Εκεί θα πρέπει στην κορυφή

του εγγράφου να γίνει μια δήλωση για αναφορά στο XSLT που χρησιμοποιούμε. Η δήλωση αυτή είναι της μορφής <?xml-stylesheet href="…"type="text/xsl"?>.

Ένα XSLT stylesheet μπορεί επίσης να περιέχει μια αναφορά σε ένα CSS stylesheet για την τελική μορφοποίηση του παραγόμενου εγγράφου XML. Έτσι με την χρήση των templates στα XSLT stylesheet έχουμε δύο επιλογές:

- Να χρησιμοποιήσουμε XSLT, το οποίο εφαρμόζεται πάνω στο επιθυμητό XML έγγραφο και να παράγουμε το τελικό HTML (XHTML) έγγραφο.
- Να χρησιμοποιήσουμε XSLT που έχει αναφορά σε κάποιο CSS και να παράγουμε ένα XML έγγραφο. Το τελικό αυτό XML έγγραφο μορφοποιείται με βάση τις οδηγίες μορφοποίησης του CSS στο οποίο υπάρχει αναφορά.

Τα παραπάνω παρουσιάζονται και σχηματικά ως εξής:



Εικόνα 4: Σύγκριση των XSLT και CSS

XSLFO

Τα XSLFO (XML Formating Objects) είναι αρχεία XSL που όμως έχουν διαφορετικά tags μέσα σε κάθε <xsl:template> element από αυτά που υπάρχουν μέσα σ' ένα XSLT stylesheet. Τα νέα tags που χρησιμοποιούνται στα XSLFO βασίζονται στην λογική των CSS stylesheets. Κανένας από τους γνωστούς browser δεν υποστηρίζει το XSLFO αλλά οι περισσότεροι υποστηρίζουν πλέον εναλλακτικά το XSLFO και το CSS2.

Η συντριπτική πλειοψηφία των εγγράφων βασίζονται στην HTML και την XHTML για την εμφάνιση τους στο Web. Με αυτή τη λογική το XSLFO μπορεί να αποτελεί μια τελείως νέα λογική, αλλά αυτό που χρησιμοποιείτε στην πράξη και θα συνεχίσει να υποστηρίζεται στο μέλλον είναι ο συνδυασμός XSLT και XHTML, ως εξέλιξη της απλής HTML.

2.6.2.5 DOM/SAX

Για την ανάγνωση και επεξεργασία των εγγράφων XML μέσα από μια εφαρμογή έχουν αναπτυχθεί κυρίως δυο API (Application Program Interfaces) τα DOM και SAX.

Το DOM έχει συσταθεί και υποστηρίζεται σαν πρότυπο από το W3C (World Wide Web Consortium). Διαθέτει επίσης ευρεία υποστήριξη από πολλά προγράμματα φυλλομέτρησης του Ιστού (Web Browsers). Το DOM εκτός από ανάγνωση προσφέρει δυνατότητες και για τροποποίηση υπαρχόντων ή για δημιουργία νέων XML εγγράφων.

Το SAX (Simple API for XML). βασίζεται σε μια λογική συμβάντων (events) και χειριστών συμβάντων (events hndlers) και παρέχει στον προγραμματιστή λειτουργίες χαμηλότερου επιπέδου. Το SAX, σε αντίθεση με το DOM, δεν υποστηρίζεται από καμία επίσημη τυποποίηση, αλλά χρησιμοποιείται ευρύτατα και θεωρείται ως μια de facto τυποποίηση. Παρακάτω αναλύουμε περαιτέρω τα χαρακτηριστικά του DOM API, διότι αυτό είναι το δημοφιλέστερο από τα δύο διαθέσιμα XML API.

Το DOM είναι αντικειμενοστρεφές και μπορεί να ενσωματωθεί σε ένα μεγάλο αριθμό από προγραμματιστικά περιβάλλοντα τόσο με γλώσσες προγραμματισμού όσο και με γλώσσες script. Το DOM αποτελείται από ένα σύνολο αντικειμένων που διαθέτουν ιδιότητες και μεθόδους για επεξεργασία των εγγράφων XML.

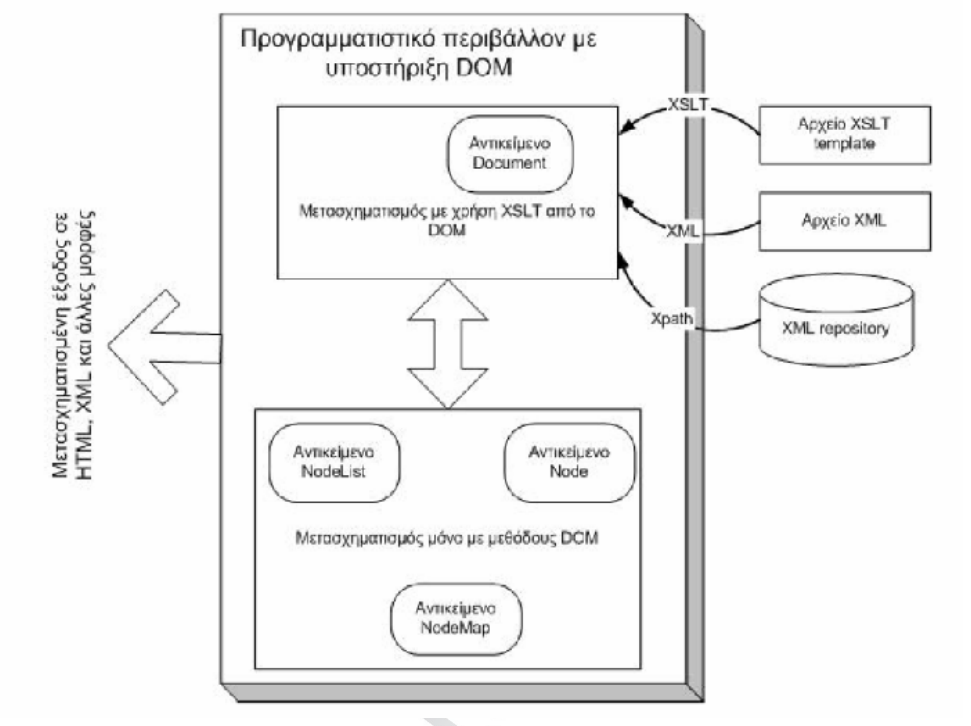
Τα βασικά αντικείμενα που διαθέτει το DOM είναι τα ακόλουθα:

- **Document:** Μας επιτρέπει να διαβάζουμε αρχεία XML, να εκτελούμε αναζητήσεις μέσα σε έγγραφα XML και να μετασχηματίζουμε έγγραφα χρησιμοποιώντας XSLT templates.
- **Node:** Μας επιτρέπει να χειριζόμαστε μεμονωμένα το υπόδεντρο ενός κόμβου μέσα στην ιεραρχία ενός εγγράφου XML και να επεξεργαζόμαστε, ανανεώνουμε, προσθέτουμε, διαγράφουμε και να αναζητούμε υποκόμβους και πόρους αυτών.
- **Node List:** Αυτό το αντικείμενο είναι μια συλλογή (collection) από αντικείμενατύπου Node.
- **Node Map:** Το αντικείμενο αυτό χειρίζεται τα attributes για μια συλλογή από κόμβους.

Η επεξεργασία ενός εγγράφου XML από το DOM περιλαμβάνει τυπικά τρία στάδια, τα οποία είναι τα ακόλουθα:

- **Εισαγωγή:** Αρχικά εισάγουμε το περιεχόμενο XML που θέλουμε να χειριστούμε σε ένα αντικείμενο Document. Το περιεχόμενο αυτό μπορεί να βρίσκεται είτε σε αρχεία XML είτε σε αρχεία που περιέχουν ένα test string, που όμως είναι καλά δομημένο XML περιεχόμενο (well formed). Παράλληλα με τα έγγραφα XML εισάγονται και τυχόν υπάρχοντα XSLT (το δομικό στοιχείο σε ένα XSLT είναι το elements <xls:stylesheet>) έγγραφα, τα οποία έχουν την ίδια σύνταξη με τα XML έγγραφα. Επίσης το περιεχόμενο μπορεί να ανακτηθεί από ένα XML repository ή μια βάση δεδομένων, χρησιμοποιώντας το Xpath.
- **Επεξεργασία:** Εδώ έχουμε δύο επιλογές. Η πρώτη επιλογή είναι να δημιουργήσουμε ένα αντικείμενο Document για το έγγραφο XML και ένα άλλο αντικείμενο Document για το XSLT έγγραφο. Μετά με εντολές του DOM εφαρμόζουμε τον XSLT μετασχηματισμό. Η δεύτερη επιλογή είναι να φορτώσουμε μόνο ένα αντικείμενο Document για το έγγραφο XML και μετά να το χειριστούμε αποκλειστικά με μεθόδους επεξεργασίας που προσφέρει το DOM χωρίς την χρήση του XSLT.
- **Εξαγωγή*:*** Η έξοδος μπορεί να είναι είτε HTML, XML (για μετατροπή από ένα XML Schema σε ένα άλλο) είτε μια σχεσιακή βάση. Στην περίπτωση της βάσης δεδομένων γίνεται αντιστοίχηση από το XML περιεχόμενο σε εγγραφές και πεδία και κατόπιν εισαγωγή του περιεχομένου στην βάση.

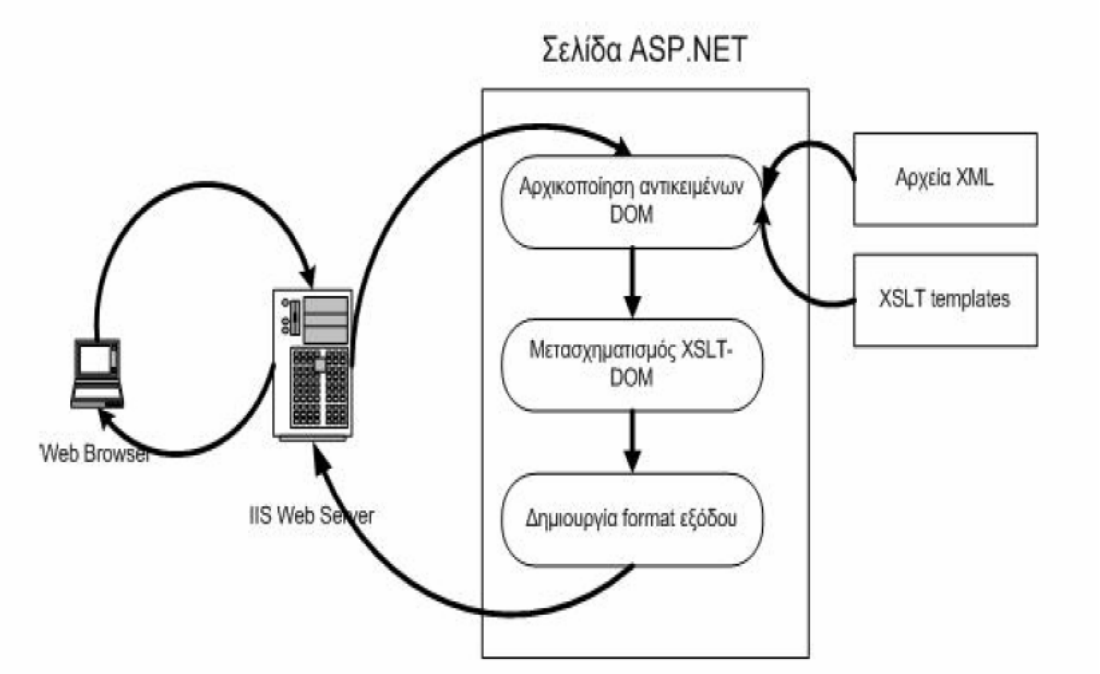
Πέραν των παραπάνω μορφών εξόδου μπορούν να παραχθούν βεβαίως και άλλες μορφές, ανάλογα με τις εκάστοτε ανάγκες. Τα παραπάνω τρία στάδια της επεξεργασίας εγγράφων XML από το DOM φαίνονται στο παρακάτω σχήμα:



Εικόνα 5:Επεξεργασία DOM με και χωρίς XSLT

Το DOM ακολουθεί επίσης τις αρχές του δικτυακού προγραμματισμού client-server και συνεπώς διαθέτει συστατικά στοιχεία και στον client (browser) και στον server. Πιο συγκεκριμένα:

Server: Κάθε ενέργεια που γίνεται σε έναν server είναι ένας XML μετασχηματισμός. Για τον μετασχηματισμό αυτόν χρησιμοποιείται το DOM API με εισόδους τα απαιτούμενα XML και XSLT έγγραφα καθώς και είσοδο του τελικού χρήστη μέσω φορμών. Η διαδικασία αυτή για την περίπτωση ενός περιβάλλοντος που χρησιμοποιεί εργαλεία Microsoft περιγράφεται ικανοποιητικά από το παρακάτω σχήμα.



Εικόνα 6:Επεξεργασία DOM στην πλευρά του Microsoft IIS server

Client: Αν και στα μεγάλα συστήματα CMS ο περισσότερος κώδικας XML βρίσκεται στον server, μπορούμε σε πολλές περιπτώσεις να δουλέψουμε εναλλακτικά με επεξεργασία XML και στον client.

Έχουμε τις εξής δύο επιλογές:

- Μπορούμε να βασιστούμε στην αυτοματοποιημένη διαδικασία που προσφέρει ο browser για επεξεργασία των εγγράφων XML. Για παράδειγμα ο Mozilla Firefox όταν του ζητηθεί ένα έγγραφο XML ελέγχει αν αυτό διαθέτει αναφορά σε κάποιο XSLT. Αν ναι τότε προβάλλεται αυτόματα η HTML σελίδα μετά την μετατροπή. Αν δεν υπάρχει αναφορά σε XSLT τότε προβάλλεται το XML έγγραφο με δενδρική δομή.
- Εκτός από τις ενσωματωμένες δυνατότητες του προγράμματος περιήγησης μπορούμε να γράψουμε τα δικά μας προγράμματα script (για παράδειγμα JavaScript) για να φορτώσουμε τα XML και XSLT και μετά να εκτελέσουμε εντολές DOM χειρωνακτικά. Αυτό προσφέρει μεγαλύτερη ευελιξία και δυνατότητες παραμετροποίησης της τελικής εμφάνισης των εγγράφων μας.

2.6.3 SOAP (Simple Object Transfer Protocol) –REST (Representational State Transfer)

2.6.3.1. SOAP

Στα τέλη της δεκαετίας του 90, και συγκεκριμένα το 1997, μεγάλες εταιρείες, πως η Microsoft, άρχισαν να διερευνούν κατά πόσο ο κατανεμημένος υπολογισμός μπορεί να βασιστεί στη γλώσσα XML. Ο σκοπός της έρευνας αυτής ήταν να γίνει εφικτή η επικοινωνία μεταξύ των εφαρμογών μέσω απομακρυσμένων κλήσεων διαδικασιών (Remote Procedure Calls – RPCs), χρησιμοποιώντας απλά πρωτόκολλα δικτύου, όπως το HTTP. Το 1999 έκανε την εμφάνισή του το SOAP, ένας RPC μηχανισμός βασισμένος σε XML. Το 2000 ο οργανισμός W3C ασχολείται με την ιδέα αυτή και ύστερα από αρκετές αλλαγές, βελτιώσεις και τροποποιήσεις 2 ολόκληρων χρόνων, το 2003 δηλαδή, το SOAP με την έκδοση 1.2 γίνεται η προτεινόμενη προδιαγραφή – πρωτόκολλο για τις υπηρεσίες διαδικτύου.

Στον πυρήνα του, το SOAP, είναι μια προδιαγραφή για ένα απλό αλλά ταυτόχρονα ευέλικτο XML πρωτόκολλο δεύτερης γενιάς. Το σημαντικό είναι ότι καθώς η έρευνα ξεκίνησε από τον κατανεμημένο υπολογισμό, το SOAP είναι το πλέον κατάλληλο πρωτόκολλο αφού ειδικεύεται σε τέτοια περιβάλλοντα. Το SOAP παρέχει τα κάτωθι χαρακτηριστικά και μηχανισμούς:

- **Μηχανισμός για τον ορισμό της πληροφορίας στην επικοινωνία:** Στο SOAP, όλη η πληροφορία είναι καταχωρημένη σε ένα ξεκάθαρο και ταυτοποιήσιμο SOAP μήνυμα (SOAP message). Αυτό γίνεται μέσω ενός SOAP φακέλου (SOAP envelope), που περικλείει όλες τις απαραίτητες πληροφορίες. Ένα μήνυμα SOAP, μπορεί να έχει σώμα (body), το οποίο περιέχει πληροφορία σε XML δομή. Επίσης, μπορεί να διαθέτει έναν αριθμό από επικεφαλίδες (headers), οι οποίες ενσωματώνουν επιπλέον πληροφορίες έξω από το σώμα του μηνύματος.
- **Διεργασιακό μοντέλο:** Αυτό το μοντέλο ορίζει ένα σύνολο κανόνων βάσει των οποίων γίνονται οι διαπραγματεύσεις μεταξύ των SOAP μηνυμάτων και του λογισμικού. Διακρίνεται από την απλότητά του.
- **Μηχανισμός για αντιμετώπιση σφαλμάτων:** Το SOAP παρέχει το μηχανισμό αυτό με τη μορφή των SOAP faults, τα οποία όταν χρησιμοποιούνται μπορεί να προσδιοριστεί η πηγή που προκάλεσε το σφάλμα. Επιπλέον, παρέχουν τη

δυνατότητα ανταλλαγής διαγνωστικών πληροφοριών μεταξύ των μελών της επικοινωνίας.

- **Μοντέλο επεκτασιμότητας:** Το μοντέλο αυτό χρησιμοποιεί SOAP επικεφαλίδες για την υλοποίηση προεκτάσεων πάνω στο SOAP. Οι επικεφαλίδες περιέχουν κομμάτια από δεδομένα με δυνατότητα επέκτασης, τα οποία ταξιδεύουν μαζί με το μήνυμα και μπορούν να γίνουν στόχος για επέκταση σε συγκεκριμένους κόμβους του δικτύου.
- **Ευέλικτος μηχανισμός για αναπαράσταση δεδομένων:** Ο μηχανισμός αυτός επιτρέπει στα δεδομένα σε οποιαδήποτε σειριακή μορφή κι αν βρίσκονται να αναπαρασταθούν σε XML μορφή.
- **Σύμβαση για αναπαράσταση των RPCs και των απαντήσεών τους σαν SOAP μηνύματα:** Οι απομακρυσμένες κλήσεις διαδικασιών είναι αρκετά διαδεδομένες στον κατανεμημένο προγραμματισμό/ υπολογισμό και μπορούν να αναπαρασταθούν καλά μέσω του μηχανισμού αυτού σαν SOAP μηνύματα.
- **Πρωτόκολλο εγκαθίδρυσης σύνδεσης:** Το πρωτόκολλο αυτό ορίζει μια αρχιτεκτονική για την οικοδόμηση συνδέσεων επικοινωνίας ώστε να είναι εφικτή η ανταλλαγή SOAP μηνυμάτων πάνω σε μέσα μεταφοράς και επικοινωνιακά κανάλια. Χρησιμοποιείται το HTTP πρωτόκολλο, καθώς είναι το πιο διαδεδομένο και ευρέως χρησιμοποιούμενο στο διαδίκτυο.

Όσον αφορά στα μηνύματα SOAP, εφόσον το πρωτόκολλο είναι βασισμένο στην XML, είναι αναμενόμενο τα μηνύματα να έχουν μια τέτοια μορφή και ουσιαστικά να μην είναι τίποτε άλλο, παρά XML έγγραφα. Το στοιχείο-ρίζα του μηνύματος είναι το soapenv:Envelope, το οποίο περικλείει το στοιχείο soapenv:Body, που περιέχει πληροφορία σχετιζόμενη με το σκοπό του μηνύματος. Το στοιχείο soapenv:Body με τη σειρά του μπορεί να περιέχει άλλα στοιχεία, τα οποία να σχετίζονται ή να αναπαριστούν την απομακρυσμένη κλήση διαδικασίας που πρόκειται να εκτελεστεί. Το όνομα της μεθόδου που θα κληθεί και το όνομα του στοιχείου αυτού περιέχονται μέσα στο σώμα του μηνύματος.

Τα μηνύματα αυτά στέλνονται μέσω πρωτοκόλλου HTTP με τη μέθοδο POST. Η απάντηση σε ένα τέτοιο μήνυμα, που και πάλι μεταφέρεται μέσω HTTP είναι και αυτή ένα SOAP μήνυμα. Το μήνυμα της απάντησης περικλείεται κι αυτό από το συστατικό soapenv:Envelope, που περιέχει το soapenv:Body, το οποίο ενσωματώνει και τη βασική πληροφορία του μηνύματος,

στην οποία συγκαταλέγεται και μια κωδικοποιημένη αναπαράσταση του αποτελέσματος της απομακρυσμένης κλήσης που πραγματοποιήθηκε.

Το SOAP παρέχει, όπως ήδη αναφέρθηκε, τη δυνατότητα σε ένα μήνυμα να συμπεριλαμβάνονται στοιχεία – επικεφαλίδες. Ο ρόλος τους είναι και ο σημαντικότερος λόγος που χρησιμοποιούμε τα μηνύματα SOAP και όχι απλά XML έγγραφα. Τα στοιχεία αυτά (headers) έχουν την ιδιότητα να αναπαριστούν ένα επιπρόσθετο και επεκτάσιμο είδος πληροφορίας το οποίο μεταφέρεται μαζί με το υπόλοιπο μήνυμα, χωρίς να τροποποιείται ο κυρίως πυρήνας του μηνύματος. Για να γίνει κατανοητό αυτό ας δούμε ένα παράδειγμα της αληθινής ζωής ακριβώς αντίστοιχο με την ιδέα των SOAP μηνυμάτων.

Ας υποθέσουμε ότι θέλουμε να στείλουμε ένα έγγραφο αλλά και κάποιες επιπρόσθετες πληροφορίες χωρίς όμως να θέλουμε να μαρκάρουμε το έγγραφο με αυτές. Τοποθετούμε, λοιπόν, το έγγραφο στο φάκελο και στη συνέχεια προσθέτουμε μία ή δύο σελίδες χαρτί, οι οποίες περιγράφουν την επιπλέον πληροφορία που θέλαμε να στείλουμε. Η αντιστοιχία του παραδείγματος αυτού με το πρωτόκολλο που περιγράφουμε είναι ότι ο φάκελος είναι το στοιχείο soapenv:Envelope, το έγγραφο που θέλουμε να στείλουμε είναι το στοιχείο soapenv:Body και οι σελίδες με την επιπλέον πληροφορία είναι τα στοιχεία επικεφαλίδες (headers).

Η χρήση των επικεφαλίδων για την προσθήκη λειτουργικότητας σε ένα μήνυμα SOAP είναι γνωστή και σαν κατακόρυφη επέκταση, γιατί τα στοιχεία αυτά τοποθετούνται στην κορυφή του μηνύματος πριν το σώμα του.

Για λόγους ασφάλειας σε περιπτώσεις επεκτασιμότητας χρησιμοποιώντας στοιχεία επικεφαλίδες, η προδιαγραφή του SOAP ορίζει τη μεταβλητή mustUnderstand. Όταν η τιμή της είναι αληθής, τότε ο παραλήπτης του μηνύματος πρέπει να συμφωνήσει ότι αποδέχεται όλους τους όρους της επικεφαλίδας του μηνύματος προκειμένου να συνεχιστεί η επεξεργασία του. Στην περίπτωση που η τιμή της μεταβλητής αυτή είναι ψευδής, τότε η επικεφαλίδα έχει προαιρετικό ρόλο και η επεξεργασία του μηνύματος μπορεί να γίνει και από παραλήπτες που αγνοούν τις προδιαγραφές που θέτει η επικεφαλίδα.

Αντίστοιχα με την κατακόρυφη επεκτασιμότητα, υπάρχει και η οριζόντια, η οποία έχει να κάνει με την αντιστοίχηση διαφορετικών κομματιών του μηνύματος με διαφορετικούς παραλήπτες. Αυτό επιτυγχάνεται μέσω των διαμεσολαβητών SOAP (SOAP intermediaries). Αυτές οι εφαρμογές μπορούν να επεξεργάζονται μέρη του μηνύματος SOAP καθώς αυτό ταξιδεύει προς τον τελικό του προορισμό. Οι μεσολαβητές μπορούν να αποδέχονται και να

προωθούν μηνύματα καθώς κάνουν και κάποιο είδος επεξεργασίας. Σε γενικές γραμμές, στον αληθινό κόσμο σπάνια ξεκινά ένα μήνυμα από κάποια τοποθεσία και φθάνει απευθείας στον προορισμό του.

Για παράδειγμα ας πάρουμε το ηλεκτρονικό ταχυδρομείο. Όταν στέλνουμε ένα μήνυμα αυτό θα περάσει από διάφορους εξυπηρετητές και δρομολογητές προτού «εντοπίσει» τον προορισμό του. Αυτός είναι ένας από τους κυριότερους λόγους που χρειάζονται οι μεσολαβητές στα μηνύματα SOAP, καθώς επίσης χάρη σε αυτούς μπορούμε να έχουμε υπηρεσίες προστιθέμενης αξίας σε κατανεμημένα συστήματα.

Έχοντας περιγράψει αρκετά το SOAP, φτάνουμε στα εξής βήματα που τελικά πρέπει να πραγματοποιηθούν από έναν παραλήπτη ή ένα μεσολαβητή κάποιου μηνύματος SOAP:

- Πρέπει να κατανοηθεί το σύνολο των κανόνων που περιέχει το μήνυμα που παραλείφθηκε. Τα στοιχεία επικεφαλίδες κυρίως μπορεί να περιέχουν τέτοια πληροφορία.
- Πρέπει να προσδιοριστούν όλα τα σύνολα που ορίζουν τα στοιχεία επικεφαλίδες με προσοχή.

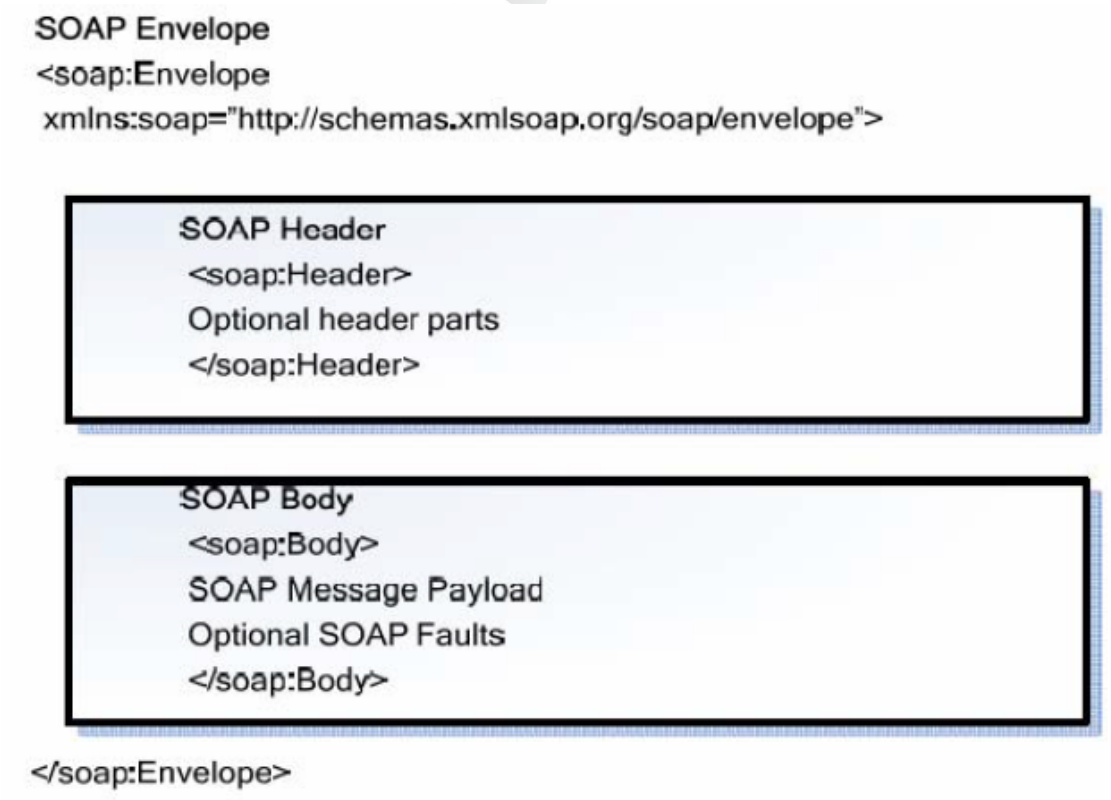
Αν κάποιο σημείο δεν είναι κατανοητό – και αυτό έχει να κάνει με τη μεταβλητή mustUnderstand – τότε πρέπει να συνταχθεί ένα μήνυμα λάθους (SOAP fault). Σε τέτοια περίπτωση η περαιτέρω επεξεργασία του μηνύματος πρέπει να σταματήσει. Λάθη που έχουν να κάνουν με το σώμα του μηνύματος δεν εντοπίζονται σε αυτό το βήμα.

Ύστερα, ακολουθεί η επεξεργασία των επικεφαλίδων. Σε περίπτωση που το μήνυμα βρίσκεται στον τελικό παραλήπτη και όχι σε κάποιον ενδιάμεσο, τότε γίνεται και η επεξεργασία του σώματος του μηνύματος. Στην περίπτωση που το μήνυμα βρίσκεται σε κάποιον ενδιάμεσο μεσολαβητή και δεν υπάρχει κάποιο πρόβλημα – με την έννοια ότι δεν υπήρξε ανάγκη να συνταχθεί κάποιο μήνυμα λάθους – τότε η πορεία του μηνύματος συνεχίζεται με την προώθησή του στον επόμενο κόμβο, όπου ακολουθείται η ίδια διαδικασία.

Ουσιαστικά, η μεταβλητή mustUnderstand έχει ως βασικό ρόλο να μας δίνει την ευκαιρία να ορίζουμε τις ακριβείς διαδικασίες που θα γίνονται σε κάθε κόμβο από τον οποίο περνάει το μήνυμα μέχρι να φτάσει στον τελικό του προορισμό.

Όσον αφορά στα μηνύματα λάθους, αυτά είναι απλά μηνύματα SOAP, τα οποία περιέχουν ένα απλό στοιχείο μέσα στο soapenv:Body, το soapenv:Fault. Η παρουσία αυτού του στοιχείου σηματοδοτεί ότι κάπου υπήρξε κάποιο λάθος. Το μήνυμα λάθους έχει κάποια συγκεκριμένη και καλά ορισμένη δομή ώστε να παρέχει πληροφορία σχετική με το λάθος που συνέβη. Το στοιχείο env:Code περιέχει όλη την απαραίτητη κωδικοποιημένη πληροφορία για το σκοπό αυτό. Από τις πληροφορίες αυτές μπορούμε να διαπιστώσουμε εάν το λάθος προέκυψε λόγω λανθασμένης ή ελλιπούς πληροφορίας από τον αποστολέα, ή λόγω κάποιου λάθους που συνέβη κατά την επεξεργασία του από κάποιον παραλήπτη. Επίσης το λάθος μπορεί να σχετίζεται με την σχέση κάποιου μεσολαβητή με την τιμή της μεταβλητής mustUnderstand. Τέλος, ο λόγος προελεύσεώς του μπορεί να είναι οι μη συμβατές εκδόσεις του πρωτοκόλλου που χρησιμοποιούνταν από κόμβο σε κόμβο. Τα μηνύματα λάθους, ως κανονικά SOAP μηνύματα που είναι, μπορούν να επεκτείνονται με την παρουσία στοιχείων επικεφαλίδων, τα οποία μπορεί να περιέχουν επιπλέον πληροφορία για την ανίχνευση του λάθους και να συμβάλλουν στη συντήρηση των υπηρεσιών πάνω στο δίκτυο.

Ένα εργαλείο για την πραγματοποίηση SOAP συναλλαγών είναι ο Apache Axis, μια μηχανή SOAP ανοιχτού λογισμικού. Η δομή ενός μηνύματος SOAP παρουσιάζεται στο παρακάτω σχήμα.

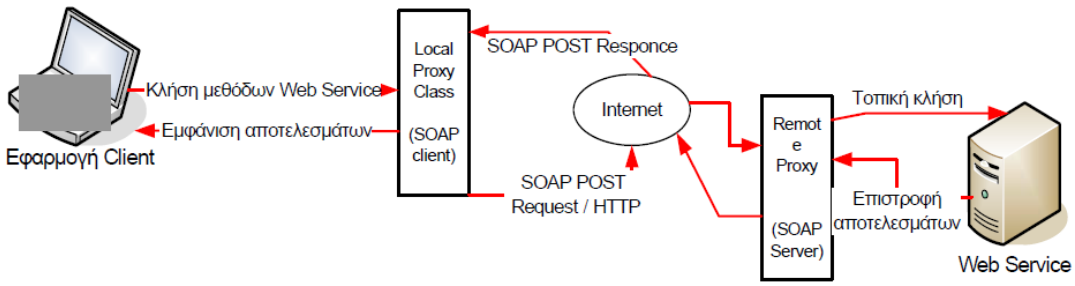


Εικόνα 7:Δομή μηνύματος SOAP

Το πρωτόκολλο SOAP περιλαμβάνει επίσης ένα σύνολο από κανόνες για το πώς θα γίνεται η κωδικοποίηση κάθε διαφορετικού τύπου δεδομένων που αποστέλλεται σε μία εφαρμογή. Η ενσωμάτωση αυτών των κανόνων στο μήνυμα SOAP γίνεται με το χαρακτηριστικό encodingStyle, το οποίο παραπέμπει σε ένα URL με τους κανόνες κωδικοποίησης. Υποστηρίζονται τόσο απλοί όσο και σύνθετοι τύποι δεδομένων.

Τα μηνύματα SOAP υλοποιούν κλήσεις RPC (Remote Procedure Call). Πιο συγκεκριμένα, για την κλήση μιας μεθόδου ενός απομακρυσμένου αντικειμένου μιας Διαδικτυακής Υπηρεσίας, στέλνεται ένα μήνυμα SOAP που περιλαμβάνει στο σώμα του το όνομα της μεθόδου, τις παραμέτρους που αυτή πρέπει να δεχθεί και το URLT της.

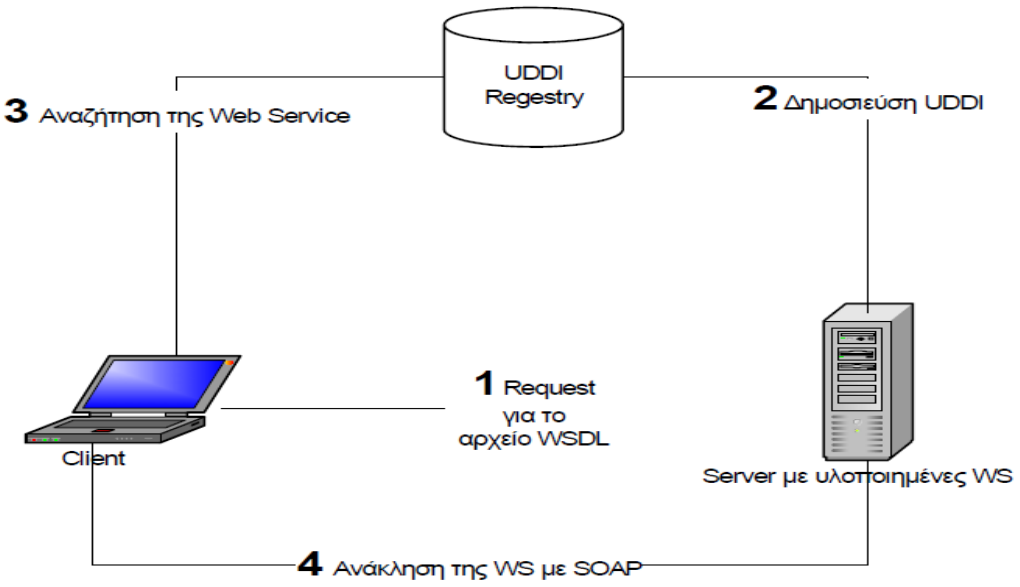
Μετά την εκτέλεση της μεθόδου από την απομακρυσμένη Υπηρεσία, τα αποτελέσματα επιστρέφονται στο πρόγραμμα που τα έχει καλέσει μέσω ενός άλλου μηνύματος SOAP. Η όλη επικοινωνία πραγματοποιείται μεταξύ του πελάτη (SOAP client) και του διακομιστή (SOAP server), οι οποίοι είναι τμήματα προγράμματός που αναλαμβάνουν την μετατροπή από την μορφή που είναι κατανοητή στην τοπική εφαρμογή ή υπηρεσία αντίστοιχα σε μορφή SOAP XML envelope και το αντίστροφο. Λειτουργούν έτσι ως μεσάζοντες (proxy) μεταξύ των δυο επικοινωνούντων μηχανών. Για να αντιληφθούμε καλύτερα την παραπάνω αμφίδρομη διαδικασία, μπορούμε να παρατηρήσουμε την εικόνα 8



Εικόνα 8:Κλήση των Web Services

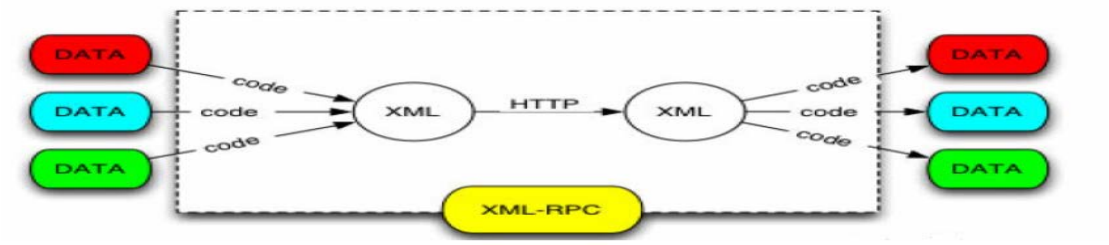
Μια πρόσθετη χρήσιμη λειτουργία του SOAP server είναι ότι παράγει το αρχείο περιγραφής της Διαδικτυακής Υπηρεσίας (WSDL) και το δημοσιεύει σε κάποιο ειδικό μητρώο στο Διαδίκτυο (UDDI Registry). Όταν η εφαρμογή client θέλει να καλέσει ένα Web Service, απλά αναζητά στο μητρώο UDDI την υπηρεσία που επιθυμεί, ανακτά το αρχείο WSDL και με βάση

αυτό διαμορφώνει κατάλληλα το SOAP μήνυμα που πρέπει να αποσταλεί στην υπηρεσία. Αυτό γίνεται περισσότερο κατανοητό στην εικόνα 9. Οι αριθμοί συμβολίζουν την χρονική σειρά των βημάτων που ακολουθούνται.



Εικόνα 9:Αναζήτηση σε UDDI

Για την μετάδοση των SOAP μηνυμάτων μπορεί να γίνει χρήση πολλών πρωτοκόλλων, π .χ SMTP ή HTTPS ή JMS (Java Messaging Service), αρκεί τα πρωτόκολλα αυτά να υποστηρίζονται και από την καλούσα και από την καλούμενη εφαρμογή. Το πιο διαδεδομένο πρωτόκολλο που χρησιμοποιείται στην πράξη είναι το HTTP. Το πρωτόκολλο SOAP χρησιμοποιεί την μέθοδο POST του HTTP για την αποστολή μηνυμάτων. Μια επικοινωνία τύπου RPC αποτελείται από ζεύγη μηνυμάτων SOAP POST Request / POST Response μήνυμα περιέχεται ένα tag <Result> το οποίο έχει σαν τιμή το αποτέλεσμα της κλήσης της μεθόδου και ένα αλφαριθμητικό (String) Unicode.



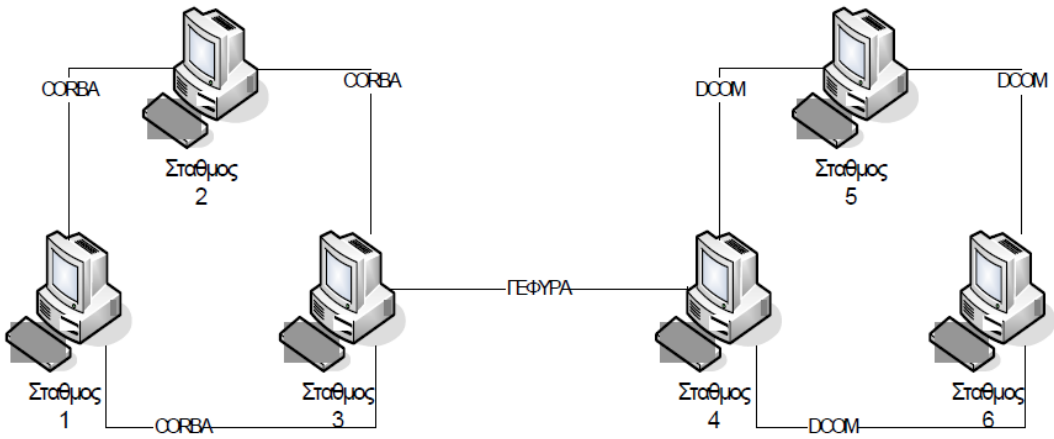
Σχήμα: XML – RPC

Εικόνα 10:XML – RPC

2.6.3.1.1. Απόδοση του SOAP

Τα πρωτόκολλα CORBA και DCOM χρησιμοποιούν δυαδική κωδικοποίηση (binary encoding) για τα μηνύματα που ανταλλάσσουν και θεωρούν δεδομένη την συμφωνία των δυο επικοινωνούντων κόμβων ως προς συμβάσεις σχετικά με το νόημα των δεδομένων που ανταλλάσσονται. Αντίθετα το SOAP χρησιμοποιεί κωδικοποίηση XML στα μηνύματα του για να προσδώσει νόημα στα ανταλλασσόμενα δεδομένα. Το γεγονός αυτό το καθιστά εύκολο στην διαχείριση, διαλειτουργικό και επεκτάσιμο, ωστόσο υστερεί κάπως σε επιδόσεις, συγκριτικά με τα COBRA/DCOM. Οι τυπικές επιδόσεις που έχουν μετρηθεί για το πρωτόκολλο SOAP είναι 500 μηνύματα/sec όταν είναι σε διαφορετικά μηχανήματα.

Το μεγάλο πλεονέκτημα της συμβατότητας του SOAP φαίνεται στα παρακάτω σχήματα.



Εικόνα 11:Ασύμβατα COBRA / DCOM δίκτυα που απαιτούν γέφυρα για να

Όπως φαίνεται από τα παραπάνω σχήματα, το μεγάλο πλεονέκτημα του SOAP και των Web Services είναι ότι μπορούν να διασυνδέουν συστήματα που χρησιμοποιούν διαφορετικά middleware σε μια ενιαία πλατφόρμα επικοινωνίας. Η πλατφόρμα αυτή είναι ανεξάρτητη από το λογισμικό που τρέχει σε κάθε ένα σύστημα. Η συμβατότητα είναι ένα πολύ σημαντικό χαρακτηριστικό του SOAP. Οι υπάρχουσες υποδομές σε middleware δεν είναι απαραίτητο να καταργηθούν.

SOAP over http

Τα μηνύματα SOAP μπορούν να μεταφερθούν μέσα στο δίκτυο με διάφορα πρωτόκολλα, όπως το SMTP και το FTP, αλλά κατά κανόνα προτιμάται το HTTP (HyperText Transfer Protocol). Ο λόγος είναι ότι το HTTP είναι το πρωτόκολλο που χρησιμοποιείται ευρέως στο Διαδίκτυο (internet), καθώς χρησιμοποιείται από τους browsers για την πλοήγηση στον παγκόσμιο ιστό. Τα δεδομένα μεταφέρονται ακριβώς όπως οι ιστοσελίδες και δεν εμποδίζονται από τείχη προστασίας (firewalls) που ενδέχεται να είναι ενεργοποιημένα στους υπολογιστές.

Το μήνυμα SOAP αποστέλλεται σαν μέρος του μηνύματος αίτησης ή απόκρισης HTTP (HTTP request ή HTTP response, αντίστοιχα) και προστίθενται στο μήνυμα και οι επικεφαλίδες HTTP (HTTP headers). Γενικά, το πρωτόκολλο HTTP χρησιμοποιεί δύο τρόπους για να στέλνει δεδομένα : HTTP – POST και HTTP – GET, με τον πρώτο να προτιμάται για τις Διαδικτυακές υπηρεσίες, διότι σύμφωνα με αυτόν το μήνυμα SOAP μεταφέρεται στο κυρίως σώμα (body) του μηνύματος HTTP.

SOAP Exceptions and Errors

Οι κανόνες κωδικοποίησης SOAP (SOAP encoding rules) παρέχουν στο πρότυπο SOAP τη δυνατότητα χειρισμού λαθών, τα οποία μπορεί να προκύψουν τόσο κατά τη μεταφορά όσο κατά την επεξεργασία του μηνύματος. Στην πρώτη κατηγορία περιλαμβάνονται η μη αναγνωσιμότητά του μηνύματος και η αλλοίωση κάποιων χαρακτήρων του μηνύματος, ενώ στη δεύτερη κατατάσσονται οι λανθασμένες τιμές σε κάποια πεδία της επικεφαλίδας (SOAP header) ή των δεδομένων που αποστέλλονται, η μη ταυτοποίηση του αποστολέα και τα λάθη κατά τη σύνδεση με τη βάση δεδομένων.

Οι πληροφορίες λάθους περιέχονται εντός του κυρίως σώματος (SOAP Body) στο στοιχείο XML <fault>, το οποίο αποτελείται από τα υποχρεωτικά στοιχεία : <faultcode> που προσδιορίζει πού συνέβη το λάθος, <faultstring> το οποίο περιγράφει πώς δημιουργήθηκε λάθος, και τα προαιρετικά στοιχεία : <faultactor> και <detail> που παρέχουν περαιτέρω λεπτομέρειες.

Πιο συγκεκριμένα, ανάλογα με την τιμή του <faultcode> διακρίνονται δύο γενικές κατηγορίες σφαλμάτων :

- **<soap:Server>** που αναφέρεται σε λάθη για τα οποία δεν ευθύνεται το μήνυμα που έστειλε ο πελάτης της διαδικτυακής υπηρεσίας και δεν μπορούν να διορθωθούν με ενέργειες του client. Εάν αποσταλεί το ίδιο μήνυμα αργότερα, ενδέχεται να επιτύχει η υπηρεσία.
- **<soap:Client>** δηλαδή λάθη τα οποία οφείλονται στο περιεχόμενο του μηνύματος που έστειλε ο client και μπορεί να βρίσκονται στην επικεφαλίδα ή / και στο κυρίως μήνυμα.

SOAP Extensions

Οι προεκτάσεις SOAP (SOAP Extensions) είναι ένα είδος component που επισυνάπτεται σε μια διαδικτυακή εφαρμογή που ανταλλάσσει μηνύματα SOAP με κάποια εξωτερική οντότητα και μπορεί να εφαρμόζεται σε κάθε SOAP μήνυμα που λαμβάνεται και αποστέλλεται ή σε κάποια SOAP μηνύματα επιλεκτικά. Ανάλογα με τη λειτουργία που επιτελεί η προέκταση καθορίζεται εάν απαιτείται να τοποθετηθεί και από την πλευρά της εξωτερικής οντότητας. Η διαδικασία που ακολουθείται έχει ως εξής: οι προεκτάσεις SOAP (SOAP Extensions) παραλαμβάνουν τα εισερχόμενα μηνύματα SOAP και τα προ-επεξεργάζονται προτού τα παραδώσουν στον χειριστή ASMX για να τα επεξεργαστεί. Μόλις δημιουργηθεί μια απόκριση στην αίτηση – εξερχόμενο μήνυμα SOAP προωθείται στην επέκταση SOAP για περαιτέρω επεξεργασία.

Η χρήση των προεκτάσεων SOAP (SOAP Extensions) συνοψίζεται στους ακόλουθους λόγους:

- Ενίσχυση της ασφάλειας με κρυπτογράφηση και αποκρυπτογράφηση του μηνύματος SOAP.
- Συμπίεση και αποσυμπίεση του μηνύματος SOAP.
- Μετατροπή των μηνυμάτων που προορίζονται για παλιότερες εκδόσεις στην τρέχουσα έκδοση της υπηρεσίας.
- Λεκτική ανάλυση των επικεφαλίδων SOAP για ταυτοποίηση και εξουσιοδότηση του πελάτη.

- Έλεγχος εάν τα μηνύματα SOAP ικανοποιούν τα κατάλληλα XML σχήματα.
- Επιβολή επιχειρηματικών κανόνων στις τιμές εισόδου και εξόδου.

2.6.3.1.2. Γιατί να χρησιμοποιήσει κανείς το SOAP;

Είναι πολύ σημαντικό η ανάπτυξη εφαρμογών να επιτρέπει την επικοινωνία των προγραμμάτων μέσα από το Internet. Οι σημερινές εφαρμογές επικοινωνούν μεταξύ τους χρησιμοποιώντας απομακρυσμένες κλήσεις μεθόδων (**Remote Procedure Calls** ή **RPC**), αλλά το HTTP δεν σχεδιάστηκε για κάτι τέτοιο. Το RPC εμπεριέχει δυο προβλήματα:

- ένα πρόβλημα συμβατότητας και
- ένα πρόβλημα ασφάλειας, αφού λογικά τα firewalls και οι proxy servers πρέπει να μπλοκάρουν αυτού του είδους την κίνηση.

Ένας καλύτερος τρόπος για την επικοινωνία μεταξύ εφαρμογών είναι πάνω από το HTTP, επειδή το HTTP υποστηρίζεται από όλους τους Internet browsers και servers.

Το SOAP σχεδιάστηκε για να επιτύχει κάτι τέτοιο. Το SOAP προσφέρει έναν τρόπο για την επικοινωνία μεταξύ εφαρμογών που τρέχουν σε διαφορετικά λειτουργικά συστήματα, με διαφορετικές τεχνολογίες και διαφορετικές προγραμματιστικές γλώσσες.

2.6.3.1.3. Κανόνες σύνταξης ενός SOAP μηνύματος

Παρακάτω αναφέρουμε μερικούς βασικούς κανόνες σύνταξης ενός μηνύματος SOAP:

- Ένα SOAP μήνυμα ΠΡΕΠΕΙ να γράφεται χρησιμοποιώντας XML
- Ένα SOAP μήνυμα ΠΡΕΠΕΙ να χρησιμοποιεί το SOAP Envelope namespace
- Ένα SOAP μήνυμα ΠΡΕΠΕΙ να χρησιμοποιεί το SOAP Encoding namespace
- Ένα SOAP μήνυμα ΔΕΝ πρέπει να χρησιμοποιεί αναφορά σε DTD
- Ένα SOAP μήνυμα ΔΕΝ πρέπει να περιέχει XML Processing Instructions

2.6.3.2. REST

Η μέθοδος αυτή που χρησιμοποιεί το πρωτόκολλο HTTP για τη μεταφορά των XML μηνυμάτων ονομάζεται REST. Η λέξη REST αποτελεί το ακρωνύμιο για το Representational State Transfer. Το REST δεν αποτελεί κάποιο πρότυπο. Για αυτό και οργανισμοί όπως ο W3C δεν έχουν εκδώσει κάποια τυποποίηση για αυτό. Το REST αποτελεί μια αρχιτεκτονική για τον σχεδιασμό των web services η οποία χρησιμοποιεί υπάρχοντα πρότυπα, όπως HTTP, XML, URL κτλ. Στην PHP η μέθοδος REST υλοποιείται με την ενσωματωμένη βιβλιοθήκη SimpleXML η οποία παρέχει μεθόδους και συναρτήσεις για τον χειρισμό της XML και την μετατροπή των εγγράφων XML σε δεδομένα που μπορούν να χρησιμοποιηθούν από μια γλώσσα προγραμματισμού

Το REST αποτελεί ένα σύνολο από αρχές σχεδίασης μιας δικτυακής υπηρεσίας που επικεντρώνει στους πόρους (π.χ δεδομένα) ενός συστήματος. Η μεταβολή της κατάστασης (ενέργεια επί) των πόρων του συστήματος περιγράφεται και μεταφέρεται στο σύστημα μέσω του πρωτοκόλλου HTTP από διάφορους clients (ανεξαρτήτως της γλώσσας στην οποία έχουν υλοποιηθεί). Για την ιστορία αξίζει να πούμε ότι το REST πρωτοεμφανίστηκε το 2000 από τον Roy Fielding στην ακαδημαϊκή του διατριβή με τίτλο «Architectural Styles and the Design of Network-based Software Architectures». Τα βασικά χαρακτηριστικά του REST παρουσιάζονται στις παρακάτω παραγράφους.

2.6.3.2.1. Αποκλειστική χρήση HTTP αιτημάτων/μεθόδων για την επικοινωνία του χρήστη με τον παροχέα της δικτυακής υπηρεσίας.

Η βασική αρχή σχεδίασης του REST είναι η ένα-προς-ένα αντιστοίχιση μεταξύ λειτουργιών CRUD (create, read, update, delete) και HTTP μεθόδων. Σύμφωνα με αυτή την αντιστοίχιση:

- Για τη δημιουργία ενός πόρου στον server, χρησιμοποιείται η μέθοδος POST.
- Για την ανάσυρση ενός πόρου, χρησιμοποιείται η GET.
- Για την αλλαγή της κατάστασης ενός πόρου ή την ενημέρωσή του, χρησιμοποιείται η PUT.
- Για την απομάκρυνση ή διαγραφή ενός πόρου, χρησιμοποιείται η DELETE.

Με βάση το REST το URI δεν χρησιμοποιείται πια για την περιγραφή της προς εκτέλεση ενέργειας αλλά μόνο τον εντοπισμό του πόρου επί του οποίου θα ασκηθεί η ενέργεια, και τα δεδομένα δεν μεταφέρονται ως παράμετροι στο URI ενός GET αιτήματος αλλά ως XML ή JSON-formatted δεδομένα στο περιεχομενα μιας POST ή PUT μεθόδου. Με άλλα λόγια σε μια υπηρεσία REST, ένα URI εκφράζει ένα αντικείμενο στο οποίο παρέχει πρόσβαση η υπηρεσία μέσω ενός HTTP αιτήματος. Το είδος του αιτήματος καθορίζει την ενέργεια που θα εφαρμοστεί στο αντικείμενο αυτό και το περιεχόμενο του αιτήματος περιέχει διάφορες εξειδικεύσεις τις ενέργειας.

Οπότε ο σχεδιασμός μιας υπηρεσίας REST περιλαμβάνει κυρίως δύο βήματα. Πρώτον, να αποφασιστούν τα αντικείμενα που θα διατίθενται δια μέσω της υπηρεσίας και δεύτερον, τι ενέργεια θα εφαρμόζεται στο κάθε αντικείμενο για κάθε μία από τις GET, POST, PUT και DELETE μεθόδους.

2.6.3.2.2. Έλλειψη Κατάστασής

Η ουσία της «έλλειψης κατάστασης» είναι ότι οποιαδήποτε κλήση σε μια υπηρεσία REST δε θα πρέπει να αναφέρεται σε άλλη προγενέστερη κλήση. Όλες οι κλήσεις πρέπει να είναι ανεξάρτητες. Ο server δε θα πρέπει να γνωρίζει το έγινε με κάποια προηγούμενη κλήση τη στιγμή που επεξεργάζεται την τρέχουσα κλήση. Για παράδειγμα μια δικτυακή υπηρεσία στην οποία ο πελάτης θα πρέπει πρώτα να κάνει μια κλήση για login στην υπηρεσία και μετά μια άλλη κλήση για ανάκτηση της πληροφορίας που επιθυμεί. Αυτή η υπηρεσία δεν είναι stateless γιατί κατά την επεξεργασία της δεύτερης κλήσης η υπηρεσία θα πρέπει να θυμάται ότι αυτός ο χρήστης έχει ήδη συνδεθεί μέσω μιας προηγούμενης κλήσης. Η κατάσταση «συνδεδεμένος» διατηρείται από την υπηρεσία και ο πελάτης δεν χρειάζεται να παρουσιάσει τα πιστοποιητικά του σε περίπτωση μιας δεύτερης ή τρίτης κλήσης της υπηρεσίας. Μια τέτοια διατήρηση καταστάσεων απαγορεύεται σε δικτυακές υπηρεσίες τύπου REST.

Μία υπηρεσία ή χρήστης του REST πρέπει να περιλαμβάνει εντός του HTTP αιτήματος (header και body) όλη την πληροφορία (action, parameters, context,...) που είναι απαραίτητη από τον server για την παραγωγή μιας απόκρισης στο αίτημα αυτό. Τα αιτήματα πρέπει να

είναι ολοκληρωμένα και αυτόνομα έτσι ώστε κατά την επεξεργασία τους από τον server να μην απαιτούν την ανάσυρση κάποιας άλλης πληροφορίας η οποία να επηρεάζει την απόκριση στο ερώτημα. Αυτή η «έλλειψη κατάστασης» σε μια υπηρεσία REST απαλλάσσει τον server από την ανάγκη συγχρονισμού των δεδομένων μιας συνόδου με μια εξωτερική εφαρμογή. Ακόμα περισσότερο επιτρέπει στις υπηρεσίες REST να προσαρμόζονται σε περιπτώσεις όπου απαιτούνται υψηλές επιδόσεις χρησιμοποιώντας clusters of servers με διαμοιρασμό φορτίου (load-balancing) και δυνατότητες για failover, proxies και άλλες τοπολογίες οι οποίες θα προωθούν τα αιτήματα από τον ένα server στον άλλο με σκοπό την ελάττωση του συνολικού χρόνου απόκρισης στην κλήση μιας δικτυακής υπηρεσίας.

2.6.3.2.3. Απεικόνιση της δομής των καταλόγων σαν URIs

Ο σχηματισμός των URI μιας υπηρεσίας REST πρέπει να είναι προβλέψιμος και ευνόητος για τον χρήστη της υπηρεσίας και να απαιτείται από λίγη έως καθόλου τεκμηρίωση ή εξήγηση σχετικά με τον πόρο προς τον οποίο δείχνει κάθε URI.

Τα URI θα πρέπει να είναι στατικά, έτσι ώστε όταν οι πόροι της υπηρεσίας υφίστανται αλλαγές (προσθήκες, ενημερώσεις, διαγραφές, κ.λ.π) ή αλλάζει η υλοποίηση της υπηρεσίας, ο σύνδεσμος προς έναν πόρο του συστήματος να παραμένει ίδιος. Αυτό επιτρέπει τη δημιουργία σελιδοδεικτών.

2.6.3.3. Σύγκριση SOAP-REST

Πλεονεκτήματα SOAP	Πλεονεκτήματα REST
είναι σχεδιασμένο για να χειριστεί κατανεμημένα υπολογιστικά περιβάλλοντα είναι το πρότυπο που επικρατεί για τις υπηρεσίες web	πολύ πιο απλό να αναπτυχθεί από το SOAP εύκολη εκμάθηση λιγότερη εξάρτηση από τα εργαλεία, συνοπτικό

▪ έχει καλύτερη υποστήριξη από άλλα πρότυπα (WSDL, WS-*) και εργαλεία από τους πωλητές ▪ ενσωματωμένη αντιμετώπιση των λαθών, ▪ επεκτασιμότητα.	▪ πιο κοντά στο σχεδιασμό και τη φιλοσοφία στο Web

Μειονεκτήματα SOAP	Μειονεκτήματα ReST
▪ Θεωρητικά πιο δύσκολο, πιο "χοντρό" από το REST ▪ Πιο ≪φλύαρο≫ ▪ σκληρότερο για την ανάπτυξη ▪ απαιτεί τη χρήση μέσων	▪ Προϋποθέτει μία επικοινωνία σημείο-προς-σημείο και όχι μεταξύ ενδιάμεσων ▪ Έλλειψη προτύπων υποστήριξης για την ασφάλεια, ▪ Έλλειψη προτύπων υποστήριξης για την πολιτική, ▪ την αξιόπιστη ανταλλαγή μηνυμάτων, ▪ συνδεδεμένο με το μοντέλο μεταφοράς HTTP.

2.6.4 Γλώσσα Περιγραφής Υπηρεσιών Διαδικτύου – WSDL

Μέχρι εδώ έχουμε μιλήσει για το SOAP. Έχουμε εξηγήσει ότι είναι βασισμένο στην XML και ότι μέσω αυτού του πρωτοκόλλου μπορούν να γίνουν αιτήσεις για απομακρυσμένες κλήσεις διαδικασιών από κάποιον πελάτη σε κάποιον εξυπηρέτη. Από τη μεριά του πελάτη όμως υπάρχουν δυσκολίες στη σύνταξη του μηνύματος SOAP, καθώς ο πελάτης δε μπορεί να ξέρει τι ακριβώς μήνυμα να στείλει. Το SOAP καθορίζει κάποιους κανόνες και προσφέρει μια συγκεκριμένη φόρμα για τα μηνύματα, αλλά πρέπει να βρεθεί κάποιος άλλος τρόπος ώστε ο πελάτης να είναι σε θέση να γνωρίζει τι μήνυμα να στείλει στον εξυπηρέτη του παροχέα της υπηρεσίας.

Επίσης, ο πελάτης πρέπει να γνωρίζει αρκετές λεπτομέρειες προτού στείλει το μήνυμα. Πρέπει να γνωρίζει καταρχάς που ακριβώς να το στείλει, ή ποιες μεθόδους θέλει να καλέσει από την υπηρεσία, ή ποια πρωτόκολλα επικοινωνίας υποστηρίζονται από τον παροχέα της υπηρεσίας.

Προκειμένου να γνωρίζει ο πελάτης όλες τις απαραίτητες πληροφορίες για την κλήση μιας υπηρεσίας διαδικτύου, πρέπει να έχει στην κατοχή του μια περιγραφή αυτής.

Για να γίνει η όλη διαδικασία ακόμη πιο εύκολη και πιο τυποποιημένη χρειάζεται ένας καθορισμένος μηχανισμός για την περιγραφή των υπηρεσιών διαδικτύου. Αυτός ο μηχανισμός θα παρέχει πληροφορίες για ακριβώς αυτά που χρειάζεται να ξέρει ο χρήστης που θέλει να καλέσει την υπηρεσία. Επιπρόσθετα, χάρη σε αυτόν το μηχανισμό είναι δυνατή η ανάπτυξη εφαρμογών που θα βοηθούν τους προγραμματιστές στην ανάπτυξη των υπηρεσιών μέσω των περιγραφών τους.

Όπως έχει αναφερθεί και σε προηγούμενη ενότητα, ο ρόλος της υπηρεσίας καταλόγου (Service Registry) σε μια SOA αρχιτεκτονική είναι πολύ σημαντικός. Ο πελάτης, προκειμένου να ανακαλύψει την υπηρεσία που τον ενδιαφέρει, ψάχνει πρώτα στον κατάλογο αυτό. Το αποτέλεσμα της αναζήτησης δεν είναι τίποτε άλλο από την περιγραφή της υπηρεσίας που ζήτησε. Ο παροχέας της υπηρεσίας διαδικτύου, λοιπόν, δημοσιεύει την περιγραφή της υπηρεσίας του γι' αυτό το λόγο. Για να ξέρει ο κάθε πιθανός πελάτης πώς να εγκαθιδρύσει επικοινωνιακή σύνδεση μαζί του και πως ακριβώς να καλέσει την υπηρεσία που του παρέχει.

Μια καλή περιγραφή μιας υπηρεσίας διαδικτύου πρέπει να αποτελείται από δύο μέρη:

- **Τη λειτουργική περιγραφή.** Η περιγραφή αυτή περιγράφει ποιες ακριβώς λειτουργίες είναι διαθέσιμες από την υπηρεσία διαδικτύου και ποια είναι η απαιτούμενη σύνταξη που θα πρέπει να έχει το SOAP μήνυμα προκειμένου να γίνει με επιτυχία η κλήση. Επίσης, πληροφορίες σχετικές με την τοποθεσία της υπηρεσίας είναι διαθέσιμες ώστε ο χρήστης να έχει το ακριβές URL στο οποίο θα γίνει η κλήση. Είναι χρέος της περιγραφής αυτής να δώσει όλες τις απαραίτητες πληροφορίες στον πελάτη ώστε να καλέσει την υπηρεσία.
- **Τη μη-λειτουργική περιγραφή**. Η λειτουργική περιγραφή είναι πολύ σημαντική, αφού χωρίς αυτή δεν είναι δυνατόν να επιτευχθεί η κλήση της υπηρεσίας. Η μη-λειτουργική περιγραφή όμως είναι εξίσου σημαντική. Όπως υποδηλώνει και το όνομά της, προσφέρει πληροφορίες που σχετίζονται με μη λειτουργικές απαιτήσεις της υπηρεσίας. Τέτοιες πληροφορίες μπορεί να περιγράφουν το λόγο που κάποιος πελάτης να επιθυμεί να καλέσει την υπηρεσία. Επίσης, μπορεί να περιγράφουν τον παροχέα της υπηρεσίας με αρκετή λεπτομέρεια. Μπορεί να προσφέρουν προσωπικά του στοιχεία και τηλέφωνα επικοινωνίας. Τέλος, πληροφορίες για την

ασφάλεια γύρω από την υπηρεσία και την πολιτική του παροχέα μπορεί να είναι διαθέσιμες μέσω της μη λειτουργικής περιγραφής της υπηρεσίας.

Η ευρέως χρησιμοποιούμενη και καθιερωμένη γλώσσα περιγραφής υπηρεσιών διαδικτύου είναι η WSDL (Web Services Description Language). Είναι μια γλώσσα βασισμένη στην XML και περιγράφει τρεις σημαντικές ιδιότητες της υπηρεσίας:

- **Τι κάνει η υπηρεσία**. Εδώ περιγράφονται οι μέθοδοι της υπηρεσίας, οι παράμετροί της και τα αποτελέσματα που επιστρέφει.
- **Πώς γίνεται η πρόσβαση στην υπηρεσία** Εδώ παρέχονται όλες οι απαραίτητες πληροφορίες για τη σύνταξη των SOAP μηνυμάτων που θα ανταλλαγούν, καθώς και τα πρωτόκολλα που υποστηρίζονται από τον παροχέα.
- **Που βρίσκεται η υπηρεσία** Εδώ παρέχονται πληροφορίες για τη δικτυακή διεύθυνση της υπηρεσίας. Συνήθως είναι ένα URL.

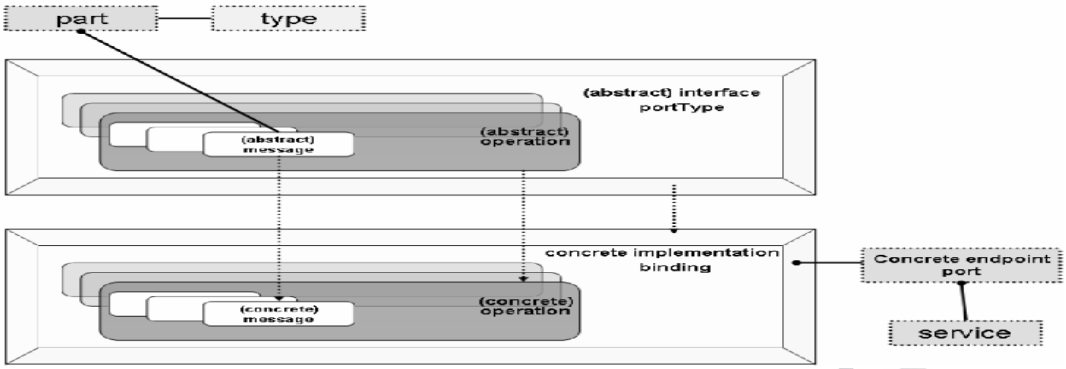
Όπως κάθε καλά ορισμένη γλώσσα βασισμένη σε XML, έτσι και η WSDL ορίζει κάποια βασικά στοιχεία που συναντώνται σε ένα WSDL έγγραφο. Τα κυριότερα από αυτά είναι:

- **PortType** Είναι ένας αφαιρετικός ορισμός της διασύνδεσης της υπηρεσίας διαδικτύου. Ουσιαστικά, σκοπός του είναι η περιγραφή αυτής της διασύνδεσης. Όπως γίνεται σε έναν κώδικα Java, που η διασύνδεση αποτελείται μόνο από τις υπογραφές των μεθόδων, έτσι κι εδώ ο ορισμός του portType είναι η συλλογή των λειτουργιών που παρέχει η υπηρεσία διαδικτύου. Ένα τέτοιο στοιχείο μπορεί να έχει ένα απλό όνομα, το οποίο πιθανόν να σχετίζεται με τη λειτουργία που προσφέρει η υπηρεσία. Εάν ένα έγγραφο WSDL περιέχει πολλά τέτοια στοιχεία, τότε το κάθε ένα οφείλει να έχει διαφορετικό όνομα.
- **Message** Καθορίζει τη μορφή του μηνύματος, καθώς και το σύνολο των παραμέτρων που σχετίζονται με τις μεθόδους της υπηρεσίας. Το στοιχείο αυτό μπορεί να αποσυντεθεί σε πολλά μέρη, που ονομάζονται parts. Κάθε στοιχείο message μέσα σε ένα WSDL έγγραφο πρέπει να έχει ένα μοναδικό όνομα ώστε να ξεχωρίζει από τα υπόλοιπα. Ουσιαστικά, τα στοιχεία αυτά δεν παρουσιάζουν ιδιαίτερο ενδιαφέρον αφού δεν είναι τίποτε άλλο από μια συλλογή από parts και κάθε part έχει ένα όνομα, που συνήθως αντικατοπτρίζει την πληροφορία που περιέχεται στο part.
- **Types** Ορίζει τη συλλογή όλων των τύπων δεδομένων που χρησιμοποιούνται από την υπηρεσία διαδικτύου. Το εξ ορισμού σύστημα των τύπων δεδομένων στην

WSDL είναι το XML σχήμα (XML Schema – XSD). Το σχήμα αυτό μπορεί να χρησιμοποιηθεί για να περιγράψει όλους τους τύπους δεδομένων που μπορεί να χρησιμοποιηθούν στα διάφορα μηνύματα που ανταλλάσσονται για την κλήση της υπηρεσίας. Το στοιχείο types είναι ένα μέρος για το WSDL έγγραφο στο οποίο ο χρήστης ορίζει τύπους δεδομένων που θα χρησιμοποιηθούν σε άλλα στοιχεία του εγγράφου.

- **Binding** Περιέχει λεπτομέρειες για το πώς τα στοιχεία σε μια αφαιρετική δομή (όπως το portType) μπορούν να μετασχηματισθούν σε μια συμπαγή αναπαράσταση των δεδομένων και πρωτοκόλλων που θα χρησιμοποιηθούν για την επικοινωνία μεταξύ του πελάτη και της υπηρεσίας. Ουσιαστικά είναι το στοιχείο που θα πει στον πελάτη πώς ακριβώς να μορφοποιήσει το μήνυμα που θα στείλει για την κλήση της υπηρεσίας. Κάθε στοιχείο portType μπορεί να έχει ένα ή περισσότερα στοιχεία binding συσχετισμένα με αυτό. Για ένα συγκεκριμένο στοιχείο portType, ένα στοιχείο binding περιγράφει τον τρόπο που πρέπει να γίνει η κλήση των μεθόδων της υπηρεσίας χρησιμοποιώντας κάποιο καθορισμένο πρωτόκολλο, όπως το SOAP, πάνω σε κάποιο επίσης καθορισμένο πρωτόκολλο μεταφοράς, όπως το HTTP. Το όνομα του binding πρέπει να είναι μοναδικό καθώς πολλά τέτοια στοιχεία μπορεί να υπάρχουν σε ένα WSDL έγγραφο.
- **Port** Περιγράφει πως ακριβώς εκφράζεται ένα binding σε μια φυσική θύρα του δικτύου. Ο ρόλος της είναι πολύ απλός αφού δεν κάνει τίποτε άλλο από μια απλή αντιστοιχία. Τα στοιχεία αυτά, όπως και τα προηγούμενα, οφείλουν να φέρουν ένα όνομα, μοναδικό μέσα στο WSDL έγγραφο. Συνήθως, τα στοιχεία αυτά υποδεικνύουν το URL στο οποίο πρέπει να σταλούν τα μηνύματα SOAP ώστε να γίνει η κλήση της υπηρεσίας. Τα στοιχεία αυτά δε συναντούνται μόνα τους μέσα στο έγγραφο της περιγραφής της υπηρεσίας. Είναι στοιχεία – παιδιά του συστατικού service.
- **Service** Ένα τέτοιο στοιχείο είναι μια συλλογή από ports. Το στοιχείο αυτό μπορεί να έχει ένα μοναδικό όνομα μέσα στο έγγραφο. Παρόλο που ο ρόλος του δεν έχει ιδιαίτερη σημασία, είναι καλό τα στοιχεία port να ομαδοποιούνται κάτω από ένα στοιχείο service. Συνήθως όμως ένα μόνο στοιχείο port συναντάται και έτσι καταλήγουμε στο να έχουμε ένα στοιχείο service, που να περιέχει ένα στοιχείο port.

Η εικόνα 12 που ακολουθεί απεικονίζει τις συσχετίσεις μεταξύ των συστατικών της WSDL.



Εικόνα 12:Ασύμβατα COBRA / DCOM δίκτυα που απαιτούν γέφυρα για να

Η δομή ενός WSDL εγγράφου ξεκινά με ένα στοιχείο ορισμών, το οποίο καλείται definitions. Το στοιχείο αυτό περιέχει όλα τα υπόλοιπα στοιχεία που μπορεί να συναντηθούν μέσα στο έγγραφο, δηλαδή μπορεί να περιέχει:

- Κανένα ή οσαδήποτε στοιχεία τεκμηρίωσης. Τα στοιχεία αυτά καλούνται documentation elements και χρησιμοποιούνται για να παρέχουν χρήσιμες πληροφορίες κατανοητές από τον άνθρωπο γύρω από την υπηρεσία διαδικτύου.
- Κανένα ή οσαδήποτε στοιχεία εισαγωγών. Τα στοιχεία αυτά καλούνται import elements. Τα στοιχεία δίνουν τη δυνατότητα επαναχρησιμοποίησης ήδη υπαρχόντων WSDL εγγράφων επιτρέποντας σε διάφορα στοιχεία να έχουν αναφορές σε άλλα ήδη υπάρχοντα που βρίσκονται σε διαφορετικό έγγραφο σε διαφορετικό αρχείο.
- Προαιρετικά ένα στοιχείο types.
- Κανένα ή οσαδήποτε στοιχεία message.
- Κανένα ή οσαδήποτε στοιχεία portType. Συνήθως μόνο ένα συναντάται.
- Κανένα ή οσαδήποτε στοιχεία binding. Συνήθως μόνο ένα συναντάται, το οποίο έχει να κάνει με το στοιχείο portType.
- Κανένα ή οσαδήποτε στοιχεία service. Και πάλι συνήθως μόνο ένα είναι αρκετό.

2.6.5 Universal Description, Discovery and Integration (UDDI)

Στις αρχές του 2000, και καθώς η ιδέα των υπηρεσιών ιστού άρχισε να κερδίζει έδαφος μέσα στην κοινότητα, έγινε σαφές ότι η καταχώρηση σε καταλόγους των υπηρεσιών ιστού ήταν κάτι απαραίτητο για να μπορέσει να εφαρμοστεί πρακτικά η ιδέα.

Επιπλέον, λόγο της γενικότερης στροφής προς τα ανοικτά πρότυπα, οτιδήποτε εκτός από έναν πρότυπο μηχανισμό αλληλεπίδρασης και αναζήτησης δε θα μπορούσε να γίνει αποδεκτό. Ένα τέτοιο πρότυπο καταλόγου, θα έπρεπε να υποστηριχθεί και από τις περισσότερες, αν όχι από όλες τις μεγάλες εταιρίες παροχής λογισμικού, και να υιοθετηθεί από τις διάφορες βιομηχανίες. Έτσι η πρωτοβουλία του UDDI, που ήταν το αποτέλεσμα πολλών μηνών συνεργασίας μεταξύ αντιπροσώπων από τις Ariba, IBM, και Microsoft, που ξεκίνησε την άνοιξη του 2000, γεννήθηκε και ανακοινώθηκε επίσημα στις 6 Σεπτεμβρίου 2000. Η υποστήριξη για το UDDI επεκτάθηκε και πέρα από τις τρεις εταιρίες που συμμετείχαν αρχικά και αυτή τη στιγμή, το έργο UDDI περιλαμβάνει μια κοινότητα που αριθμεί πάνω από 310 εταιρίες.

Ο σκοπός του UDDI είναι να διευκολύνει την ανακάλυψη υπηρεσιών και κατά το χρόνο σχεδίασης, και δυναμικά κατά το χρόνο εκτέλεσης. Συνεπώς το UDDI λειτουργεί ως ένα κοινό άμεσα συνδεδεμένο κατάλογο (και των αντιστοίχων υπηρεσιών), το οποίο λειτούργησε για πρώτη φορά στις 2 Μαΐου 2001. Ο κατάλογος αυτός, συνήθως αναφέρεται ως κατάλογος επιχειρήσεων UDDI (UDDI Business Registry). Ο κατάλογος αυτός, στην πραγματικότητα αποτελείται από 2 πανομοιότυπους καταλόγους, που διατηρούνται αυτή τη στιγμή από δύο εταιρίες (IBM και Microsoft), οι οποίοι και αποκαλούνται οι διαχειριστές του UDDI.

Για να γίνει κάποιος διαχειριστής ενός τέτοιου καταλόγου, πρέπει να ακολουθήσει αυστηρές συμφωνίες για την αντιγραφή των δεδομένων, το απόρρητο των δεδομένων και τις διάφορες πολιτικές. Από την οπτική μιας καταχωρημένης επιχείρησης αλλά και από την οπτική του απλού χρήστη, η απαίτηση είναι να μην υπάρχει διαφορά ανάμεσα στα διαφορετικά αντίγραφα του καταλόγου. Οι επιχειρήσεις μπορούν να κάνουν εγγραφή σε οποιονδήποτε διαχειριστή UDDI, και τα δεδομένα τους θα αντιγραφούν σε όλα τα άλλα αντίγραφα του καταλόγου στους άλλους διαχειριστές.

Επομένως, οι χρήστες μπορούν να αναζητήσουν στον κατάλογο οποιουδήποτε διαχειριστή για να βρουν τις επιχειρήσεις που αναζητούν, άσχετα με το διαχειριστή που έχει επιλέξει η συγκεκριμένη επιχείρηση για να εγγραφεί. Υπάρχουν όμως κάποια λεπτά θέματα που έχουν σχέση με την απόφαση σχετικά με το ποιον διαχειριστή να χρησιμοποιήσει μια επιχείρηση για να εγγραφεί. Για παράδειγμα, κάποιοι διαχειριστές μπορεί να ζητούν επιπρόσθετες

προαιρετικές πληροφορίες που δεν απαιτούνται από το UDDI. Αυτή η πληροφορία δεν αντιγράφεται στα αντίγραφα του καταλόγου στους άλλους διαχειριστές. Επιπλέον, αυτό που πρέπει να σημειωθεί είναι ότι από τη στιγμή που μια επιχείρηση επιλέξει κάποιο διαχειριστή για να εγγραφεί, η οποιαδήποτε ενημέρωση και τροποποίηση των στοιχείων θα πρέπει να γίνει από τον ίδιο διαχειριστή. Αυτό πρέπει να γίνεται έτσι δεδομένου ότι ο κάθε διαχειριστής ακολουθεί διαφορετικές πολιτικές πιστοποίησης και ασφάλειας, και επομένως η πληροφορία πιστοποίησης δεν είναι εύκολο να αντιγραφεί στους άλλους διαχειριστές.

Το UDDI είναι κάτι παραπάνω από ένας κατάλογος επιχειρήσεων και υπηρεσιών. Ορίζει ένα σύνολο από δομές δεδομένων και προδιαγραφές API, για την προγραμματική αναζήτηση και καταχώρηση εταιριών, υπηρεσιών, δεσμών, και ειδών υπηρεσιών. Στα τυπικά σενάρια υπηρεσιών διαδικτύου , οι παροχείς υπηρεσιών διαδικτύου θα θελήσουν να δημοσιεύσουν τις περιγραφές των εταιριών και των υπηρεσιών τους σε ένα κατάλογο, και όσοι αναζητούν υπηρεσίες, είτε κατά τη σχεδίαση, είτε κατά την εκτέλεση, θα θελήσουν να αναζητήσουν στον κατάλογο περιγραφές υπηρεσιών. Οι προδιαγραφές API του UDDI παρέχουν γι' αυτό το λόγο ένα σύνολο από προγραμματιστικές διεπαφές API) δημοσίευσης για την καταχώρηση υπηρεσιών, και αντίστοιχες διεπαφές ερωτήσεων API για την εύρεση υπηρεσιών.

Επιπλέον από την παροχή μιας προγραμματιστικής διεπαφής, οι διαχειριστές του καταλόγου UDDI, παρέχουν ένα σύστημα αλληλεπίδρασης με το χρήστη, βασισμένο στο διαδίκτυο, για την καταχώρηση, διαχείριση, ανεύρεση εταιριών και υπηρεσιών μέσα στον κατάλογο.

2.6.5.1 Οι απαιτήσεις του καταλόγου UDDI

Οι κατάλογοι γενικά, είτε είναι φτιαγμένα για στοιχεία λογισμικού, για υπηρεσίες, είτε για κάποιο άλλο σκοπό, έχουν κάποιες βασικές απαιτήσεις:

- Ένα σύνολο από προδιαγραφές δομών δεδομένων για τα μεταδεδομένα που θα αποθηκευθούν στον κατάλογο.
- Ένα σύνολο από προδιαγραφές λειτουργιών Δημιουργίας, Ανάγνωσης, Ενημέρωσης, Διαγραφής (CRUD – Create, Read, Update, Delete) για την αποθήκευση, διαγραφή, και ερώτηση των δεδομένων του καταλόγου.

Οι συνήθεις απαιτήσεις για τα μεταδεδομένα είναι:

- Ιδιοκτησία και περιεχόμενο, δηλαδή ότι κάποιος πρέπει να είναι ο ιδιοκτήτης των δεδομένων που δημοσιεύει, καθώς και όλων των δεδομένων που περιέχονται σε αυτά, εκτός εάν δηλωθεί διαφορετικά.
- Κατηγοριοποίηση, δηλαδή τα δεδομένα μπορούν να είναι ταξινομημένα σε μια ή περισσότερες κατηγορίες, για να διευκολύνουν την οργάνωση και την αναζήτηση.

Οι συνήθεις απαιτήσεις για τις λειτουργίες του καταλόγου είναι:

- Η πιστοποίηση για διαδικασίες που αλλάζουν δεδομένα και για δημόσιους καταλόγους.
- Η ελεύθερη πρόσβαση για λειτουργίες ερώτησης και ανάγνωσης.

Το τμήμα του καταλόγου UDDI, διαθέτει ένα μοντέλο πληροφορίας που αντιπροσωπεύει τις προδιαγραφές των δομών δεδομένων, και μια προγραμματιστική διεπαφή για τις προδιαγραφές των λειτουργιών. Το μοντέλο πληροφοριών του UDDI σχεδιάστηκε με τέτοιο τρόπο ώστε να είναι ευέλικτο και επεκτάσιμο, επιτρέποντας έτσι και τη χρήση οποιουδήποτε μοντέλου περιγραφής υπηρεσιών που θα επιθυμούσε να ορίσει κάποιος χρήστης.

Κλείνοντας την αναφορά στην προγραμματιστική διεπαφή του UDDI, θα πρέπει να τονιστεί, ότι όλες σχεδόν οι λειτουργίες των προγραμματιστικών διεπαφών δημοσίευσης και ερώτησης είναι δυνατόν να πραγματοποιηθούν και από τις ιστοσελίδες των μεμονωμένων διαχειριστών. Η προγραμματιστική διεπαφή όμως, εκτός από του ότι είναι και πολύ πιο πλήρης από το σύστημα αλληλεπίδρασης ενός επόπτη διαδικτύου, παρέχει επίσης τη δυνατότητα στις επιχειρήσεις να εκτελούν όλες τις λειτουργίες προγραμματιστικά, βοηθώντας στην δυναμική (κατά το χρόνο εκτέλεσης) ανεύρεση υπηρεσιών.

Κεφάλαιο 3. Εργαλεία υλοποίησης

3.1 Εισαγωγή

Υπάρχουν αρκετές πλατφόρμες στις οποίες μπορεί να βασιστεί κανείς για τη δημιουργία ενός Web Service. Η ανάπτυξη των Web Services που περιγράφονται σε αυτή την εργασία βασίστηκαν σε πλατφόρμες της Microsoft. Σε αυτή την ενότητα γίνεται μια παρουσίαση των εργαλείων που χρησιμοποιήθηκαν

3.2 Microsoft Visual Studio 2010

Το Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE) που χρησιμοποιείται για την ανάπτυξη διαδικτυακών και desktop εφαρμογών καθώς και διαδικτυακών υπηρεσιών (Web Services).

Το Visual Studio εισάγει μια νέα διαδικασία ανάπτυξης προγραμμάτων, τον παραστατικό προγραμματισμό, που αλλάζει τον τρόπο εγγραφής και εκτέλεσης των προγραμμάτων, οδηγώντας σε αύξηση της παραγωγικότητας. Παρέχει προχωρημένα εργαλεία για τη διόρθωση λαθών, για τη τεκμηρίωση και εγγραφή κώδικα, για την ανάπτυξη διεπαφών χρήστη, για τη σχεδίαση κλάσεων καθώς και για τη σχεδίαση του σχήματος μίας βάσης δεδομένων (database schema). Επίσης υποστηρίζει πολλά plug-ins που προσφέρουν προηγμένες λειτουργίες όπως unit testing και refactoring.

Οι ενσωματωμένες γλώσσες προγραμματισμού του Visual Studio είναι οι Visual C++, Visual C# και Visual Basic. Επίσης, παρέχεται υποστήριξη και για άλλες γλώσσες όπως τις F#, Python, Ruby οι οποίες εγκαθίστανται ξεχωριστά μέσω των language services.

3.3 .NET Framework

Το .Net Framework είναι αναπόσπαστο κομμάτι των Windows και υποστηρίζει την ανάπτυξη και λειτουργία εφαρμογών και διαδικτυακών υπηρεσιών (Web Services).

Το .NET Framework έχει σχεδιαστεί για να εκπληρώσει τους ακόλουθους στόχους:

- Να παράσχει ένα συνεκτικό αντικειμενοστραφή προγραμματιστικό περιβάλλον όπου ο κώδικας είτε αποθηκεύεται και εκτελείται τοπικά, είτε εκτελείται τοπικά αλλά διανέμεται στο διαδίκτυο, είτε εκτελείται από απόσταση (remotely).
- Να παράσχει ένα περιβάλλον εκτέλεσης κώδικα που αυξάνει τη παραγωγικότητα του προγραμματιστή και ελαχιστοποιεί τις πιθανές συγκρούσεις μεταξύ διαφορετικών εκδόσεων του .Net Framework.
- Να παράσχει ένα περιβάλλον εκτέλεσης κώδικα που να προάγει την ασφαλή εκτέλεση του κώδικα, περιλαμβανομένου του κώδικα που δημιουργήθηκε από άγνωστο ή μη έμπιστο τρίτο μέρος.
- Να παρέχει ένα σταθερό περιβάλλον ανάπτυξης για όλα τα είδη εφαρμογών (Windows-based ή Web-based εφαρμογές).
- Να εξασφαλίσει ότι ο κώδικας που βασίζεται στο .NET Framework μπορεί να ενσωματωθεί σε οποιονδήποτε άλλο κώδικα.

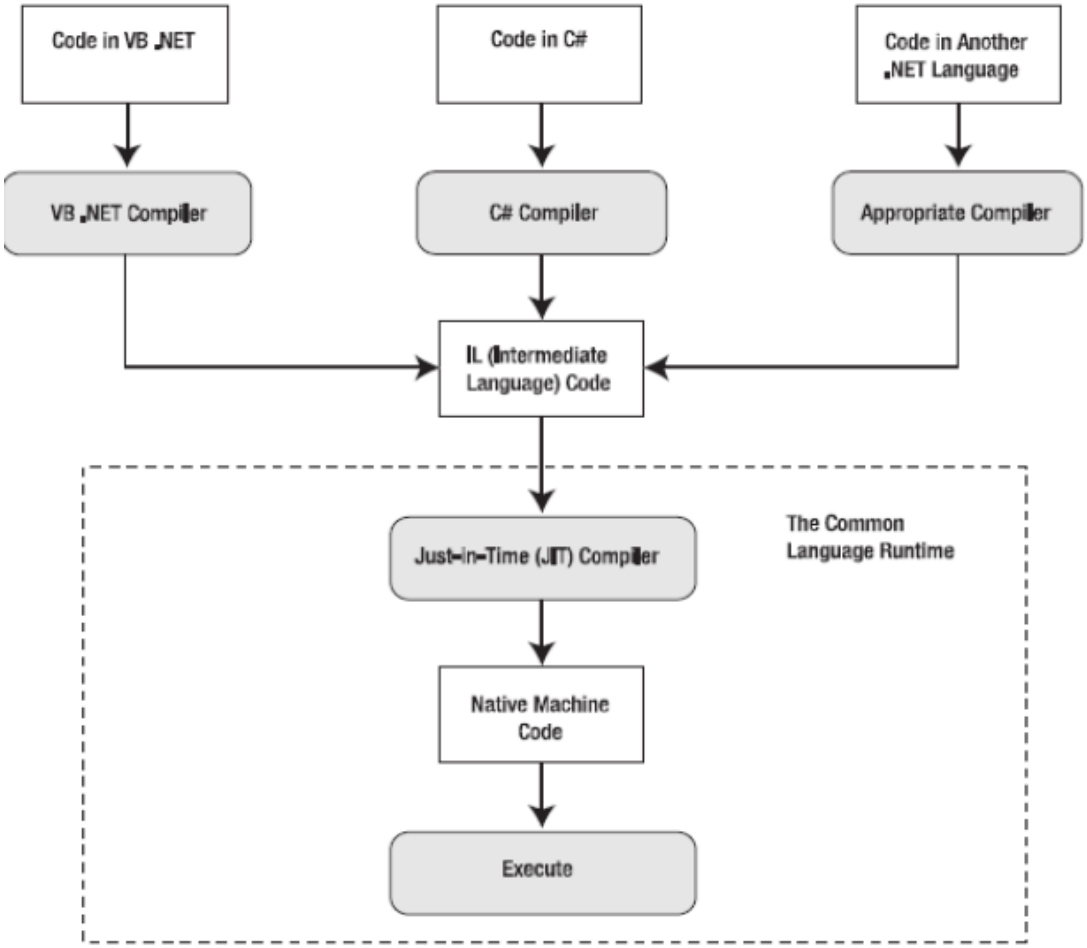
Το .NET Framework έχει δύο κύρια στοιχεία: Το Common Language Runtime (CLR) και τη βιβλιοθήκη κλάσεων του .NET Framework. Το CLR καθορίζει το περιβάλλον για την εκτέλεση κώδικα του προγράμματος και οι προγραμματιστές που χρησιμοποιούν το CLR γράφουν κώδικα σε μια γλώσσα όπως τη Visual Basic ή τη C#. Κατά τη διάρκεια της μεταγλώττισης του προγράμματος ο .NET μεταγλωττιστής μετατρέπει τον εν λόγω κώδικα σε ενδιάμεσο κώδικα (byte code) που ονομάζεται Common Intermediate Language (CIL), ο οποίος ουσιαστικά ορίζει οδηγίες για το (CLR) που λειτουργεί ως εικονική μηχανή (virtual machine).

Κατόπιν κατά τη διάρκεια της εκτέλεσης ο μεταγλωττιστής του CLR μετατρέπει τον CIL κώδικα σε κώδικα μηχανής. Το CLR επιτρέπει στους προγραμματιστές να αγνοήσουν πολλές λεπτομέρειες της συγκεκριμένης CPU που θα εκτελέσει το πρόγραμμα και παρέχει και άλλες σημαντικές υπηρεσίες, όπως:

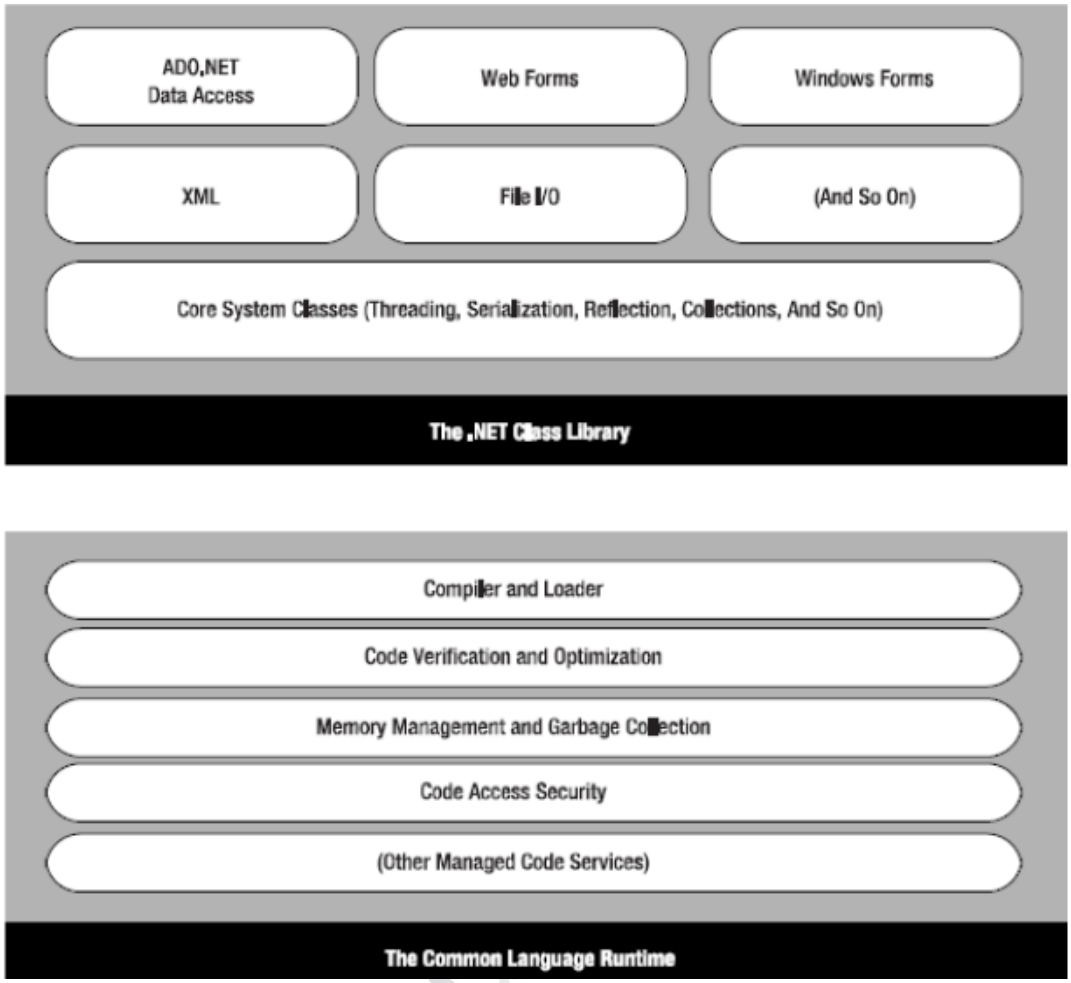
- Διαχείριση μνήμης
- Διαχείριση νημάτων (threads)
- Διαχείριση εξαιρέσεων
- Συλλογή απορριμμάτων (garbage collection)

- Ασφάλεια

Η βιβλιοθήκη κλάσεων, το άλλο βασικό στοιχείο του .NET Framework, είναι μια ολοκληρωμένη συλλογή από επαναχρησιμοποιήσιμους τύπους που μπορούν να χρησιμοποιηθούν για την ανάπτυξη εφαρμογών που κυμαίνονται από command-line εφαρμογές ή GUI εφαρμογές μέχρι και εφαρμογές που βασίζονται στις τελευταίες καινοτομίες που παρέχονται από το ASP.NET, όπως Web Forms και XML διαδικτυακές υπηρεσίες.



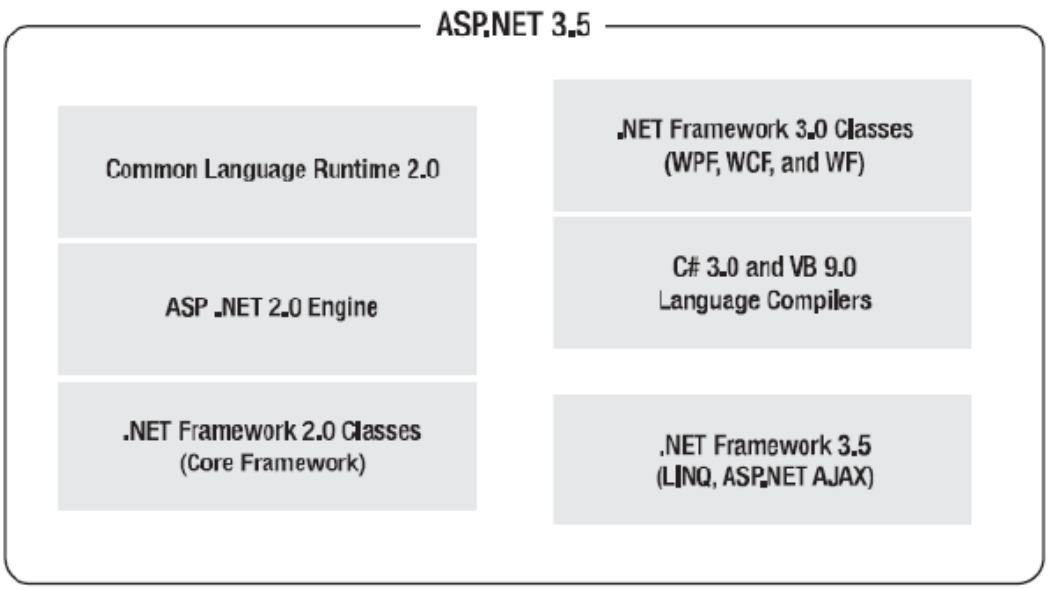
Εικόνα 13:Μεταγλώττιση μιας ASP.NET ιστοσελίδας



Εικόνα 14:Τα συστατικά στοιχεία του CLR και του .NET Framework

3.4 ASP.NET 3.5

Το Microsoft Active Server Pages .NET (ASP.NET) αποτελεί ένα σύνολο τεχνολογιών του .NET Framework για τη δημιουργία εφαρμογών διαδικτύου (web) και διαδικτυακών υπηρεσιών (Web Services). Οι σελίδες ASP.NET εκτελούνται στον εξυπηρετητή και δημιουργούν markup, όπως HTML, WML ή XML, που στέλνεται σε οποιονδήποτε browser υπολογιστή ή κινητού τηλεφώνου. Χρησιμοποιούν ένα μοντέλο προγραμματισμού μεταγλωττιζόμενο και καθοδηγούμενο από γεγονότα το οποίο βελτιώνει την επίδοση και διαχωρίζει το στρώμα εφαρμογής (application logic) και τη διεπαφή χρήστη (user interface).



Εικόνα 15:Τα συστατικά στοιχεία του ASP.NET 3.5

3.5 Visual C#.NET

Η Visual C#.NET είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που αναπτύχθηκε από τη Microsoft σαν μέρος της πρωτοβουλίας του .NET και το 2003 πιστοποιήθηκε ως πρότυπο από την ISO (ISO / IEC 23270).

Η Microsoft έχει δώσει μεγάλη βαρύτητα στην ανάπτυξη και στην εξέλιξη της Visual C#.NET και αποτελεί αναμφισβήτητα τη "ραχοκοκαλιά" της πλατφόρμας .NET με τις υπόλοιπες γλώσσες να διαδραματίζουν δευτερεύοντα ρόλο.

Η Visual C#.NET ανήκει στην ίδια κατηγορία γλωσσών με τη Java και τη C++ και η μετάβαση του προγραμματιστή από τη μια γλώσσα στην άλλη είναι άμεση.

Ορισμένα από τα κύρια χαρακτηριστικά της γλώσσας είναι τα εξής:

- **Garbage collection:** Αποδεσμεύει αυτόματα τη μνήμη από αντικείμενα που δεν χρησιμοποιούνται πλέον από το πρόγραμμα.
- **Διαχείριση εξαιρέσεων:** Παρέχει μια δομημένη και επεκτάσιμη προσέγγιση για την ανίχνευση σφαλμάτων.
- **Component-oriented προγραμματισμός:** Υποστηρίζει το μοντέλο προγραμματισμού που βασίζεται σε components, δηλαδή σε στοιχεία

προγράμματος που μπορούν να χρησιμοποιηθούν από άλλους χρήστες, οι οποίοι μπορέι να γνωρίζουν μονάχα ό,τι κρίνει απαραίτητο ο συγγραφέας του component και χωρίς ο ίδιος να ξέρει για τους χρήστες.

- **Type-safe σχεδιασμός:** Καθιστά αδύνατη την ανάγνωση μεταβλητών που δεν έχουν αρχικοποιηθεί, τη χρήση δεικτών που βρίσκονται εκτός ορίων πίνακα καθώς και την εκτέλεση unchecked type casts.

3.6 Internet Information Services (IIS)

Ο IIS είναι ένας ευέλικτος, ασφαλής και εύκολα διαχειρίσιμος εξυπηρετητής ιστού (Web Server) που αναπτύχθηκε από τη Microsoft για τη φιλοξενία διαδικτυακών εφαρμογών. Παρόλο που ο IIS παρέχει κυρίως φιλοξενία σε εφαρμογές που έχουν αναπτυχθεί με .NET τεχνολογία, μπορεί να χρησιμοποιηθεί και από εφαρμογές που έχουν αναπτυχθεί σε PHP, JSP ή Perl.

Τα πρωτόκολλα που υποστηρίζονται από τον IIS7 είναι τα εξής: FTP, FTPS, SMTP, NNTP, και HTTP/HTTPS.

3.7 SQL Server 2008 R2

Ο SQL Server είναι ένα σύστημα διαχείρισης βάσεων δεδομένων (Database Management System) που έχει αναπτυχθεί και προωθείται από τη Microsoft. Ως σύστημα διαχείρισης βάσης δεδομένων υποστηρίζει τα εξής χαρακτηριστικά: Περιγραφή δεδομένων, ανεξαρτησία δεδομένων και λειτουργιών, αποδοτικότερη διαχείριση δεδομένων, προστασία δεδομένων και δικαιώματα χρηστών, μηχανισμούς ταυτόχρονης προσπέλασης, και επεκτασιμότητα. Η σχεσιακή γλώσσα που χρησιμοποιεί ο SQL Server για την επικοινωνία των εφαρμογών με το DBMS ονομάζεται Transact – SQL (T – SQL). Είναι μια διάλεκτος της πιο σημαντικής γλώσσας βάσεων δεδομένων που υπάρχει σήμερα, της Δομημένης Γλώσσας Ερωτημάτων SQL (Structured Query Language).

Ο SQL Server περιλαμβάνει επίσης μια σειρά πρόσθετων υπηρεσιών οι οποίες, αν και δεν είναι απαραίτητες για τη λειτουργία του συστήματος βάσης δεδομένων, παρέχουν

υπηρεσίες προστιθέμενης αξίας πάνω από το κεντρικό σύστημα διαχείρισης βάσης δεδομένων. Οι υπηρεσίες αυτές λειτουργούν είτε ως μέρος κάποιου στοιχείου (component) του SQL Server, είτε αυτόνομα ως υπηρεσία των Windows η οποία χρησιμοποιεί το δικό της API για να αλληλεπιδρά με αυτές.

Οι υπηρεσίες αυτές είναι οι εξής:

- Service Broker: Παρέχει μια αξιόπιστη πλατφόρμα ανταλλαγής δεδομένων για SQL Server εφαρμογές. Για τις cross instance εφαρμογές χρησιμοποιεί το πρωτόκολλο TCP/IP ώστε να μπορέσουν τα διαφορετικά στοιχεία (components) να συγχρονιστούν μεταξύ τους. μέσω της ανταλλαγής μηνυμάτων.
- Replication Services: Χρησιμοποιούνται για την αντιγραφή και τον συγχρονισμό αντικειμένων της βάσης δεδομένων, είτε εξ ολοκλήρου είτε ως υποσύνολο των υπαρχόντων αντικειμένων.
- Analysis Services: Προσφέρει την online επεξεργασία δεδομένων και υποστηρίζει τη λειτουργία εξόρυξης δεδομένων (data mining) για business intelligence εφαρμογές.
- Reporting Services: Επιβλέπει, μέσω μιας διαδικτυακής διεπαφής (web interface), δεδομένα που έχουν συγκεντρωθεί από μία SQL Server βάση δεδομένων.
- Integration Services: Χρησιμοποιείται για την ενσωμάτωση δεδομένων που προέρχονται από δεδομένα διαφορετικών προελεύσεων.
- Full Text Search Service: Προσφέρει εξειδικευμένη αναζήτηση κειμένου που είναι αποθηκευμένο σε μια SQL Server βάση δεδομένων.

Κεφάλαιο 4. Περιγραφή Συστήματος

4.1 Περιγραφή προβλήματος

Στην εποχή που ζούμε είναι σημαντικό οι επιχειρήσεις να είναι ευέλικτες, εύκολα προσιτές στους πελάτες, εναρμονισμένες με την τεχνολογία και βέβαια να είναι σε θέση να παρακολουθούν την οικονομική εξέλιξη τους και τις πωλήσεις τους. Είναι πλέον γεγονός ότι πολλές εταιρίες εστίασης διαθέτουν πλέον πληροφοριακά συστήματα παραγγελιοληψίας καθώς και ηλεκτρονικές αναφορές παρακολούθησης των οικονομικών στοιχείων τους.

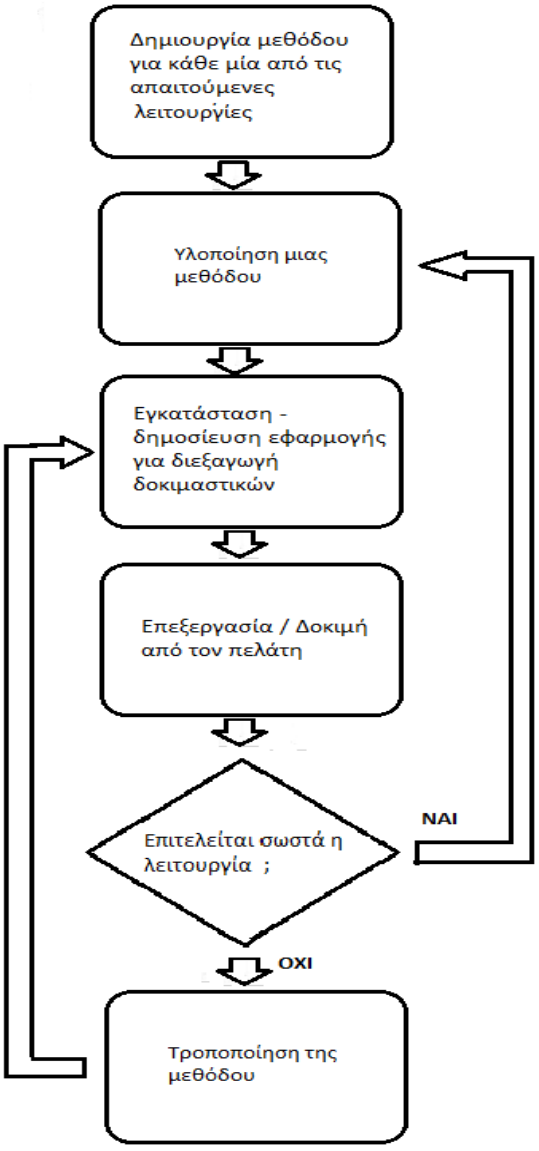
Η παρούσα εργασία ασχολείται με την ανάπτυξη ενός πληροφοριακού συστήματος που θα μεσολαβήσει ώστε να λυθούν προβλήματα που αφορούν στην επικοινωνία/ασυμβατότητα υποκαταστημάτων εστίασης με την κεντρική εταιρία. Πιο συγκεκριμένα, δύο είναι οι βασικοί λόγοι που οδήγησαν στην ανάπτυξη αυτού του συστήματος, οι οποίοι αναλύονται ακριβώς παρακάτω.

Πρώτον, λόγω ανταγωνιστικότητας, η κεντρική εταιρία που διαθέτει την αλυσίδα υποκαταστημάτων εστίασης θέλησε να δημιουργήσει μια ιστοσελίδα μέσω της οποίας ο πελάτης θα παραγγέλνει ηλεκτρονικά είτε για παράδοση κατ' οίκον είτε για παραλαβή από το κατάστημα. Υπήρχε επομένως ανάγκη να μεσολαβήσει ένα πληροφοριακό σύστημα μεταξύ της ιστοσελίδας και του πληροφοριακού συστήματος παραγγελιοληψίας ώστε να καταφέρουν να επικοινωνήσουν μεταξύ τους και να συγχρονιστούν οι βάσεις δεδομένων τους.

Δεύτερον, η κεντρική εταιρία επιθυμούσε να γνωρίζει τα έσοδα και τη γενική οικονομική πορεία του κάθε υποκαταστήματος της, και να διαχειρίζεται κεντρικά τους τιμοκαταλόγους και τις τιμές πώλησης των προϊόντών της καθώς και να γνωρίζει πότε και ποια υποκαταστήματα είναι ενεργά. Με αυτές τις πληροφορίες μπορεί να ελέγξει πλήρως τη λειτουργία και να δρομολογήσει αλλαγές εφόσον χρειάζονται. Και σε αυτή την περίπτωση υπήρχε ανάγκη για ένα πληροφοριακό σύστημα που να μεσολαβήσει ανάμεσα στο πληροφοριακό σύστημα παραγγελιοληψίας του κάθε υποκαταστήματος και στο διαχειριστικό-οικονομικό σύστημα της κεντρικής εταιρίας.

4.2. Μεθοδολογία Επίλυσης – Σχεδιασμού

Το πληροφοριακό σύστημα που ανέκυψε από τις ανάγκες της εταιρίας αυτής θα υλοποιηθεί με web services. Όπως αναλύθηκε και στο κεφάλαιο 2 η τεχνολογία των web services ενδείκνυται σε περιπτώσεις όπου δύο συστήματα είναι ασύμβατα τεχνολογικά αλλά επιθυμούν να ανταλλάξουν πληροφορία. Για την ανάπτυξη των web services ακολουθήθηκε η εξής μεθοδολογία:



Εικόνα 16: Μεθοδολογία επίλυσης προβλήματος

Με αυτόν τον τρόπο ο κώδικας 'χτίζεται' κομμάτι κομμάτι και μπορεί εύκολα να τροποποιηθεί χωρίς να χρειάζεται τροποποιήσεις όλο το πληροφοριακό σύστημα.

4.3 Τομείς Υλοποίησης

Με τη χρήση των τεχνολογιών που περιγράφηκαν στο Κεφάλαιο 3 , στο σημείο αυτό περιγράφεται η υλοποίηση και ανάπτυξη δύο Web Service που ονομάζονται InternetDeliveryService και Storemanager.

Το Web service **InternetDeiveryService** δημιουργήθηκε για τη διασύνδεση διαφορετικών συστημάτων με ένα κεντρικό σύστημα παραγγελιοληψίας. Τα τρίτα συστήματα μπορεί να είναι WebSites ή mobile applications. Η πληροφορία που ανταλλάσσεται δίνει τη δυνατότητα στα τρίτα συστήματα:

- να καταχωρούν παραγγελίες στην κεντρική βάση δεδομένων του συστήματος
- να έχουν πρόσβαση στα στοιχεία καταχωρημένων πελατών όπως κατάστημα εξυπηρέτησης πελάτη, τιμοκατάλογος πελάτη , στοιχεία διεύθυνσης και λοιπά στοιχεία
- να δέχονται παραγγελίες μόνο για τα ανοιχτά καταστήματα και να μπορούν να ενημερώσουν τον πελάτη τους για τον εκάστοτε χρόνο εξυπηρέτησης του
- να ενημερώνονται για την εκάστοτε διαθεσιμότητα προϊόντων
- να γνωστοποιούν στον πελάτη εφόσον το ζητήσει το στάδιο που βρίσκεται η παραγγελία του

Όλες οι παραπάνω λειτουργίες γίνονται On demand σε πραγματικό χρόνο και αποτελούν διαφορετικές κλήσεις του Web Service InternetDeliveryService που θα περιγραφούν παρακάτω.

Το Web Service Storemanager δημιουργήθηκε με σκοπό την ενημέρωση ενός κεντρικού server μιας εταιρίας με δεδομένα από τα υποκαταστήματα της. Πιο συγκεκριμένα το web service αυτό δινει τη δυνατότητα:

- να ενημερώνεται η κεντρική ΒΔ της εταιρίας με τις ωριαίες πωλήσεις των υποκαταστημάτων
- να ενημερώνεται η κεντρική ΒΔ της εταιρίας με οικονομικά στοιχεία καθημερινά από κάθε σημείο πώλησης κάθε υποκαταστήματος. Τέτοια στοιχεία είναι ο τζίρος, ο αριθμός αποδείξεων, ο αριθμός ειδών κ.α

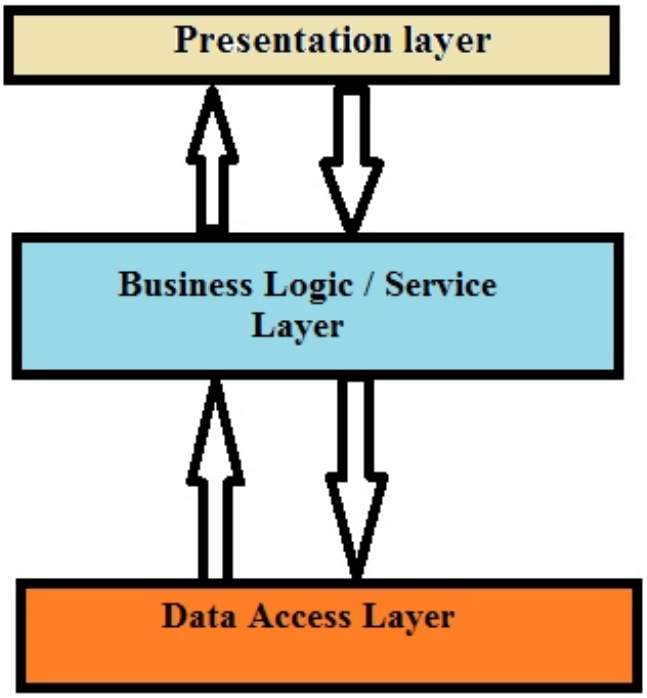
- να δίνει η κεντρική ΒΔ πληροφορία σε κάθε σημείο πώλησης κατά την έναρξη λειτουργίας του για το αν έχει αλλαγή στον τιμοκατάλογό του
- ο διαχειριστής της εταιρίας να γνωρίζει ποια σημεία πώλησης λειτουργούν ανά πάσα στιγμη
- να υπάρχει πλήρη εικόνα σχετικά με το πόσα σημεία πώλησης διαθέτει κάθε υποκατάστημα

4.4 Αρχιτεκτονική Συστήματος

Για την υλοποίηση του συστήματος επιλέχθηκε η αρχιτεκτονική τριών επιπέδων (three-tier architecture). Η αρχιτεκτονική τριών επιπέδων είναι μια αρχιτεκτονική client-server στην οποία η διεπαφή χρήστη (user interface), η επιχειρηματική λογική (business logic) και η αποθήκευση και πρόσβαση δεδομένων αναπτύσσονται και συντηρούνται ως ξεχωριστές ενότητες ή πιο συχνά σε ξεχωριστές πλατφόρμες.

Η αρχιτεκτονική αυτή αποτελείται από τα εξής επίπεδα:

- **Το επίπεδο παρουσίασης (presentation tier):** Είναι το αρχικό επίπεδο της εφαρμογής και αυτό με το οποίο αλληλεπιδρά ο τελικός χρήστης μέσω των στοιχείων διεπαφών χρήστη (user interfaces). Το επίπεδο αυτό περιλαμβάνει μηχανισμούς ελέγχου των στοιχείων που εισάγει ο χρήστης. Στην παρούσα εργασία τα δεδομένα εισάγονται από τον τελικό χρήστη μέσω κάποιας ιστοσελίδας.
- **Το επιχειρηματικό επίπεδο (business tier):** Το επίπεδο αυτό συντονίζει την εφαρμογή, επεξεργάζεται εντολές, αξιολογεί τα δεδομένα που δέχεται από το επίπεδο παρουσίασης και εκτελεί υπολογισμούς. Δέχεται αιτήσεις από το επίπεδο παρουσίασης και καλεί το επίπεδο δεδομένων ώστε να ανακτήσει πληροφορίες τις οποίες θα στείλει πίσω στο επίπεδο παρουσίασης. Εδώ γίνεται η επεξεργασία όλων των αιτημάτων / κλήσεων των κλάσεων του Web Service.
- **Το επίπεδο δεδομένων (data tier):** Στο επίπεδο αυτό τα δεδομένα αποθηκεύονται σε μια βάση δεδομένων ή σε έναν φάκελο του συστήματος. Η ζητούμενη πληροφορία ανακτάται από αυτό το επίπεδο και μεταβιβάζεται στο επιχειρηματικό επίπεδο το οποίο με τη σειρά του την προωθεί στο επίπεδο παρουσίασης.

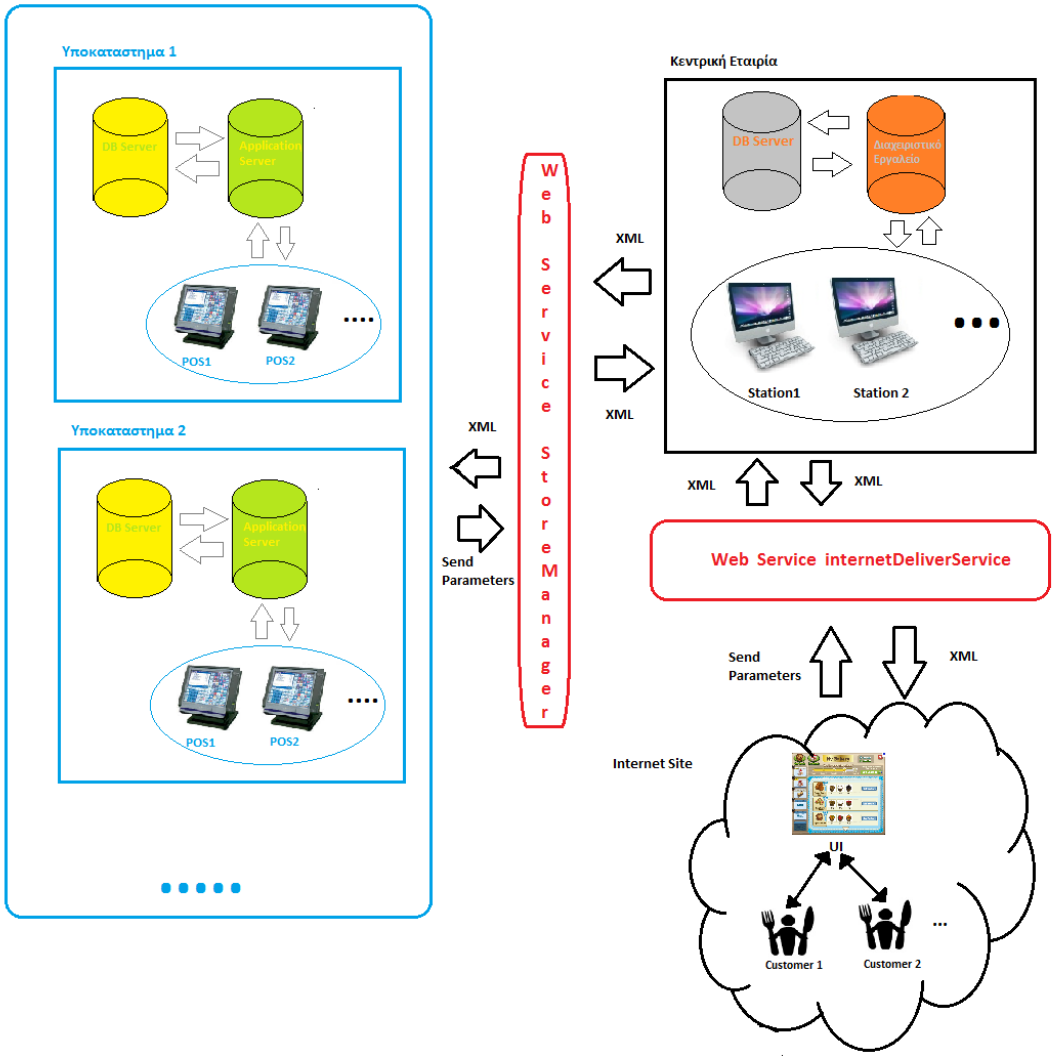


Εικόνα 17: Αρχιτεκτονική Συστήματος

Τα κυριότερα πλεονεκτήματα που προσφέρει η αρχιτεκτονική τριών επιπέδων σε σχέση με μια κλασική εφαρμογή client-server δύο επιπέδων, είναι τα εξής:

- Συντηρησιμότητα: Κάθε επίπεδο είναι ανεξάρτητο από τα υπόλοιπα με αποτέλεσμα οι ενημερώσεις και οι αλλαγές να μπορούν να πραγματοποιηθούν χωρίς να επηρεάζουν την εφαρμογή στο σύνολό της.
- Επεκτασιμότητα: Η δομή της αρχιτεκτονικής αυτής καθιστά εύκολη την αναβάθμιση της εφαρμογής, αφού κάθε επίπεδο αναπτύσσεται και συντηρείται ως ξεχωριστή ενότητα.
- Ευελιξία: Αυξάνεται η ευελιξία, επειδή κάθε επίπεδο διαχειρίζεται και αναπτύσσεται ανεξάρτητα από τα υπόλοιπα.

Λειτουργικά η αρχιτεκτονική της εφαρμογής που αναπτύχθηκε αναπαρίσταται στο παρακάτω σχήμα.



Εικόνα 18 Αρχιτεκτονική Εφαρμογής

4.5 Ιεραρχία των Διαδικτυακών Υπηρεσιών

Μία σημαντική προσέγγιση της λειτουργίας της εφαρμογής θα φανεί εφ'όσον αναλυθεί η ιεραρχία των διαδικτυακών υπηρεσιών που δημιουργήθηκαν στην παρούσα εργασία. Η προσέγγιση θα γίνει κατά Rosen.

Σύμφωνα με την προσεγγιση των Rosen *et al.,* (2008) στο ανώτερο επίπεδο της ιεραρχίας βρίσκονται οι επιχειρησιακές διαδικασίες. Αυτές εν συνεχεία διασπόνται σε μικρότερα τμήματα και αντιστοιχίζονται σε επιχειρησιακές υπηρεσίες. Σε κατώτερο επίπεδο βρίσκονται οι τοπικές υπηρεσίες (domain services), οι υπηρεσίες ολοκλήρωσης (integration services), οι εξωτερικές υπηρεσίες (external services), οι υπηρεσίες υποστήριξης (utility services) και οι βασικές υπηρεσίες (foundation services). Παρακάτω περιγράφονται αναλυτικά τα χαρακτηριστικά που καθορίζουν την κάθε ομάδα διαδικτυακών υπηρεσιών.

- **Επιχειρησιακές Υπηρεσίες (Business Services) :** Πρόκειται για υψηλού επιπέδου υπηρεσίες που συνθέτουν τη λειτουργικότητα τους από web services που βρίσκονται σε κατώτερα επίπεδα από αυτές. Ο σκοπός τους είναι διττός. Αφενός προσφέρουν επιχειρησιακή λειτουργικότητα που μπορεί να χρησιμοποιηθεί από μία ή περισσότερες διαδικασίες είτε εντός είτε εκτός οργανισμού. Αφετέρου επιτρέπουν σε εφαρμογές να χρησιμοποιήσουν κοινές λειτουργίες προκειμένου να εξασφαλιστεί ένας δομημένος και συνεπής τρόπος συμπεριφοράς. Οι επιχειρησιακές υπηρεσίες είναι οι πιο σύνθετες από όλες τις διαδικτυακές υπηρεσίες και έχουν υψηλό βαθμό λειτουργικότητας.

- **Τοπικές Υπηρεσίες (Domain Services):** Οι τοπικές υπηρεσίες έχουν μεσαίο βαθμό λειτουργικότητας και παρέχουν υπηρεσίες που σχετίζονται με μία συγκεκριμένη περιοχή και συνήθως χρησιμοποιούνται εντός της περιοχής αυτής (π.χ. επιβεβαίωση κράτησης θέσης σε ταξίδι). Η υπηρεσία αυτή συνήθως χρησιμοποιείται στην περιοχή που ορίζεται (σπάνια χρησιμοποιείται εκτός της περιοχής αυτής).

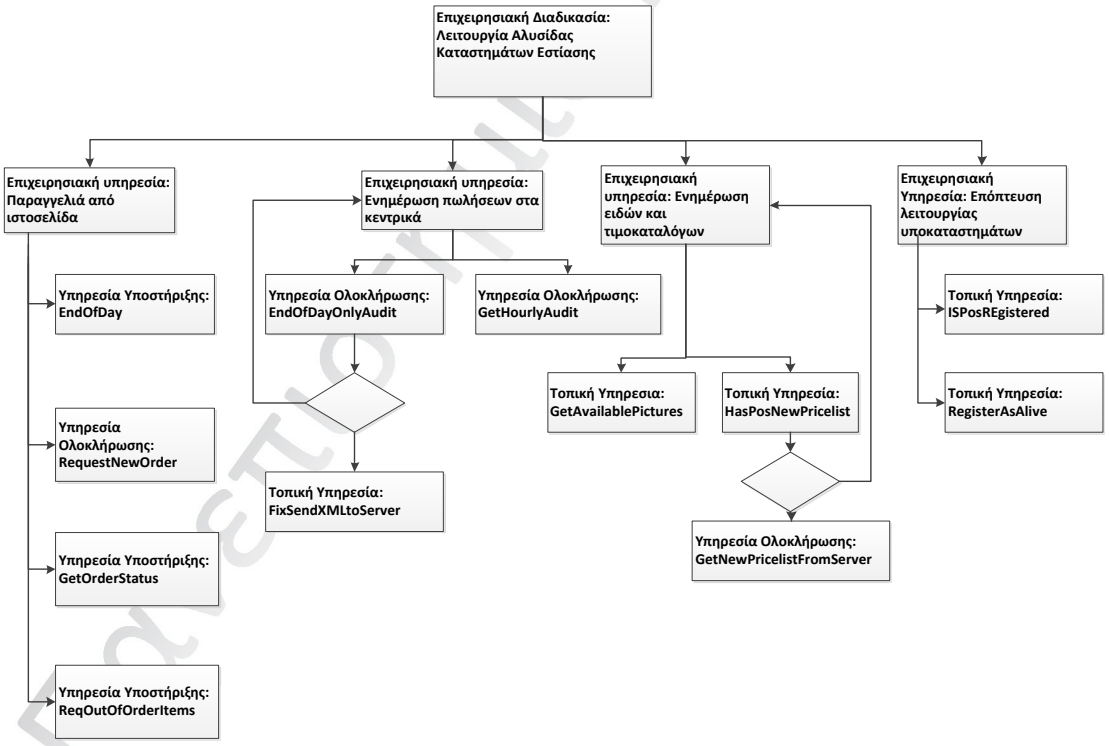
- **Υπηρεσίες Ολοκλήρωσης (Integration Services):** Οι υπηρεσίες αυτού του τύπου έχουν ως στόχο την ολοκλήρωση υπαρχόντων εφαρμογών. Στις βασικές λειτουργίες τους συγκαταλέγεται η συνεπής πρόσβαση σε διαφορετικές πηγές δεδομένων, η εξασφάλιση επικοινωνίας με συγκεκριμένες εφαρμογές, η μετάφραση στοιχείων από και προς τις εφαρμογές που ολοκληρώνονται κλπ. Ο βαθμός πολυπλοκότητας της λειτουργίας των υπηρεσιών ολοκλήρωσης εξαρτάται από τη φύση των εφαρμογών που ολοκληρώνουν.

- **Εξωτερικές Υπηρεσίες (External Services):** Οι εξωτερικές υπηρεσίες διασφαλίζουν την επικοινωνία μεταξύ εφαρμογών που βρίσκονται στο εξωτερικό περιβάλλον του οργανισμού με το οποίο συνεργάζονται. Για παράδειγμα, στην κατηγορία αυτή μπορεί να ενταχθεί μία υπηρεσία που επιτρέπει σε ένα μια υπηρεσία ηλεκτρονικών πληρωμών και εκκαθαρίζει πληρωμές που γίνονται μέσω του συστήματος PayPal. Παρομοίως με τις υπηρεσίες ολοκλήρωσης έτσι και στις εξωτερικές υπηρεσίες, το

πλήθος των λειτουργιών τους εξαρτάται από τις εφαρμογές (εξωτερικές σε αυτή την περίπτωση) που ολοκληρώνουν.

- **Υπηρεσίες Υποστήριξης (Utility Services):** Οι υπηρεσίες υποστήριξης έχουν χαμηλό βαθμό λειτουργικότητας και παρέχουν βασική λειτουργικότητα που είναι σημαντική για αρκετές υπηρεσίες και διαδικασίες. Ένα παράδειγμα τέτοιας υπηρεσίας αφορά το web service πελατολόγιο, το οποίο έχει πρόσβαση σε όλα τα στοιχεία των πελατών και μπορεί να τα ανακτήσει σε περίπτωση που του ζητηθεί.
- **Βασικές Υπηρεσίες (Foundation Services):** Οι υπηρεσίες αυτού του τύπου αναφέρονται σε απλές υπηρεσίες που μπορούν να χρησιμοποιηθούν για τη σύνθεση, την ενορχήστρωση και χορογραφία υπηρεσιών υψηλότερου επιπέδου.

Σύμφωνα με τους παραπάνω ορισμούς το διάγραμμα που περιγράφει την ιεραρχία και τη διασύνδεση των διαδικτυακών υπηρεσιών της παρούσας εργασίας φαίνεται παρακάτω.



Εικόνα 19 Ιεραρχία των Web Services της εφαρμογής

4.6 Σχεδίαση και υλοποίηση της βάσης δεδομένων

Εδώ θα περιγραφεί η βάση δεδομένων του συστήματος καθώς η μορφή της καθορίζει σε μεγάλο βαθμό την συνολική οργάνωση του συστήματος. Η ανάπτυξη της βάσης δεδομένων έγινε με γνώμονα την απλότητα, την ευχρηστία, την πληρότητα αλλά και την έλλειψη επανάληψης πληροφορίας. Αρχικά, θα ερμηνευτεί η χρησιμότητα κάθε πίνακα και ακολούθως θα φαίνονται όλοι οι πίνακες με τις επιμέρους ιδιότητές τους καθώς και οι σχέσεις μεταξύ των πινάκων.

Οι πίνακες της βάσης είναι οι εξής:

4.6.1.Πίνακας Items

Αναλυτικά τα πεδία που περιέχει ο πίνακας είναι: **Codart, Desca, Descb, Price1, Kitchen, Vat1, Ggroup, Ttime, Contiment, Recipe, active, correlation.**

Πρόκειται για τον πίνακα που περιέχει πληροφορίες για όλα τα προϊόντα που μπορούν να πωληθούν, καθώς και πληροφορίες σχετικά με αυτά όπως κωδικό του προϊόντος (codart), την κατηγορία ΦΠΑ που ανήκει (Vat1), την περιγραφή αυτού (desca,descb), διαφορετικές τιμές ανά τιμοκατάλογο (price1), την ομάδα που ανήκει (π.χ Pizzas ή Sandwich κτλ- ggroup) , το χρόνο προετοιμασίας του (Ttime) κ.τ.λ. Το πρωτεύον κλειδί αυτού του πίνακα είναι ο κωδικός του προϊόντος δηλαδή το πεδίο Codart.

Column Name	Data Type	Allow Nulls
Codart	int	☐
Desca	nvarchar(50)	☑
DescB	nvarchar(50)	☑
GGroup	int	☑
Kitchen	int	☑
TTime	smalldatetime	☑
Recipe	bit	☑
Active	bit	☑
Contiment	bit	☑
correlation	int	☑
Price1	money	☑
Vat1	int	☑

Εικόνα 20: Πίνακας Items

4.6.2. Πίνακας Labcustomer

Αναλυτικά τα πεδία αυτού του πίνακα είναι: **Customerid, Name, Fname, Title, Profession, Tel1, Address1, Address_No, Orofos1, Orofos2, City, Zipcode, VIP, Tomeas, Sector, Bonus, Store, Last_Order, First_Order, Order_Comments.**

Αυτός ο πίνακας περιέχει όλες τις πληροφορίες που αφορούν τα στοιχεία του εκάστοτε πελάτη όπως το όνομα του (Name), το επίθετό του (Fname), τη διεύθυνση του (Address1, address_No, City, Zipcode), το κατάστημα που τον εξυπηρετεί (Store), την ημερομηνία που έκανε την πρώτη του παραγγελία (First_Order) αλλά και την πιο πρόσφατη ημερομηνία που παρήγγειλε (Last_Order) κ.τ.λ. Κάθε πελάτης έχει ένα μοναδικό κωδικό, το πεδίο Customerid, το οποίο είναι και το πρωτεύον κλειδί για αυτόν τον πίνακα.

Column Name	Data Type	Allow Nulls
Customerid	decimal(18, 0)	☐
Name	char(50)	☑
Fname	char(50)	☑
Title	nvarchar(50)	☑
Profession	nvarchar(50)	☑
Tel1	nvarchar(20)	☑
Address1	char(50)	☑
Address_No	nvarchar(50)	☑
Orofos1	nvarchar(10)	☑
Orofos2	nvarchar(10)	☑
City	char(50)	☑
Zipcode	char(10)	☑
VIP	nchar(5)	☑
Store	int	☑
Tomeas	nvarchar(10)	☑
Sector	nvarchar(10)	☑
Order_Comments	nvarchar(200)	☑
First_Order	datetime	☑
Last_Order	datetime	☑
Bonus	int	☑

Εικόνα 21: Πίνακας Labcustomer

4.6.3. Πίνακας Kdsfb_Hq

Τα πεδία του πίνακα αυτού είναι: **Id, Orderno, Pos, Shop_Id, Item_Group, Item_Code, Item_Descr, Item_Subgroup, Item_Vat, Sp, Qty, Amount, Total, Agent, Listino, Receipt, Kdws, Status, Status_Time, Room, Payment, Type, Comments, Fo_Day**.

Στον πίνακα αυτό καταχωρούνται οι παραγγελίες που πραγματοποιούνται από τους πελάτες αλλά και από άλλες ομάδες χειριστών (όπως τηλεφωνητές, υπάλληλοι καταστημάτων κ.τ.λ) καθώς και οι λεπτομέρειες που τις αφορούν όπως ο αριθμός παραγγελίας (Orderno), ο κωδικός του προϊόντος (Item_Code), η τιμή του (Amount), η ποσότητα (Qty), ο κωδικός του καταστήματος που αφορά η εκάστοτε παραγγελία (Shop_Id), ο κωδικός του πελάτη που παρήγγειλε (Room), τυχόν σχόλια για αυτή την παραγγελία (Remarks), καθώς και το πεδίο Status της παραγγελίας (1->Έχει εκτυπωθεί στην κουζίνα, 2-> Έτοιμη άρα μπορεί να εκδοθεί απόδειξη, 3-> Έχει εκδοθει απόδειξη, 4-> Ο οδηγός την παρέδωσε, 6-> Ακυρώθηκε). Εδώ το πρωτεύον κλειδί είναι το πεδίο Id.

Column Name	Data Type	Allow Nulls
Id	int	☐
Orderno	nvarchar(50)	☑
Pos	int	☑
Shop_Id	int	☑
Item_Group	int	☑
Item_Code	int	☑
Item_Descr	nvarchar(50)	☑
Item_Subgroup	int	☑
Item_Vat	int	☑
Sp	nchar(1)	☑
Qty	int	☑
Amount	int	☑
Total	int	☑
Agent	int	☑
Listino	int	☑
Receipt	int	☑
Kdws	int	☑
Status	int	☑
Status_Time	datetime	☑
Room	int	☑
Payment	nvarchar(50)	☑
Type	nvarchar(50)	☑
Comments	nvarchar(255)	☑
Fo_Day	datetime	☑

Εικόνα 22: Πίνακας kdsfb_hq

4.6.4. Πίνακας OutOfStock

Τα πεδία αυτού του πίνακα είναι: **id, Itemcode, Timetocarryout, Timetodeliver, Status, Storeid.**

Εδώ καταχωρούνται πληροφορίες σχετικές με το κατάστημα δηλαδή το χρόνο που δηλώνει το κάθε κατάστημα ότι κάνει παράδοση στο σπίτι (Tmetodeliver) ή στο κατάστημα (Timetocarryout), ποια είδη έχει έλλειψη ώστε να μην δίνονται τέτοιες παραγγελίες (itemcode) καθώς και το status του κάθε καταστήματος (π.χ 1->Ανοικτό και 2->Κλειστό). Πρωτεύον κλειδί εδώ είναι το Id.

Column Name	Data Type	Allow Nulls
id	int	☐
Itemcode	int	☑
Timetocarryout	datetime	☑
Timetodeliver	datetime	☑
status	int	☑
Storeid	int	☑

Εικόνα 23: Πίνακας Outofstock

4.6.5.Πίνακας Dominos

Τα πεδία αυτού του πίνακα είναι τα εξής: **Shop_Id, Shop_Name, Address, Number, Area, City, ZipCode, Tel, Ip, Socket, Status, Active, Orderno.**

Σε αυτόν τον πίνακα καταχωρούνται στοιχεία από όλα τα καταστήματα όπως το όνομα του καταστήματος (shop_name), η διεύθυνση (Address, Number,City, ZipCode), το τηλέφωνο (Tel) κ.τ.λ. Κάθε κατάστημα έχει μοναδικό κωδικό (Shop_Id) που είναι και το πρωτεύον κλειδί του.

Column Name	Data Type	Allow Nulls
Shop_Id	int	☐
Shop_Name	char(25)	☑
Address	char(50)	☑
Number	int	☑
Area	char(50)	☑
City	char(50)	☑
ZipCode	char(10)	☑
Tel	char(20)	☑
Ip	char(20)	☑
Socket	int	☑
Status	bit	☑
Active	int	☑
Orderno	int	☑

Εικόνα 24: Πίνακας Dominos

4.6.6. Πίνακας Orders

Τα πεδία του πίνακα αυτού είναι: **Id, Orderno, Pos, Shop_Id, Item_Group, Item_Code, Item_Descr, Item_Subgroup, Item_Vat, Sp, Qty, Amount, Total, Agent, Listino, Receipt, Kdws, Status, Status_Time, Room, Payment, Type, Comments, Fo_Day**.

Στον πίνακα αυτό καταχωρούνται οι παραγγελίες που πραγματοποιούνται από τους πελάτες καθώς και οι λεπτομέρειες που τις αφορούν όπως ο αριθμός παραγγελίας (Orderno), ο κωδικός του προϊόντος (Item_Code), η τιμή του (Amount), η ποσότητα (Qty), ο κωδικός του καταστήματος που αφορά η εκάστοτε παραγγελία (Shop_Id), ο κωδικός του πελάτη που παρήγγειλε (Room), τυχόν σχόλια για αυτή την παραγγελία (Remarks), καθώς και το πεδίο Status της παραγγελίας (1->Έχει εκτυπωθεί στην κουζίνα, 2-> Έτοιμη άρα μπορεί να εκδοθεί απόδειξη, 3-> Έχει εκδοθει απόδειξη, 4-> Ο οδηγός την παρέδωσε, 6-> Ακυρώθηκε). Εδώ το πρωτεύον κλειδί είναι το πεδίο Id. Η διαφορά του με τον πίνακα kdsfb_hq είναι ότι εδώ καταχωρούνται μόνο παραγγελίες κατευθείαν από πελάτες. Η δομή του φαίνεται παρακάτω:

Column Name	Data Type	Allow Nulls
Id	int	☐
Orderno	nvarchar(50)	☑
Pos	int	☑
Shop_Id	int	☑
Item_Group	int	☑
Item_Code	int	☑
Item_Descr	nvarchar(50)	☑
Item_Subgroup	int	☑
Item_Vat	int	☑
Sp	nchar(1)	☑
Qty	int	☑
Amount	int	☑
Total	int	☑
Agent	int	☑
Listino	int	☑
Receipt	int	☑
Kdws	int	☑
Status	int	☑
Status_Time	datetime	☑
Room	int	☑
Payment	nvarchar(50)	☑
Type	nvarchar(50)	☑
Comments	nvarchar(255)	☑
Fo_Day	datetime	☑

Εικόνα 25: Πίνακας Orders

4.6.7. Πίνακας EndofDay

Τα πεδία αυτού του πίνακα είναι: **Id, FoDAy, CloseId, Gross, Net, TicketsCount, ItemCountd, TicketAverage,Pos, Status, FinalisationDAte, dayAuditId, LabFBTotal** και **PosInfoId.**

Σε κάθε κλείσιμο ημέρας δημιουργείται στον πίνακα μια καινουρια γραμμή που περιέχει πληροφορίες όπως η ημερομηνία κλεισίματος (FODAY), τα καθαρά κέρδη (NET), τα μεικτά κέρδη (GROSS), ο αριθμός αποδείξεων(TicketsCount), το μέσο ποσό ανά απόδειξη κ.α. Πρόκειται για μια σύντομη και χρήσιμη πληροφορία.

Column Name	Data Type	Allow Nulls
Id	uniqueidentifier	☐
FODay	datetime	☑
CloseId	int	☑
Gross	int	☑
Net	money	☑
TicketsCount	int	☑
ItemCount	int	☑
TicketAverage	numeric(9, 2)	☑
Pos	nvarchar(50)	☐
Status	tinyint	☑
FinalisationDate	datetime	☑
DayAuditId	uniqueidentifier	☑
LabFBTotal	int	☑
PosInfoId	uniqueidentifier	☑

Εικόνα 26: Πίνακας EndOfDay

4.6.8. Πίνακας HourAuditDetail

Τα πεδία αυτού του πίνακα είναι: **Id, HourAuditId, CloseID, Gross, Net, TicketsCount, ItemsCount, TicketAverage, Hour**

Ο πίνακας αυτός περιέχει λεπτομέρειες πωλήσεων ανά ώρα. Τέτοιες λεπτομέρειες είναι ο αριθμός αποδείξεων (TicketsCount), τα καθαρά (NET) και τα μεικτά (Gross) κέρδη, το σύνολο ειδών (ItemsCount) πάντα ανά ώρα πώλησης (Hour).

Column Name	Data Type	Allow Nulls
Id	uniqueidentifier	☐
HourAuditId	uniqueidentifier	☑
CloseId	int	☑
Gross	money	☑
Net	money	☑
TicketsCount	int	☑
ItemsCount	int	☑
TicketAverage	numeric(9, 2)	☑
Hour	tinyint	☑

Εικόνα 27: Πίνακας HourAuditDetails

4.6.9. Πίνακας Scheduler

Τα πεδία αυτού του πίνακα είναι: **Id, ShopInfoId, StoreId, UpdateDate, Status, ChangeOrder, ChangeSet**

Ο πίνακας αυτός αφορά στις αλλαγές τιμολαταλόγου. Τα πεδία του είναι η ημερομηνία (UpdateDate)που πρέπει να γίνει η ενημέρωση τιμών, το κατάστημα (StoreId) καθώς και αν τελικά έχει γίνει ή όχι η αλλαγή (Status).

Column Name	Data Type	Allow Nulls
Id	uniqueidentifier	☐
ShopInfoId	uniqueidentifier	☑
StoreId	int	☑
UpdateDate	datetime	☑
Status	tinyint	☑
ChangeOrder	int	☑
Changeset	bigint	☑
		☐

Εικόνα 28:Πίνακας Scheduler

4.6.10 Πίνακας PosInfo

Τα πεδία αυτού του πίνακα είναι: **Id, ShopInfo, PosNo,PosUrl**.

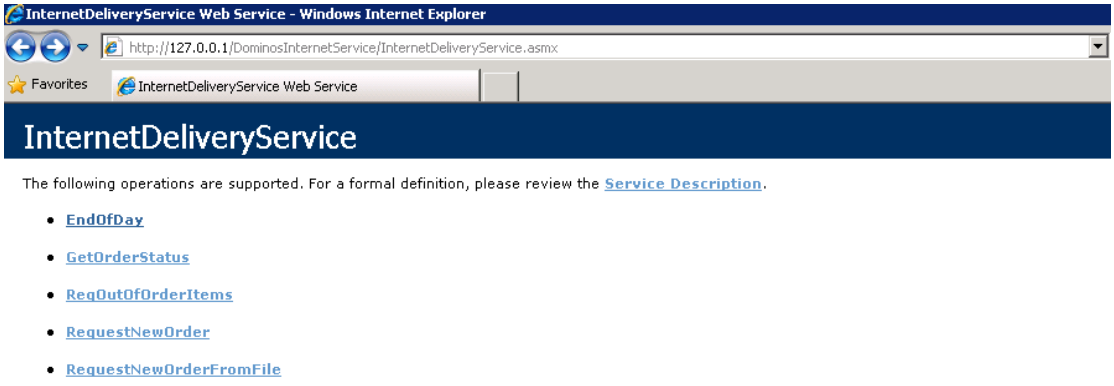
Σε αυτόν τον πίνακα καταγράφονται οι πληροφορίες για κάθε σημείο πώλησης της αλυσίδας. Συγκεκριμμένα, για κάθε σημείο πώλησης (PosNo) αναφέρεται το υποκατάστημα (ShopInfoId) στο οποίο ανήκει καθώς και η ip διεύθυνσή του (PosUrl).

Column Name	Data Type	Allow Nulls
Id	uniqueidentifier	☐
ShopInfoId	uniqueidentifier	☑
PosNo	nvarchar(50)	☑
PosUri	nvarchar(500)	☑
		☐

Εικόνα 29: Πίνακας PosInfo

4.7 Μέθοδοι του Web Service InternetDeliveryService

Η λίστα των μεθόδων για το συγκεκριμένο WebService φαίνεται στο παρακάτω σχήμα. Η κλήσεις γίνονται χειροκίνητα, με χρήση κάποιων παραμέτρων εισόδου που περιγράφονται αναλυτικά για κάθε μία.



Εικόνα 30:Λίστα Μεθόδων InternetDeliveryService

Όπως κάθε Web Service έτσι και σε αυτό έχουμε την περιγραφή του (WSDL) που φαίνεται στο παρακάτω πίνακα.

```
<?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://hit.com.gr/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" targetNamespace="http://hit.com.gr/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
- <wsdl:types>
- <s:schema elementFormDefault="qualified" targetNamespace="http://hit.com.gr/">
- <s:element name="GetOrderStatus">
- <s:complexType>
- <s:sequence>
 <s:element minOccurs="1" maxOccurs="1" name="orderNo" type="s:int" />
 </s:sequence>
 </s:complexType>
 </s:element>
- <s:element name="GetOrderStatusResponse">
- <s:complexType>
- <s:sequence>
 <s:element minOccurs="1" maxOccurs="1" name="GetOrderStatusResult" type="s:int" />
 </s:sequence>
 </s:complexType>
 </s:element>
- <s:element name="ReqOutOfOrderItems">
- <s:complexType>
- <s:sequence>
 <s:element minOccurs="1" maxOccurs="1" name="StoreId" type="s:int" />
 </s:sequence>
 </s:complexType>
 </s:element>
```

```
- <s:element name="ReqOutOfOrderItemsResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="ReqOutOfOrderItemsResult" type="s:string"
/>
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="RequestNewOrder">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="RequestNewOrder" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="RequestNewOrderResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="RequestNewOrderResult" type="s:boolean"
/>
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="EndOfDay">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="Day" type="s:dateTime" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="EndOfDayResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="EndOfDayResult" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="RequestNewOrderFromFile">
  <s:complexType />
  </s:element>
- <s:element name="RequestNewOrderFromFileResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="RequestNewOrderFromFileResult"
type="s:boolean" />
  </s:sequence>
  </s:complexType>
  </s:element>
  </s:schema>
  </wsdl:types>
- <wsdl:message name="GetOrderStatusSoapIn">
  <wsdl:part name="parameters" element="tns:GetOrderStatus" />
  </wsdl:message>
- <wsdl:message name="GetOrderStatusSoapOut">
  <wsdl:part name="parameters" element="tns:GetOrderStatusResponse" />
  </wsdl:message>
- <wsdl:message name="ReqOutOfOrderItemsSoapIn">
  <wsdl:part name="parameters" element="tns:ReqOutOfOrderItems" />
  </wsdl:message>
```

```
- <wsdl:message name="ReqOutOfOrderItemsSoapOut">
  <wsdl:part name="parameters" element="tns:ReqOutOfOrderItemsResponse" />
  </wsdl:message>
- <wsdl:message name="RequestNewOrderSoapIn">
  <wsdl:part name="parameters" element="tns:RequestNewOrder" />
  </wsdl:message>
- <wsdl:message name="RequestNewOrderSoapOut">
  <wsdl:part name="parameters" element="tns:RequestNewOrderResponse" />
  </wsdl:message>
- <wsdl:message name="EndOfDaySoapIn">
  <wsdl:part name="parameters" element="tns:EndOfDay" />
  </wsdl:message>
- <wsdl:message name="EndOfDaySoapOut">
  <wsdl:part name="parameters" element="tns:EndOfDayResponse" />
  </wsdl:message>
- <wsdl:message name="RequestNewOrderFromFileSoapIn">
  <wsdl:part name="parameters" element="tns:RequestNewOrderFromFile" />
  </wsdl:message>
- <wsdl:message name="RequestNewOrderFromFileSoapOut">
  <wsdl:part name="parameters" element="tns:RequestNewOrderFromFileResponse" />
  </wsdl:message>
- <wsdl:portType name="InternetDeliveryServiceSoap">
- <wsdl:operation name="GetOrderStatus">
  <wsdl:input message="tns:GetOrderStatusSoapIn" />
  <wsdl:output message="tns:GetOrderStatusSoapOut" />
  </wsdl:operation>
- <wsdl:operation name="ReqOutOfOrderItems">
  <wsdl:input message="tns:ReqOutOfOrderItemsSoapIn" />
  <wsdl:output message="tns:ReqOutOfOrderItemsSoapOut" />
  </wsdl:operation>
- <wsdl:operation name="RequestNewOrder">
  <wsdl:input message="tns:RequestNewOrderSoapIn" />
  <wsdl:output message="tns:RequestNewOrderSoapOut" />
  </wsdl:operation>
- <wsdl:operation name="EndOfDay">
  <wsdl:input message="tns:EndOfDaySoapIn" />
  <wsdl:output message="tns:EndOfDaySoapOut" />
  </wsdl:operation>
- <wsdl:operation name="RequestNewOrderFromFile">
  <wsdl:input message="tns:RequestNewOrderFromFileSoapIn" />
  <wsdl:output message="tns:RequestNewOrderFromFileSoapOut" />
  </wsdl:operation>
  </wsdl:portType>
- <wsdl:binding name="InternetDeliveryServiceSoap" type="tns:InternetDeliveryServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="GetOrderStatus">
  <soap:operation soapAction="http://hit.com.gr/GetOrderStatus" style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="ReqOutOfOrderItems">
  <soap:operation soapAction="http://hit.com.gr/ReqOutOfOrderItems" style="document" />
- <wsdl:input>
  <soap:body use="literal" />
```

```
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="RequestNewOrder">
  <soap:operation soapAction="http://hit.com.gr/RequestNewOrder" style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="EndOfDay">
  <soap:operation soapAction="http://hit.com.gr/EndOfDay" style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="RequestNewOrderFromFile">
  <soap:operation soapAction="http://hit.com.gr/RequestNewOrderFromFile" style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="InternetDeliveryServiceSoap12" type="tns:InternetDeliveryServiceSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="GetOrderStatus">
  <soap12:operation soapAction="http://hit.com.gr/GetOrderStatus" style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="ReqOutOfOrderItems">
  <soap12:operation soapAction="http://hit.com.gr/ReqOutOfOrderItems" style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="RequestNewOrder">
  <soap12:operation soapAction="http://hit.com.gr/RequestNewOrder" style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
  </wsdl:input>
```

```
- <wsdl:output>
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="EndOfDay">
  <soap12:operation soapAction="http://hit.com.gr/EndOfDay" style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="RequestNewOrderFromFile">
  <soap12:operation soapAction="http://hit.com.gr/RequestNewOrderFromFile" style="document"
/>
- <wsdl:input>
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="InternetDeliveryService">
- <wsdl:port name="InternetDeliveryServiceSoap" binding="tns:InternetDeliveryServiceSoap">
  <soap:address location="http://127.0.0.1/DominosInternetService/InternetDeliveryService.asmx"
/>
  </wsdl:port>
- <wsdl:port name="InternetDeliveryServiceSoap12"
binding="tns:InternetDeliveryServiceSoap12">
  <soap12:address
location="http://127.0.0.1/DominosInternetService/InternetDeliveryService.asmx" />
  </wsdl:port>
  </wsdl:service>
  </wsdl:definitions>
```

Πίνακας 1: WSDL Web Service

Κάθε μία από τις κλήσεις αναλύεται λεπτομερώς παρακάτω.

4.7.1. EndofDay

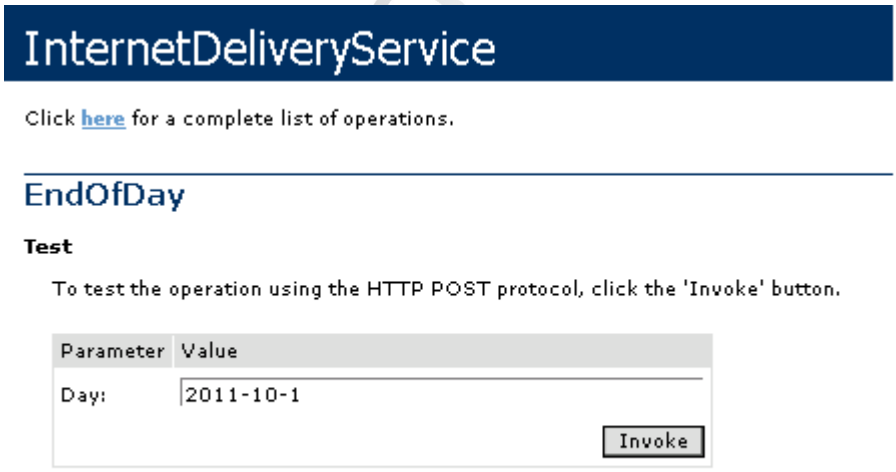
Όταν πραγματοποιείται η κλήση αυτή δίνεται σαν είσοδος μία ημερομηνία υπό την μορφή YYYY-MM-DD και επιστρέφεται ένα xml που έχει πληροφορίες για τους πελάτες που παρήγγειλαν εκείνη την ημέρα και τον κατάλογο των ειδών που ήταν διαθέσιμα τότε.

EndofDay			
	Περιγραφή	Τύπος	Παράδειγμα

Είσοδος	Ημερομηνία Ενδιαφέροντος	datetime	2011-10-1
Έξοδος	Επιστροφή Δεδομένων για την ημερομηνία ενδιαφέροντος	string	<EndOfDay> <CurrentDate>2011-10-01T00:00:00</CurrentDate> <Customers> <labcustomer CUSTOMERID="1" NAME="ΜΕΝΕΛΑΟΣ " FNAME="ΜΑΡΚΟΥ " TITLE="" PROFESSION="" TEL1="2101234567" ADDRESS1="ΜΕΤΣΟΒΟΥ " ADDRESS_NO="23" OROFOS1="ΙΣ" OROFOS2="" CITY=" " ZIPCODE=" " VIP="1 " TOMEAS="" SECTOR="A23" BONUS="10" STORE="1" LAST_ORDER="2011-10-01T00:00:00" …

Πίνακας 2: Είσοδος/'Εξοδος Endofday

Ένα παράδειγμα φαίνεται παρακάτω:



<?xml version="1.0" encoding="utf-8" ?>
string xmlns="http://hit.com.gr/"><?xml version="1.0" encoding="utf-8" standalone="yes"?><EndOfDay>
<CurrentDate>2011-10-01T00:00:00</CurrentDate> <Customers> <labcustomer CUSTOMERID="1"
NAME="ΜΕΝΕΛΑΟΣ " FNAME="ΜΑΡΚΟΥ " TITLE="" PROFESSION="" TEL1="2101234567"

ADDRESS1="ΜΕΤΣΟΒΟΥ " ADDRESS_NO="23" OROFOS1="ΙΣ" OROFOS2="" CITY=" " ZIPCODE=" " VIP="1 " TOMEAS="" SECTOR="A23" BONUS="10" STORE="1" LAST_ORDER="2011-10-01T00:00:00" ORDER_COMMENTS="ΟΛΑ ΑΝΑΛΑΤΑ " /> </Customers> <Items>2011-10-01T00:00:00<item codart="1" desca="ESPRESSO" descb="ESPRESSO" price1="300" kitchen="0" vat1="2" ggroup="2" ttime="" contiment="false" recipe="false" active="true" correlation="" /><item codart="2" desca="ESPRESSO MACCHIATTO" descb="ESPRESSO MACCHIATTO" price1="310" kitchen="0" vat1="2" ggroup="2" ttime="" contiment="false" recipe="false" active="true" correlation="" /><item codart="3" desca="ESPRESSO CON PANNA" descb="ESPRESSO CON PANNA" price1="310" kitchen="0" vat1="2" ggroup="2" ttime="" contiment="false" recipe="false" active="true" correlation="" /><item codart="4" desca="ESPRESSO CORETTO" descb="ESPRESSO CORETTO" price1="350" kitchen="0" vat1="2" ggroup="2" ttime="" contiment="false" recipe="false" active="true" correlation="" /><item codart="5" desca="ESPRESSO DOPPIO" descb="ESPRESSO DOPPIO" price1="380" kitchen="0" vat1="2" ggroup="2" ttime="" contiment="false" recipe="false" active="true" correlation="" /><item codart="6" desca="ESPRESSO FREDDO" descb="ESPRESSO FREDDO" price1="400" kitchen="0" vat1="2" ggroup="2" ttime="" contiment="false" recipe="false" active="true" correlation="" /><item codart="7" desca="ESPRESSO CARAMEL CALDO" descb="ESPRESSO CARAMEL CALDO" price1="490" kitchen="0" vat1="2" ggroup="2" ttime="" contiment="false" recipe="false" active="true" correlation="" /><item codart="8" desca="ESPRESSO AMERICANO" descb="ESPRESSO AMERICANO" price1="310" kitchen="0" vat1="2" ggroup="2" ttime="" contiment="false" recipe="false" active="true" correlation="" /><item codart="9" desca="ESPRESSO CON PANNA FREDDO" descb="ESPRESSO CON PANNA FREDDO" price1="410" kitchen="0" vat1="2" ggroup="2" ttime="" contiment="false" recipe="false" active="true" correlation="" /></Items></EndOfDay></string>

Εικόνα 31: Είσοδος Έξοδος μεθόδου EndofDay

Τα μηνύματα SOAP ερώτησης και απόκρισης της κλήσης αυτής φαίνονται στο παρακάτω σχήμα:

```
POST /DominosInternetService/InternetDeliveryService.asmx HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://hit.com.gr/EndOfDay"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <EndOfDay xmlns="http://hit.com.gr/">
      <Day>dateTime</Day>
    </EndOfDay>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <EndOfDayResponse xmlns="http://hit.com.gr/">
      <EndOfDayResult>string</EndOfDayResult>
    </EndOfDayResponse>
  </soap:Body>
</soap:Envelope>
```

Εικόνα 32: SOAP EndOfDay

4.7.2. GetOrderStatus

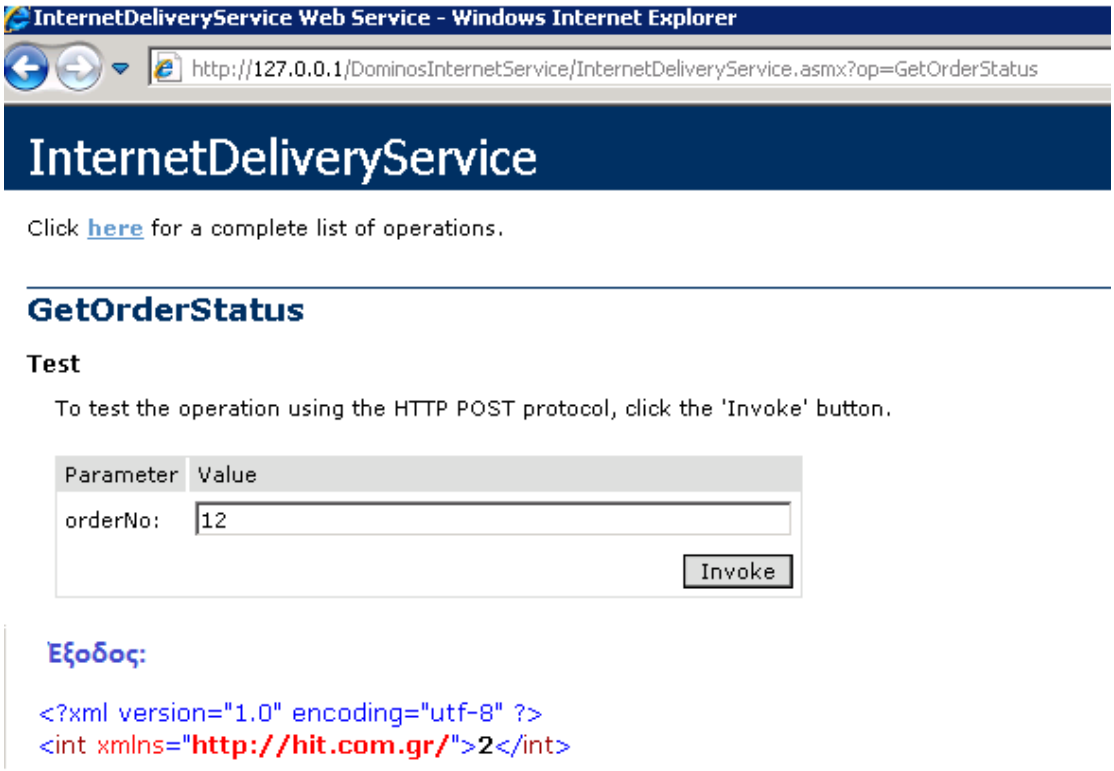
Πρόκειται για μια κλήση κατά την οποία δίνεται ο αριθμός της παραγγελίας και επιστρέφεται το στάδιο στο οποίο βρίσκεται. Συγκεκριμένα, όταν το στάδιο έχει την τιμή 1 η παραγγελία έχει εκτυπωθεί στην κουζίνα, όταν έχει 2 είναι έτοιμη άρα μπορεί να εκδοθεί απόδειξη, όταν έχει τιμή 3 τότε έχει εκδοθεί απόδειξη, όταν είναι 4 η παραγγελία έχει παραδοθεί και τέλος όταν είναι 6 τότε έχει ακυρωθεί η παραγγελία.

GetOrderStatus			
	Περιγραφή	Τύπος	Παράδειγμα
Είσοδος	Αριθμός Παραγγελίας	int	12
Έξοδος	Αριθμός που χαρακτηρίζει την κατάσταση της παραγγελίας	int	2

Πίνακας 3: Είσοδος/'Εξοδος GetOrderStatus

Ένα παράδειγμα κλήσης και απάντησης της κλήσης αυτής φαίνεται παρακάτω.

Είσοδος:



Έξοδος:

```
<?xml version="1.0" encoding="utf-8" ?>
<int xmlns="http://hit.com.gr/">2</int>
```

Εικόνα 33: Είσοδος/'Εξοδος GetOrderStatus

Τα μηνύματα SOAP ερώτησης και απόκρισης της διαδικασίας φαίνονται παρακάτω.

```
POST /DominosInternetService/InternetDeliveryService.asmx HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://hit.com.gr/GetOrderStatus"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetOrderStatus xmlns="http://hit.com.gr/">
      <orderNo>int</orderNo>
    </GetOrderStatus>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetOrderStatusResponse xmlns="http://hit.com.gr/">
      <GetOrderStatusResult>int</GetOrderStatusResult>
    </GetOrderStatusResponse>
  </soap:Body>
</soap:Envelope>
```

Εικόνα 34: SOAP GetorderStatus

4.7.3. ReqOutOfOrderItems

Όταν πραγματοποιείται αυτή η κλήση δίνεται σαν είσοδος η παράμετρος του κωδικού ενός καταστήματος και μας επιστρέφει πληροφορίες σχετικά με αυτό, δηλαδή αν υπάρχουν είδη που έχει έλλειψη το συγκεκριμένο κατάστημα καθώς και τους χρόνους που παραδίδει στο κατάστημα ή στην οικία το συγκεκριμένο κατάστημα. Οι χρόνοι αλλάζουν ανάλογα με τους φόρτους εργασίας του κάθε καταστήματος. Επίσης μας επιστρέφει την κατάσταση του καταστήματος, δηλαδή 1 αν είναι κλειστό και 2 αν είναι ανοιχτό.

ReqOutOfOrderItems			
	Περιγραφή	Τύπος	Παράδειγμα
Είσοδος	Αριθμός Καταστήματος	Int	12
Έξοδος	Στοιχεία για το ζητούμενο κατάστημα π.χ Ανοιχτό ή Κλειστό,	String	<ResponseOutOfStockItems> <Store ID="12" Status="1" /> <TimeToDeliver>0:17</TimeToDeliver> <TimeToCarryOut>0:14</TimeToCarryOut> <OutOfStock> <Item code="8" /> </OutOfStock></ResponseOutOfStockItems>

	Χρόνοι παραγγελιών

Πίνακας 4: Είσοδος/'Εξοδος ReqOutOfOrderItems

Ένα παράδειγμα για το κατάστημα με κωδικό 12 φαίνεται παρακάτω.

Εικόνα 35: Είσοδος/'Εξοδος ReqOutOfOrderItems

Τα μηνύματα SOAP ερώτησης και απόκρισης της κλήσης αυτής φαίνονται παρακάτω:

```
POST /DominosInternetService/InternetDeliveryService.asmx HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://hit.com.gr/ReqOutOfOrderItems"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ReqOutOfOrderItems xmlns="http://hit.com.gr/">
      <StoreId>int</StoreId>
    </ReqOutOfOrderItems>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ReqOutOfOrderItemsResponse xmlns="http://hit.com.gr/">
      <ReqOutOfOrderItemsResult>string</ReqOutOfOrderItemsResult>
    </ReqOutOfOrderItemsResponse>
  </soap:Body>
</soap:Envelope>
```

Εικόνα 36: SOAP ReqOutOfOrderItems

4.7.4. RequestNewOrder

Σε αυτή την κλήση δίνεται είσοδος μια συμβολοσειρά που περιέχει τα στοιχεία της παραγγελίας και του πελάτη και επιστρέφεται True αν η παραγγελία καταχωρηθεί στη βάση δεδομένων ή False αν αποτύχει. Σε περίπτωση αποτυχίας, δημιουργείται αρχείο καταγραφής σφάλματος με ονομασία OrdersError.Log στο app path του Web Service. Η καταχώρηση γίνεται στον πίνακα orders.

RequestNewOrder			
	Περιγραφή	Τύπος	Παράδειγμα
Είσοδος	Εισερχόμενη Παραγγελία	string	<RequestNewOrder> <Order No="4" Type="Type" bonus="3" Agent="WebSite" StoreId="10"><Customer id="1" phone="2101234567" LName="ΜΕΝΕΛΑΟΣ" FName="ΜΑΡΚΟΥ" Address="ΜΕΤΣΟΒΟΥ" AddressNo="23" city="ΚΟΡΥΔΑΛΛΟΣ" zipCode=""/><Remarks><Payment>CASH</Payment><Customers /><Dominos /><Orders /></Remarks><Item code="1" Description="ESPRESSO" Type="" Price="300" Group="1" Quantity="1" /><Item code="2" Description="ESPRESSO MACHIATTO" Type="" Price="310" Group="1" Quantity="2" /></Order></RequestNewOrder>
Έξοδος	Αποδοχή ή όχι της παραγγελιας	Boolean	true

Πίνακας 5: Είσοδος/'Εξοδος RequestNewOrder

Ένα παράδειγμα της δομής της συμβολοσειράς εισόδου φαίνεται στο παρακάτω σχήμα.

```
<RuqestNewOrder>
<Order No="4" Type="Type" bonus="3" Agent="WebSite" StoreId="10">
```

```
  <Customer id="1" phone="2101234567" LName="ΜΕΝΕΛΑΟΣ"
FName="ΜΑΡΚΟΥ" Address="ΜΕΤΣΟΒΟΥ" AddressNo="23" city="ΚΟΡΥΔΑΛΛΟΣ"
zipCode="" />
<Remarks>
  <Payment>CASH</Payment>
  <Customers />
  <Dominos />
  <Orders />
  </Remarks>
  <Item code="1" Description="ESPRESSO" Type="" Price="300" Group="1"
Quantity="1" />
  <Item code="2" Description="ESPRESSO MACHIATTO" Type="" Price="310"
Group="1" Quantity="2" />
  </Order>
  </RequestNewOrder>
```

Εικόνα 37: Έισοδος RequestNewOrder

Υποβάλλοντας αυτή τη συμβολοσειρά, αν δεν προκύψει σφάλμα καταγράφεται στον πίνακα Orders και παίρνουμε απάντηση όπως στο παρακάτω σχήμα.

```
<?xml version="1.0" encoding="utf-8" ?>
<boolean xmlns="http://hit.com.gr/">true</boolean>
```

Ειδάλλως, παίρνουμε την απάντηση που φαίνεται παρακάτω.

```
<?xml version="1.0" encoding="utf-8" ?>
<boolean xmlns="http://hit.com.gr/">false</boolean>
```

Τα μηνύματα ερώτησης και απόκρισης για αυτή την μέθοδο φαίνονται παρακάτω:

```
POST /DominosInternetService/InternetDeliveryService.asmx HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://hit.com.gr/RequestNewOrder"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
  <RequestNewOrder xmlns="http://hit.com.gr/">
   <RequestNewOrder>string</RequestNewOrder>
  </RequestNewOrder>
 </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <RequestNewOrderResponse xmlns="http://hit.com.gr/">
      <RequestNewOrderResult>boolean</RequestNewOrderResult>
    </RequestNewOrderResponse>
  </soap:Body>
</soap:Envelope>
```

Εικόνα 38: SOAP RequestNewOrder

4.7.5. RequestNewOrderFromFile

Η μέθοδος αυτή έχει δημιουργηθεί κυρίως για λόγους ελέγχου του webservice. Με τη κλήση δίνεται η δυνατότητα καταχώρησης μιας παραγγελίας από συγκεκριμένο αρχείο μέσα στο application Directory του webservice και αποκρίνεται με true ή false. Χρησιμοποιείται κυρίως όταν αποτυγχάνει η μέθοδος RequestNewOrder.

RequestNewOrderFromFile			
	Περιγραφή	Τύπος	Παράδειγμα
Είσοδος	Παραγγελία από συγκεκριμένο αρχείο	-	-
Έξοδος	Αποδοχή ή όχι της παραγγελιας	Boolean	true

Πίνακας 6: Είσοδος/Εξοδος RequestNewOrderFromFile

Παράδειγμά Απάντησης.

```
<?xml version="1.0" encoding="utf-8" ?>
<boolean xmlns="http://hit.com.gr/">true</boolean>
```

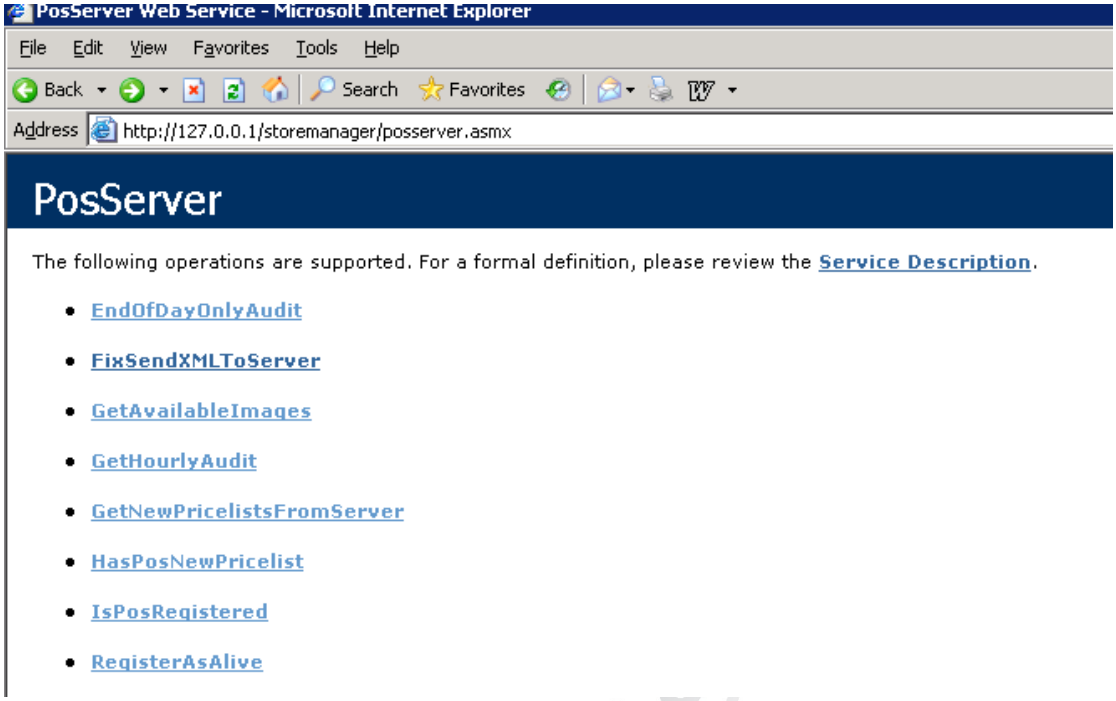
Το SOAP για αυτή τη μέθοδο φαίνεται παρακάτω:

```
SOAP 1.1
The following is a sample SOAP 1.1 request and response. The placeholders shown need to be
replaced with actual values.
POST /DominosInternetService/InternetDeliveryService.asmx HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://hit.com.gr/RequestNewOrderFromFile"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <RequestNewOrderFromFile xmlns="http://hit.com.gr/" />
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <RequestNewOrderFromFileResponse xmlns="http://hit.com.gr/">
      <RequestNewOrderFromFileResult>boolean</RequestNewOrderFromFileResult>
    </RequestNewOrderFromFileResponse>
  </soap:Body>
</soap:Envelope>
```

Εικόνα 39: SOAP RequestNewOrderFromFile

4.8 Μέθοδοι του Web Service Storemanager

Η λίστα των μεθόδων για το συγκεκριμένο WebService φαίνεται στο παρακάτω σχήμα. Η κλήσεις γίνονται από τρίτες εφαρμογές ή άλλα Web Services, με χρήση κάποιων παραμέτρων εισόδου που περιγράφονται αναλυτικά για κάθε μία.



Εικόνα 40: Λίστα Μεθόδων Storemanager

Το αρχείο Wsdl που το περιγράφει φαίνεται στο παρακάτω πίνακα

```
<?xml version="1.0" encoding="utf-8" ?>

-<wsdl:definitionsxmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://hit.com.gr/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" targetNamespace="http://hit.com.gr/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

- <wsdl:types>

- <s:schema elementFormDefault="qualified" targetNamespace="http://hit.com.gr/">

- <s:element name="SendOpenProductToStoreServer">

- <s:complexType>

- <s:sequence>

 <s:element minOccurs="0" maxOccurs="1" name="openProductXML" type="s:string" />

 </s:sequence>

 </s:complexType>

 </s:element>

- <s:element name="SendOpenProductToStoreServerResponse">

- <s:complexType>

- <s:sequence>
```

```
 <s:element minOccurs="1" maxOccurs="1" name="SendOpenProductToStoreServerResult"
type="s:boolean" />
 </s:sequence>
 </s:complexType>
 </s:element>
- <s:element name="EndOfDayOnlyAudit">
- <s:complexType>
- <s:sequence>
 <s:element minOccurs="1" maxOccurs="1" name="foDay" type="s:dateTime" />
 <s:element minOccurs="1" maxOccurs="1" name="closeId" type="s:int" />
 <s:element minOccurs="1" maxOccurs="1" name="totalLabDay" type="s:int" />
 <s:element minOccurs="1" maxOccurs="1" name="ItemsCount" type="s:int" />
 <s:element minOccurs="1" maxOccurs="1" name="totalLabFB" type="s:int" />
 <s:element minOccurs="1" maxOccurs="1" name="gross" type="s:double" />
 <s:element minOccurs="1" maxOccurs="1" name="net" type="s:double" />
 <s:element minOccurs="1" maxOccurs="1" name="ticketsCount" type="s:int" />
 <s:element minOccurs="1" maxOccurs="1" name="ticketsAverage" type="s:double" />
 <s:element minOccurs="0" maxOccurs="1" name="StoreId" type="s:string" />
 <s:element minOccurs="0" maxOccurs="1" name="PosId" type="s:string" />
 </s:sequence>
 </s:complexType>
 </s:element>
- <s:element name="EndOfDayOnlyAuditResponse">
- <s:complexType>
- <s:sequence>
 <s:element minOccurs="1" maxOccurs="1" name="EndOfDayOnlyAuditResult" type="s:boolean" />
 </s:sequence>
 </s:complexType>
 </s:element>
- <s:element name="SendXmlToStoreServer">
- <s:complexType>
- <s:sequence>
 <s:element minOccurs="1" maxOccurs="1" name="foDay" type="s:dateTime" />
```

```
<s:element minOccurs="1" maxOccurs="1" name="closeId" type="s:int" />
<s:element minOccurs="0" maxOccurs="1" name="posId" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="xmlFile" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="xmlFileName" type="s:string" />
<s:element minOccurs="1" maxOccurs="1" name="isServer" type="s:boolean" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="SendXmlToStoreServerResponse">
- <s:complexType>
- <s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="SendXmlToStoreServerResult" type="s:boolean" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="FixSendXMLToServer">
- <s:complexType>
- <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="EndOfDayDetailsId" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="XMLFileName" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="FixSendXMLToServerResponse">
- <s:complexType>
- <s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="FixSendXMLToServerResult" type="s:boolean" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="RegisterAsAlive">
- <s:complexType>
- <s:sequence>
```

```
 <s:element minOccurs="0" maxOccurs="1" name="posNo" type="s:string" />
 </s:sequence>
 </s:complexType>
 </s:element>
- <s:element name="RegisterAsAliveResponse">
- <s:complexType>
- <s:sequence>
 <s:element minOccurs="1" maxOccurs="1" name="RegisterAsAliveResult" type="s:int" />
 </s:sequence>
 </s:complexType>
 </s:element>
- <s:element name="GetNewPricelistsFromServer">
- <s:complexType>
- <s:sequence>
 <s:element minOccurs="1" maxOccurs="1" name="scheduleDate" type="s:dateTime" />
 <s:element minOccurs="1" maxOccurs="1" name="changeOrder" type="s:int" />
 <s:element minOccurs="0" maxOccurs="1" name="itemsXml" type="s:string" />
 <s:element minOccurs="0" maxOccurs="1" name="items1Xml" type="s:string" />
 <s:element minOccurs="0" maxOccurs="1" name="pageItemsXml" type="s:string" />
 </s:sequence>
 </s:complexType>
 </s:element>
- <s:element name="GetNewPricelistsFromServerResponse">
- <s:complexType>
- <s:sequence>
 <s:element minOccurs="1" maxOccurs="1" name="GetNewPricelistsFromServerResult" type="s:boolean"
/>
</s:sequence>
 </s:complexType>
 </s:element>
- <s:element name="HasPosNewPricelist">
- <s:complexType>
- <s:sequence>
```

```
<s:element minOccurs="0" maxOccurs="1" name="posid" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="HasPosNewPricelistResponse">
- <s:complexType>
- <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="HasPosNewPricelistResult" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="IsPosRegistered">
- <s:complexType>
- <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="storeId" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="posId" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="IsPosRegisteredResponse">
- <s:complexType>
- <s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="IsPosRegisteredResult" type="s:boolean" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="GetHourlyAudit">
- <s:complexType>
- <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="storeId" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="posId" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="hourlySalesXml" type="s:string" />
</s:sequence>
```

```
  </s:complexType>
  </s:element>
- <s:element name="GetHourlyAuditResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="GetHourlyAuditResult" type="s:boolean" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="GetAvailableImages">
  <s:complexType />
  </s:element>
- <s:element name="GetAvailableImagesResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="GetAvailableImagesResult" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:element>
  </s:schema>
  </wsdl:types>
- <wsdl:message name="EndOfDayOnlyAuditSoapOut">
  <wsdl:part name="parameters" element="tns:EndOfDayOnlyAuditResponse" />
  </wsdl:message>
-<wsdl:message name="RegisterAsAliveSoapIn">
  <wsdl:part name="parameters" element="tns:RegisterAsAlive" />
  </wsdl:message>
- <wsdl:message name="RegisterAsAliveSoapOut">
  <wsdl:part name="parameters" element="tns:RegisterAsAliveResponse" />
  </wsdl:message>
- <wsdl:message name="GetNewPricelistsFromServerSoapIn">
  <wsdl:part name="parameters" element="tns:GetNewPricelistsFromServer" />
  </wsdl:message>
```

```
- <wsdl:message name="GetNewPricelistsFromServerSoapOut">
  <wsdl:part name="parameters" element="tns:GetNewPricelistsFromServerResponse" />
  </wsdl:message>
- <wsdl:message name="HasPosNewPricelistSoapIn">
  <wsdl:part name="parameters" element="tns:HasPosNewPricelist" />
  </wsdl:message>
- <wsdl:message name="HasPosNewPricelistSoapOut">
  <wsdl:part name="parameters" element="tns:HasPosNewPricelistResponse" />
  </wsdl:message>
- <wsdl:message name="IsPosRegisteredSoapIn">
  <wsdl:part name="parameters" element="tns:IsPosRegistered" />
 </wsdl:message>
- <wsdl:message name="IsPosRegisteredSoapOut">
  <wsdl:part name="parameters" element="tns:IsPosRegisteredResponse" />
  </wsdl:message>
- <wsdl:message name="GetHourlyAuditSoapIn">
  <wsdl:part name="parameters" element="tns:GetHourlyAudit" />
  </wsdl:message>
- <wsdl:message name="GetHourlyAuditSoapOut">
  <wsdl:part name="parameters" element="tns:GetHourlyAuditResponse" />
  </wsdl:message>
- <wsdl:message name="GetAvailableImagesSoapIn">
  <wsdl:part name="parameters" element="tns:GetAvailableImages" />
  </wsdl:message>
- <wsdl:message name="GetAvailableImagesSoapOut">
  <wsdl:part name="parameters" element="tns:GetAvailableImagesResponse" />
  </wsdl:message>
- <wsdl:portType name="PosServerSoap">
- <wsdl:operation name="SendOpenProductToStoreServer">
  <wsdl:input message="tns:SendOpenProductToStoreServerSoapIn" />
  <wsdl:output message="tns:SendOpenProductToStoreServerSoapOut" />
  </wsdl:operation>
- <wsdl:operation name="EndOfDayOnlyAudit">
```

```
<wsdl:input message="tns:EndOfDayOnlyAuditSoapIn" />
<wsdl:output message="tns:EndOfDayOnlyAuditSoapOut" />
</wsdl:operation>
- <wsdl:operation name="SendXmlToStoreServer">
<wsdl:input message="tns:SendXmlToStoreServerSoapIn" />
<wsdl:output message="tns:SendXmlToStoreServerSoapOut" />
</wsdl:operation>
- <wsdl:operation name="FixSendXMLToServer">
<wsdl:input message="tns:FixSendXMLToServerSoapIn" />
<wsdl:output message="tns:FixSendXMLToServerSoapOut" />
</wsdl:operation>
- <wsdl:operation name="RegisterAsAlive">
<wsdl:input message="tns:RegisterAsAliveSoapIn" />
<wsdl:output message="tns:RegisterAsAliveSoapOut" />
</wsdl:operation>
- <wsdl:operation name="GetNewPricelistsFromServer">
<wsdl:input message="tns:GetNewPricelistsFromServerSoapIn" />
<wsdl:output message="tns:GetNewPricelistsFromServerSoapOut" />
</wsdl:operation>
- <wsdl:operation name="HasPosNewPricelist">
<wsdl:input message="tns:HasPosNewPricelistSoapIn" />
<wsdl:output message="tns:HasPosNewPricelistSoapOut" />
</wsdl:operation>
- <wsdl:operation name="IsPosRegistered">
<wsdl:input message="tns:IsPosRegisteredSoapIn" />
<wsdl:output message="tns:IsPosRegisteredSoapOut" />
</wsdl:operation>
- <wsdl:operation name="GetHourlyAudit">
<wsdl:input message="tns:GetHourlyAuditSoapIn" />
<wsdl:output message="tns:GetHourlyAuditSoapOut" />
</wsdl:operation>
- <wsdl:operation name="GetAvailableImages">
<wsdl:input message="tns:GetAvailableImagesSoapIn" />
```

```
<wsdl:output message="tns:GetAvailableImagesSoapOut" />
</wsdl:operation>
</wsdl:portType>
- <wsdl:binding name="PosServerSoap" type="tns:PosServerSoap">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="SendOpenProductToStoreServer">
<soap:operation soapAction="http://hit.com.gr/SendOpenProductToStoreServer" style="document" />
- <wsdl:input>
<soap:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap:body use="literal" />
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="EndOfDayOnlyAudit">
<soap:operation soapAction="http://hit.com.gr/EndOfDayOnlyAudit" style="document" />
- <wsdl:input>
<soap:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap:body use="literal" />
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="SendXmlToStoreServer">
<soap:operation soapAction="http://hit.com.gr/SendXmlToStoreServer" style="document" />
- <wsdl:input>
<soap:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap:body use="literal" />
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="FixSendXMLToServer">
```

```
<soap:operation soapAction="http://hit.com.gr/FixSendXMLToServer" style="document" />
- <wsdl:input>
<soap:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap:body use="literal" />
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="RegisterAsAlive">
<soap:operation soapAction="http://hit.com.gr/RegisterAsAlive" style="document" />
- <wsdl:input>
<soap:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap:body use="literal" />
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="GetNewPricelistsFromServer">
<soap:operation soapAction="http://hit.com.gr/GetNewPricelistsFromServer" style="document" />
- <wsdl:input>
<soap:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap:body use="literal" />
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="HasPosNewPricelist">
<soap:operation soapAction="http://hit.com.gr/HasPosNewPricelist" style="document" />
- <wsdl:input>
<soap:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap:body use="literal" />
```

```
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="IsPosRegistered">
  <soap:operation soapAction="http://hit.com.gr/IsPosRegistered" style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="GetHourlyAudit">
  <soap:operation soapAction="http://hit.com.gr/GetHourlyAudit" style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="GetAvailableImages">
  <soap:operation soapAction="http://hit.com.gr/GetAvailableImages" style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="PosServerSoap12" type="tns:PosServerSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="SendOpenProductToStoreServer">
```

```
<soap12:operation soapAction="http://hit.com.gr/SendOpenProductToStoreServer" style="document" />
- <wsdl:input>
<soap12:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap12:body use="literal" />
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="EndOfDayOnlyAudit">
<soap12:operation soapAction="http://hit.com.gr/EndOfDayOnlyAudit" style="document" />
- <wsdl:input>
<soap12:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap12:body use="literal" />
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="SendXmlToStoreServer">
<soap12:operation soapAction="http://hit.com.gr/SendXmlToStoreServer" style="document" />
- <wsdl:input>
<soap12:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap12:body use="literal" />
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="FixSendXMLToServer">
<soap12:operation soapAction="http://hit.com.gr/FixSendXMLToServer" style="document" />
- <wsdl:input>
<soap12:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap12:body use="literal" />
```

```
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="RegisterAsAlive">
<soap12:operation soapAction="http://hit.com.gr/RegisterAsAlive" style="document" />
- <wsdl:input>
<soap12:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap12:body use="literal" />
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="GetNewPricelistsFromServer">
<soap12:operation soapAction="http://hit.com.gr/GetNewPricelistsFromServer" style="document" />
- <wsdl:input>
<soap12:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap12:body use="literal" />
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="HasPosNewPricelist">
<soap12:operation soapAction="http://hit.com.gr/HasPosNewPricelist" style="document" />
- <wsdl:input>
<soap12:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap12:body use="literal" />
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="IsPosRegistered">
<soap12:operation soapAction="http://hit.com.gr/IsPosRegistered" style="document" />
- <wsdl:input>
<soap12:body use="literal" />
```

```
</wsdl:input>
- <wsdl:output>
<soap12:body use="literal" />
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="GetHourlyAudit">
<soap12:operation soapAction="http://hit.com.gr/GetHourlyAudit" style="document" />
- <wsdl:input>
<soap12:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap12:body use="literal" />
</wsdl:output>
</wsdl:operation>
- <wsdl:operation name="GetAvailableImages">
<soap12:operation soapAction="http://hit.com.gr/GetAvailableImages" style="document" />
- <wsdl:input>
<soap12:body use="literal" />
</wsdl:input>
- <wsdl:output>
<soap12:body use="literal" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
- <wsdl:service name="PosServer">
- <wsdl:port name="PosServerSoap" binding="tns:PosServerSoap">
<soap:address location="http://127.0.0.1/storemanager/posserver.asmx" />
</wsdl:port>
- <wsdl:port name="PosServerSoap12" binding="tns:PosServerSoap12">
<soap12:address location="http://127.0.0.1/storemanager/posserver.asmx" />
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```


Πίνακας 7: WSDL Web Service

4.8.1.EndofDayonlyAudit

Η κλήση της μεθόδου EndofDayOnlyAdit γίνεται από το πρόγραμμα που υπάρχει στα υποκαταστήματα της εταιρείας κατά το κλείσιμο της Ημέρας. Ως είσοδο δίνονται μια σειρά δεδομένων από τη ΒΔ του υποκαταστήματος με σκοπό να καταγρούν στην κεντρική ΒΔ της εταιρείας στον πίνακα EndofDay. Σε περιπτωση επιτυχίας εισαγωγής των δεδομένων στη βάση η μέθοδος απαντάει με true ενώ σε αντίθετη περίπτωση με false.

EndofDayonlyAudit			
	Περιγραφή	Τύπος	Παράδειγμα
Είσοδος	Ημερομηνία (FoDay)	DateTime	'2012-6-15'
	Αυξων Αριθμός Κλεισίματος (CloseId)	Int	15
	Εσοδά από Πωλήσεις (TotalLabday)	Int	10000
	Συνολο Ειδών (ItemCount)	Int	10
	Έσοδα Πωλήσεων Ειδών (TotalLabfb)	Int	10000
	Μικτές Πωλήσεις (Gross)	Double	100,00
	Καθαρές Πωλήσεις	Double	77,00

	(Net)		
	Αριθμός Αποδείξεων (TicketCount)	Int	4
	Μέσος όρος Αποδείξεων (TicketAverage)	Double	25,00
	Κωδικός Καταστηματος (StoreID)	String	Store1
	Κωδικός Ταμειακής Μηχανής (PosID)	String	Pos1
Έξοδος	Καταχώρηση ή όχι των δεδομένων στη βάση	Boolean	true

Πίνακας 8: Είσοδος/Εξοδος EndOfDayOnlyAudit

Παρακάτω φαίνεται ένα παράδειγμα κλήσης της μεθόδου

Είσοδος

PosServer

Click here for a complete list of operations.

EndOfDayOnlyAudit

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
foDay:	2012-6-15
closeId:	15
totalLabDay:	10000
ItemsCount:	10
totalLabFB:	10000
gross:	100
net:	77
ticketsCount:	4
ticketsAverage:	25
StoreId:	STORE1
PosId:	POS1

Invoke

Εξοδος

```
<?xml version="1.0" encoding="utf-8" ?>
<boolean xmlns="http://hit.com.gr/">true</boolean>
```

Εικόνα 41: Είσοδος/Εξοδος EndOfDayOnlyAudit

Το SOAP μήνυμα κλήσης και απάντησης της μεθόδου

```
POST /storemanager/posserver.asmx HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://hit.com.gr/EndOfDayOnlyAudit"
```

```
<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

 <soap:Body>

  <EndOfDayOnlyAudit xmlns="http://hit.com.gr/">

   <foDay>dateTime</foDay>

   <closeId>int</closeId>

   <totalLabDay>int</totalLabDay>

   <ItemsCount>int</ItemsCount>

   <totalLabFB>int</totalLabFB>

   <gross>double</gross>

   <net>double</net>

   <ticketsCount>int</ticketsCount>

   <ticketsAverage>double</ticketsAverage>

   <StoreId>string</StoreId>

   <PosId>string</PosId>

  </EndOfDayOnlyAudit>

 </soap:Body>

</soap:Envelope>

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: length


<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

 <soap:Body>

  <EndOfDayOnlyAuditResponse xmlns="http://hit.com.gr/">

   <EndOfDayOnlyAuditResult>boolean</EndOfDayOnlyAuditResult>

  </EndOfDayOnlyAuditResponse>
```

```
  </soap:Body>
</soap:Envelope>
```

Εικόνα 42: SOAP EndOfDayOnlyAudit

4.8.2.FixSendXmlToServer

Η μέθοδος αυτή καλείται σε περίπτωση που η προηγούμενη μέθοδος (EndOfDayOnlyAudit) απαντήσει με false.Πιο συγκεκριμένα, η μέθοδος FixSendXmlToServer δίνει τη δυνατότητα δημιουργίας από τη πλευρά του Client ένα Xml με όλα τα δεδομένα, το οποίο θα γίνει προσπάθεια να καταγραφεί στην κεντρική ΒΔ της εταιρίας. Αν η εισαγωγή είναι επιτυχής τότε απαντάει με true ειδάλλως με false.

FixSendXmlToServer			
	Περιγραφή	Τύπος	Παράδειγμα
Είσοδος	Κωδικός Κλεισίματος (EndOfDayDetailsID)	String	'Eod2012-6-15'
	Δεδομένα Κλεισίματος (XmlFileName)	String	<Id="6AC41B9A-FC61-4C42-A547-0167219F72A9" FODay="2010-03-14T00:00:00" CloseId="173" Gross="178290" Net="1620.4500" TicketsCount="2…
Έξοδος	Καταχώρηση ή όχι των δεδομένων στη βάση	Boolean	true

Πίνακας 9: Είσοδος/Έξοδος FixSendXmlToServer

Παρακάτω φαίνεται ένα παράδειγμα κλήσης της μεθόδου

Είσοδος

FixSendXMLToServer

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
EndOfDayDetailsId:	eod2012-6-15
XMLFileName:	3" Gross="178290" Net="1620.4500" TicketsCount="2

Invoke

Έξοδος

```
<?xml version="1.0" encoding="utf-8" ?>
<boolean xmlns="http://hit.com.gr/">true</boolean>
```

Εικόνα 43: Είσοδος/Εξοδος FixSendXmlToServer

Το SOAP μήνυμα κλήσης και απάντησης της μεθόδου

```
POST /storemanager/posserver.asmx HTTP/1.1

Host: 127.0.0.1

Content-Type: text/xml; charset=utf-8

Content-Length: length

SOAPAction: "http://hit.com.gr/FixSendXMLToServer"



<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

 <soap:Body>

  <FixSendXMLToServer xmlns="http://hit.com.gr/">

   <EndOfDayDetailsId>string</EndOfDayDetailsId>

   <XMLFileName>string</XMLFileName>

  </FixSendXMLToServer>
```

```
 </soap:Body>

</soap:Envelope>

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: length


<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

 <soap:Body>

  <FixSendXMLToServerResponse xmlns="http://hit.com.gr/">

                <FixSendXMLToServerResult>boolean</FixSendXMLToServerResult>

  </FixSendXMLToServerResponse>

 </soap:Body>

</soap:Envelope>
```

Πίνακας 10: SOAP FixSendXmlToServer

4.8.3.GetAvailableImages

Με τη κλήση αυτής της μεθόδου τα καταστήματα κατεβάζουν νέες φωτογραφίες των προιόντων που έχουν για χρήση στο κατάλογο τους.

GetAvailableImages			
	Περιγραφή	Τύπος	Παράδειγμα
Έξοδος	Xml που περιέχει εικόνες	String	?xml version="1.0" encoding="utf-8" ?> string xmlns="http://hit.com.gr/"> http://127.0.0.1/storemanager//BMPSA/0.jpg|http://127.0.0.1/storemanager//BMPSA/1.jpg

Πίνακας 11: Είσοδος/Εξοδος GetAvailableImages

Παρακάτω φαίνεται ένα παράδειγμα απάντησης της μεθόδου


```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://hit.com.gr/">http://127.0.0.1/storemanager//BMPSA/0.jpg|http://127.0.0.1/storemanager//BMPSA/1.jpg|http://127.0.0.1/storemanager//BMPSA/10.jpg|http://127.0.0.1/storemanager//BMPSA/11.jpg|http://127.0.0.1/storemanager//BMPSA/12.jpg|http://127.0.0.1/storemanager//BMPSA/13.jpg|http://127.0.0.1/storemanager//BMPSA/14.jpg|http://127.0.0.1/storemanager//BMPSA/15.jpg|http://127.0.0.1/storemanager//BMPSA/16.jpg|http://127.0.0.1/storemanager//BMPSA/17.jpg|http://127.0.0.1/storemanager//BMPSA/172.JPG|http://</string>
```


Εικόνα 44: Παράδειγμα Εξόδου GetAvailableImages

Το SOAP μήνυμα κλήσης και απάντησης της μεθόδου

```
POST /storemanager/posserver.asmx HTTP/1.1

Host: 127.0.0.1

Content-Type: text/xml; charset=utf-8

Content-Length: length

SOAPAction: "http://hit.com.gr/GetAvailableImages"

<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

  <soap:Body>

    <GetAvailableImages xmlns="http://hit.com.gr/" />

  </soap:Body>

</soap:Envelope>


HTTP/1.1 200 OK
```

```
Content-Type: text/xml; charset=utf-8

Content-Length: length



<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

 <soap:Body>

  <GetAvailableImagesResponse xmlns="http://hit.com.gr/">

   <GetAvailableImagesResult>string</GetAvailableImagesResult>

  </GetAvailableImagesResponse>

 </soap:Body>

</soap:Envelope>
```

Εικόνα 45: SOAP GetAvailableImages

4.8.4.GetHourlyAudit

Με τη κλήση αυτής της μεθόδου τα καταστήματα καταχωρούν στο κεντρικό server τις πωλήσεις τους ανα ώρα κατά τη διάρκεια της ημέρας. Η καταγραφή γίνεται στον πίνακα HourAuditDetail στην κεντρική ΒΔ της εταιριας. Σε περίπτωση επιτυχούς καταχώρησης επιστρέφεται True ειδάλλως False.

GetHourlyAudit			
	Περιγραφή	Τύπος	Παράδειγμα
Είσοδος	Κωδικός Καταστήματος (StoreId)	String	Store1
	Κωδικός Ταμειακής Μηχανής (PosId)	String	POS1

	Δεδομένα Ανα Ώρα (HourlySalesXml)	String	<hitpos.posuser.HourAuditDetail Id="62788FD9-2BD3-4F2B-81E3-4C75774EA2ED" HourAuditId="37FCB28B-B44E-4C54-BF43-E6DCD39A8FA5" Gross="22.0000" Net="17.8862" ItemsCount="4" TicketAverage="5.00" Hour="12" /> …
Έξοδος	Καταχώρηση ή όχι των δεδομένων στη βάση	Boolean	true

Πίνακας 12: Είσοδος/'Εξοδος GetHourlyAdit

Παρακάτω φαίνεται ένα παράδειγμα κλήσης της μεθόδου

Εισοδος

PosServer

Click here for a complete list of operations.

GetHourlyAudit

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
storeId:	Store1
posId:	pos1
hourlySalesXml:	ItemsCount="4" TicketAverage="5.00" Hour="12" />

Invoke

Εξοδος

```
<?xml version="1.0" encoding="utf-8" ?>
<boolean xmlns="http://hit.com.gr/">true</boolean>
```

Εικόνα 46:Είσοδος/'Εξοδος GetHourlyAdit

Το SOAP μήνυμα κλήσης και απάντησης της μεθόδου

```
POST /storemanager/posserver.asmx HTTP/1.1

Host: 127.0.0.1

Content-Type: text/xml; charset=utf-8

Content-Length: length

SOAPAction: "http://hit.com.gr/GetHourlyAudit"



<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

 <soap:Body>

  <GetHourlyAudit xmlns="http://hit.com.gr/">

   <storeId>string</storeId>

   <posId>string</posId>

   <hourlySalesXml>string</hourlySalesXml>

  </GetHourlyAudit>

 </soap:Body>

</soap:Envelope>

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: length



<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

 <soap:Body>

  <GetHourlyAuditResponse xmlns="http://hit.com.gr/">

   <GetHourlyAuditResult>boolean</GetHourlyAuditResult>
```

```
  </GetHourlyAuditResponse>
 </soap:Body>
</soap:Envelope>
```

Εικόνα 47:SOAP GetHourlyAudit

4.8.5. HasPosNewPriceList

Η αλυσίδα καταστημάτων ανα τακτά χρονικά διαστήματα αλλάζει τις τιμές των ειδών για να ανταπεξέλθει στον ανταγωνισμό. Οι τιμές αλλάζουν κεντρικά. Με τη κλήση αυτής της μεθόδου τα καταστήματα έχουν πρόσβαση στην κεντρική ΒΔ και ενημερώνονται για το εάν έχουν τιμές προς ενημέρωση και για το πότε θα γίνουν. Κάθε σημείο πώλησης του καταστήματος όταν ανοίγει κάθε μέρα κάνει ένα request δίνοντας το κωδικό του και λαμβάνει ως απάντηση ένα xml με το εάν έχει ή όχι αλλαγές ειδών-τιμών καθώς και την ώρα που αυτές πρέπει να γίνουν.Τα δεδομένα αυτά θα τα χρησιμοποιήσει ο client αργοτερα για τη κλήση της μεθοδου GetNewPriceListFromServer.

HasPosNewPriceList			
	Περιγραφή	Τύπος	Παράδειγμα
Είσοδος	Κωδικός Ταμειακής Μηχανής (PosId)	String	POS1
Έξοδος	Υπάρχουν Είδη για ενημέρωση και εάν ναι πότε?	String	`Id="0F09AB3A-2D96-4393-8A3A-2029C52179A4" UpdateDate="2011-08-29T18:28:34" Status="1" ChangeOrder="1" />`

Πίνακας 13: Έισοδος/Εξοδος HasPosNewPriceList

Παρακάτω φαίνεται ένα παράδειγμα κλήσης της μεθόδου

Είσοδος

HasPosNewPricelist

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
posid:	bd77f14f-e51c-4265-b291-f580360b0837

Invoke

Εξοδος

```
Id="0F09AB3A-2D96-4393-8A3A-2029C52179A4"          UpdateDate="2011-08-
29T18:28:34" Status="1" ChangeOrder="1" />
```

Εικόνα 48: Έισοδος/Έξοδος GetNewPriceListFromServer

Το SOAP μήνυμα κλήσης και απάντησης της μεθόδου

```
POST /storemanager/posserver.asmx HTTP/1.1

Host: 127.0.0.1

Content-Type: text/xml; charset=utf-8

Content-Length: length

SOAPAction: "http://hit.com.gr/HasPosNewPricelist"



<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

  <soap:Body>

    <HasPosNewPricelist xmlns="http://hit.com.gr/">

      <posid>string</posid>

    </HasPosNewPricelist>

  </soap:Body>

</soap:Envelope>
```

```
HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: length



<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

 <soap:Body>

  <HasPosNewPricelistResponse xmlns="http://hit.com.gr/">

   <HasPosNewPricelistResult>string</HasPosNewPricelistResult>

  </HasPosNewPricelistResponse>

 </soap:Body>

</soap:Envelope>
```

4.8.6.GetNewPriceListsFromServer

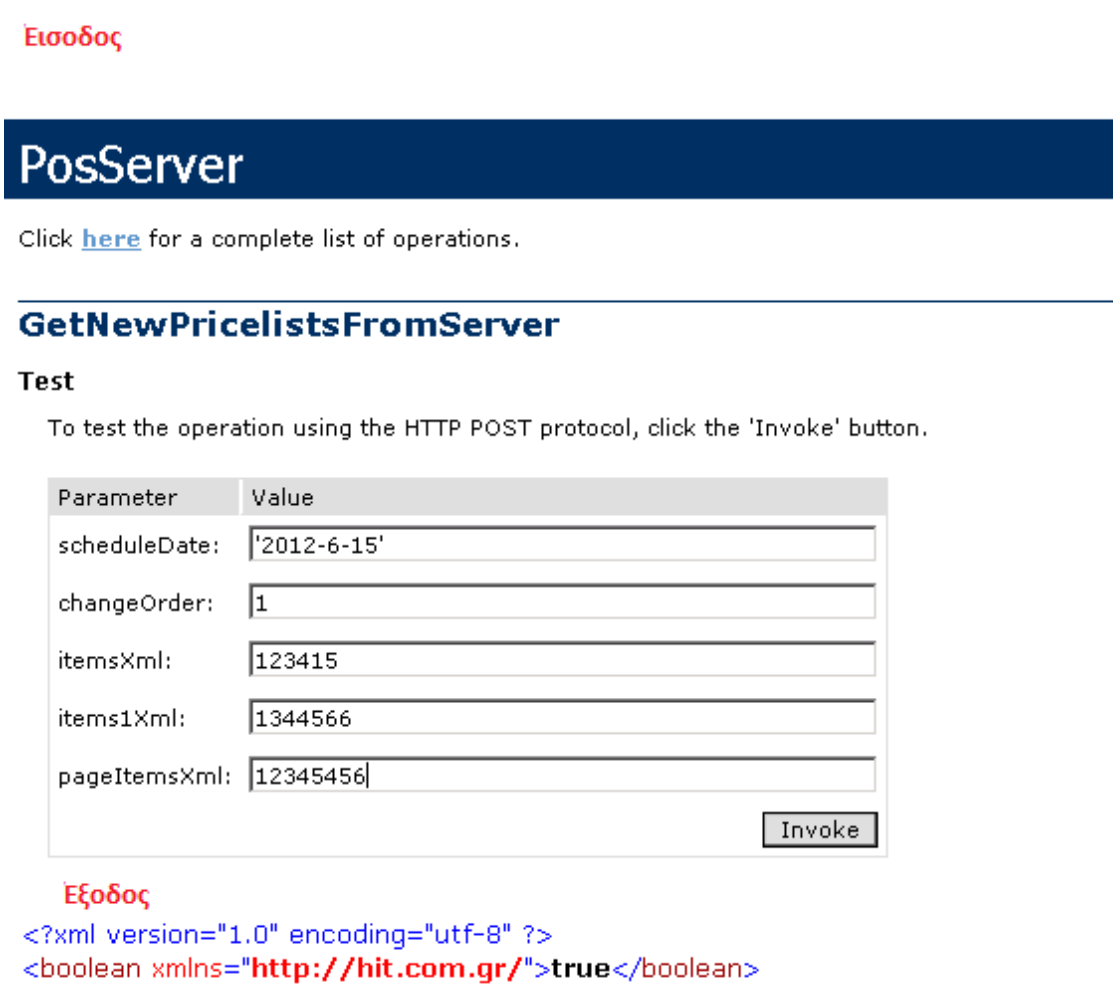
Ο client καλεί τη μέθοδο GetNewPriceListsFromServer δίνοντας ως ορίσματα τις απαντήσεις που έχει πάρει από την HasPosNewPricelists, ώστε να ενημερωθεί με τις νέες τιμές και τα νέα είδη. Εάν η ενημέρωση ολοκληρωθεί η μέθοδος απαντάει με true αλλιώς απαντάει false.

GetNewPriceListFromServer			
	Περιγραφή	Τύπος	Παράδειγμα
Είσοδος	Ημερομηνία που πρέπει να γίνει η Αλλαγή (ScheduleDate)	String	POS1
	Αυξων Αριθμός Αλλαγής(ChangeOrder)	Int	1
	Ταυτότητα πίνακα Ειδών (ItemsXml)	String	123456-567

	Ταυτότητα Πίνακα Τιμών (Items1Xml)	String	123456-567
	Ταυτότητα Πίνακα Οθονών (PageItemsXml)	String	123456-567
Έξοδος	Ολοκληρώθηκε επιτυχώς η ενημέρωση	boolean	True

Πίνακας 14: Είσοδος/Εξοδος GetNewPriceListaFromServer

Παρακάτω φαίνεται ένα παράδειγμα κλήσης της μεθόδου



Εικόνα 49:Είσοδος/Εξοδος GetNewPriceListaFromServer

Το SOAP μήνυμα κλήσης και απάντησης της μεθόδου

```
POST /storemanager/posserver.asmx HTTP/1.1

Host: 127.0.0.1

Content-Type: text/xml; charset=utf-8

Content-Length: length

SOAPAction: "http://hit.com.gr/GetNewPricelistsFromServer"


<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

 <soap:Body>

  <GetNewPricelistsFromServer xmlns="http://hit.com.gr/">

   <scheduleDate>dateTime</scheduleDate>

   <changeOrder>int</changeOrder>

   <itemsXml>string</itemsXml>

   <items1Xml>string</items1Xml>

   <pageItemsXml>string</pageItemsXml>

  </GetNewPricelistsFromServer>

 </soap:Body>

</soap:Envelope>

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: length


<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

 <soap:Body>

  <GetNewPricelistsFromServerResponse xmlns="http://hit.com.gr/">

   <GetNewPricelistsFromServerResult>boolean</GetNewPricelistsFromServerResult>

  </GetNewPricelistsFromServerResponse>

 </soap:Body>
```

```
</soap:Envelope>
```

Εικόνα 50: SOAP GetNewPriceListaFromServer

4.8.7 . IsPosRegistered

Τη πρώτη φορά που γίνεται εγκατάσταση κάποιου σημείο πώλησης σε κάποιο κατάστημα πρέπει να γίνει Register στο κεντρικό Server ώστε να ανοίξει δίαυλος επικοινωνίας. Το registration γίνεται με τη μέθοδο IsPosRegistered.

IsPosRegistered			
	Περιγραφή	Τύπος	Παράδειγμα
Είσοδος	Κωδικός Καταστήματος (StoreId)	String	Store1
	Κωδικός Ταμειακής Μηχανής (PosID)	String	Pos1
Έξοδος	Ολοκληρώθηκε επιτυχώς η ενημέρωση	Boolean	True

Πίνακας 15: Είσοδος/Εξοδος IsPosRegistered

Παρακάτω φαίνεται ένα παράδειγμα κλήσης της μεθόδου

Εισοδος

PosServer

Click here for a complete list of operations.

IsPosRegistered

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
storeId:	Store1
posId:	Pos1

Invoke

Εξοδος

```
<?xml version="1.0" encoding="utf-8" ?>
<boolean xmlns="http://hit.com.gr/">true</boolean>
```

Εικόνα 51: Είσοδος /Έξοδος IsPosRegistered

Το SOAP μήνυμα κλήσης και απάντησης της μεθόδου

```
POST /storemanager/posserver.asmx HTTP/1.1

Host: 127.0.0.1

Content-Type: text/xml; charset=utf-8

Content-Length: length

SOAPAction: "http://hit.com.gr/IsPosRegistered"



<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

 <soap:Body>

   <IsPosRegistered xmlns="http://hit.com.gr/">

     <storeId>string</storeId>
```

```
    <posId>string</posId>
   </IsPosRegistered>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length


<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
   <IsPosRegisteredResponse xmlns="http://hit.com.gr/">
    <IsPosRegisteredResult>boolean</IsPosRegisteredResult>
   </IsPosRegisteredResponse>
  </soap:Body>
</soap:Envelope>
```

Εικόνα 52: SOAP IsPosRegistered

4.8.8. RegisterAsAlive

Κάθε μέρα που ένα κατάστημα ανοίγει στέλνει ένα μήνυμα στο κεντρικό Server ότι έχει ανοίξει. Η κλήση της μεθόδου αυτής δηλώνει τη λειτουργία του καταστήματος.

RegisterAsAlive			
	Περιγραφή	Τύπος	Παράδειγμα
Είσοδος	Κωδικός Ταμειακής Μηχανής (PosId)	String	POS1
Έξοδος	Ολοκληρώθηκε το Register	Boolean	True

Πίνακας 16:Είσοδος/'Εξοδος RegisterAsAlive

Εισοδος

PosServer

Click here for a complete list of operations.

RegisterAsAlive

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
posNo:	

Invoke

Εξοδος

```
<?xml version="1.0" encoding="utf-8" ?>
<boolean xmlns="http://hit.com.gr/">true</boolean>
```

Εικόνα 53:Είσοδος/'Εξοδος RegisterAsAlive

Το SOAP μήνυμα κλήσης και απάντησης της μεθόδου

```
POST /storemanager/posserver.asmx HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
SOAPAction: "http://hit.com.gr/RegisterAsAlive"


<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

 <soap:Body>

  <RegisterAsAlive xmlns="http://hit.com.gr/">

   <posNo>string</posNo>

  </RegisterAsAlive>

 </soap:Body>

</soap:Envelope>

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: length


<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

 <soap:Body>

  <RegisterAsAliveResponse xmlns="http://hit.com.gr/">

   <RegisterAsAliveResult>boolean</RegisterAsAliveResult>

  </RegisterAsAliveResponse>

 </soap:Body>

</soap:Envelope>
```

Εικόνα 54: SOAP RegisterAsAlive

Κεφάλαιο 5. Επίλογος

Στην παρούσα εργασία παρουσιάστηκαν αναλυτικά τα Web Services όσον αφορά στην αρχιτεκτονική τους, στη χρησιμότητά τους, στις τεχνολογίες που χρησιμοποιούν, στα εργαλεία που χρησιμοποιούνται για το σχεδιασμό, την ανάλυση και την υλοποίησή τους.

Επιπρόσθετα, αναπτύχθηκαν δύο Web Services. Το ένα δημιουργήθηκε με σκοπό να διασυνδέονται εξωτερικά συστήματα (Web Sites ή mobile applications) με ένα κεντρικό σύστημα παραγγελιοληψίας ώστε τα πρώτα να μπορούν να καταχωρούν παραγγελίες, να αντλήσουν πληροφορίες σχετικά με στοιχεία πελατών, να γνωρίζουν το στάδιο που βρίσκεται η παραγγελία του πελάτη και ποια καταστήματα παραδίδουν και σε τι χρόνους, Το δεύτερο web service σχεδιάστηκε ώστε να εξυπηρετεί μονάδες εστίασης που διαθέτουν υποκαταστήματα με σκοπό να παρακολουθείτε η οικονομική και λειτουργική πορεία τους καθώς και για να έχουν ένα ενιαίο σύστημα διαχείρισης των τομών και των τιμοκαταλόγων τους.

Η ανάπτυξη και των δύο διαδικτυακών υπηρεσιών βασίστηκε στο περιβάλλον ανάπτυξης Visual Studio και ως σύστημα διαχείρισης βάσεων δεδομένων χρησιμοποιήθηκε ο Microsoft SQL Server 2008 R2. Αρχικά, αναλύθηκαν οι πίνακες που απαιτούνταν για το σχεδιασμό της βάσης δεδομένων και μετά περιγράφηκαν οι μέθοδοι των υπηρεσιών.

Τα Web Services που αναπτύχθηκαν αποτελούν ένα εύχρηστο περιβάλλον με καλώς ορισμένους πίνακες και κλήσεις μεθόδων ώστε να μπορούν να χρησιμοποιηθουν για τη διασύνδεση παρόμοιων συστημάτων με ελάχιστο επιπλέον χρόνο παραμετροποίησης.

Βιβλιογραφία

- http://www.w3.org/TR/xmlschema-0/
- http://www.w3.org/TR/wsdl/
- http://www.w3.org/TR/soap/
- http://www.w3.org/TR/ws-arch/
- http://www.w3.org/XML/
- http://www.w3schools.com/xml/default.asp
- http://www.w3.org/DOM/
- http://www.xmlfiles.com/xsl/
- http://www.w3schools.com/dtd/default.asp
- R. Elmasri, S.B. Navathe "Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων", 4η έκδοση
- Συστήματα Βάσεων Δεδομένων - Η Πλήρης Θεωρία των Βάσεων Δεδομένων - Silberschatz, Korth, Sudarshan - Εκδόσεις Μ.Γκιούρδας - 4η Έκδοση
- I. Sommerville, Software Engineering (8th Edition). Addison-Wesley, 2006.
- Troelsen, Pro C# 2008 and the .NET 3.5 Platform (4rd Edition). Apress, 2007.
- M. MacDonald, Beginning ASP.NET 3.5 in C# 2008: From Novice to Professional (2nd Edition). Apress, 2007.
- C. Darie, Beginning ASP.NET E-Commerce in C#: From Novice to Professional. Apress,2009.
- M. MacDonald, Pro ASP.NET 3.5 in C# 2008 (Second Edition). Apress, 2007.
- K. Delaney et al, Microsoft SQL Server 2008 Internals. Microsoft Press, 2009.
- E. Castro, HTML, XHTML, and CSS (6th Edition). Peachpit Press, 2006.
- Υπηρεσίες Παγκόσμιου Ιστού και Υπηρεσιοστρεφείς Αρχιτεκτονικές (Web Services and SOA). Μαρίνος Γ. Θεμιστοκλέους, Βασιλική Γ. Μαντζάνα

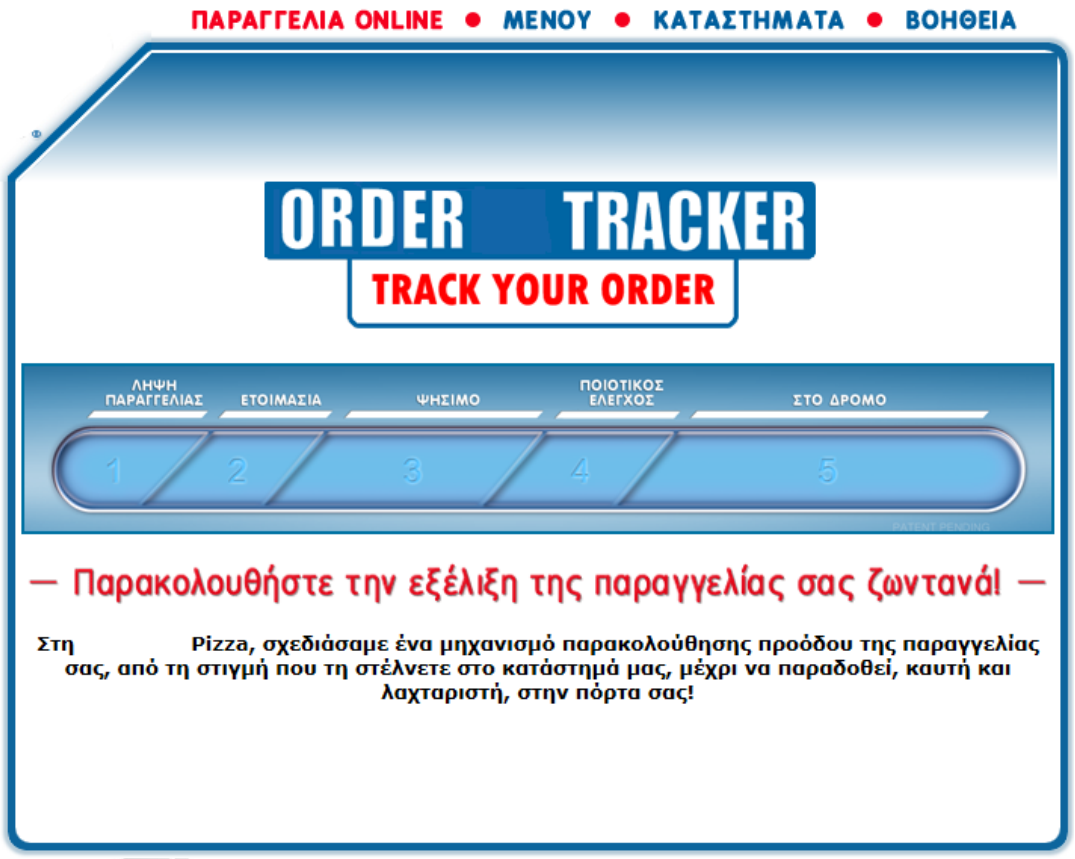
Παράρτημα

1.Οθόνες για το InternetDeliveryService

Ακολουθούν λειτουργιές του e-shop κατά τις οποίες γίνονται κλήσεις μεθόδων του Web Service InternetDeiveryService.

1.1.Πορεία παραγγελίας

Όταν ο πελάτη ζητήσει να δει που βρίσκεται μια παραγγελία που έχει κάνει ήδη τότε γίνεται κλήση της μεθόδου GetOrderStatus όπου επιστρέφει το status της παραγγελίας

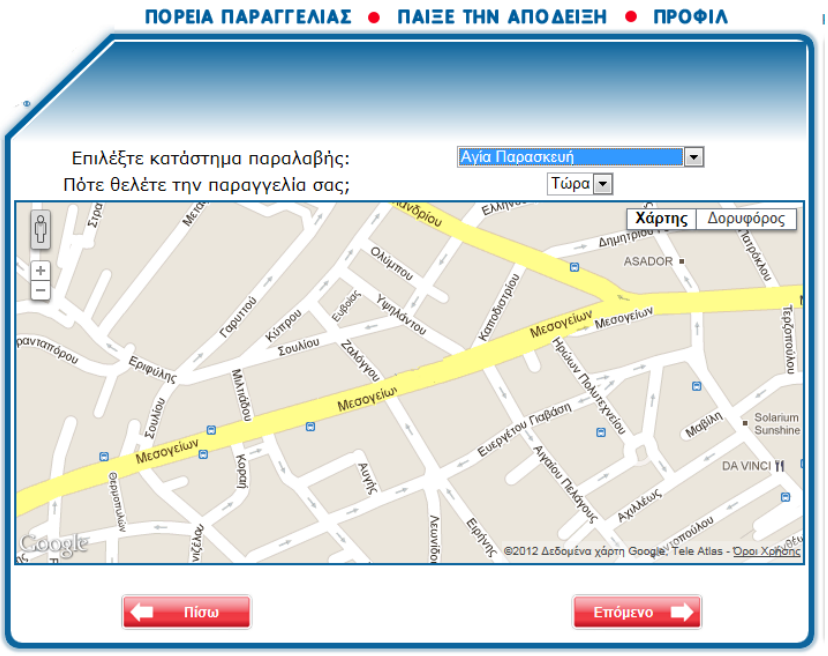


Εικόνα 55: Διαδικασία κλήσης GetOrderStatus

1.2.Επιλογή καταστήματος εξυπηρέτησης.

Κατά την εισαγωγή του ο πελάτης επιλέγει από πιο κατάστημα θέλει να εξυπηρετηθεί. Αμέσως μετά την επιλογή γίνεται κλήση της μεθόδου ReqOutofOrderItem όπου επιστρέφονται στο e-shop

- Εάν το κατάστημα είναι ανοιχτό και εξυπηρετεί
- Οι χρόνοι εξυπηρέτησης του καταστήματος
- Πιθανές ελλείψεις προϊόντων του καταστήματος



Εικόνα 56:Διαδικασία κλήσης ReqOutofOrderItems

1.3.Καταχώρηση Νέας Παραγγελίας.

Όταν ο πελάτης ολοκληρώσει την παραγγελία του γίνεται κλήση της μεθόδου ReqNewOrder όπου γίνεται η καταχώρηση της παραγγελίας του πελάτη στο κεντρικό σύστημα παραγγελιών και η μεταφορά του στο κατάστημα που πρέπει να παρασκευαστεί η παραγγελία:

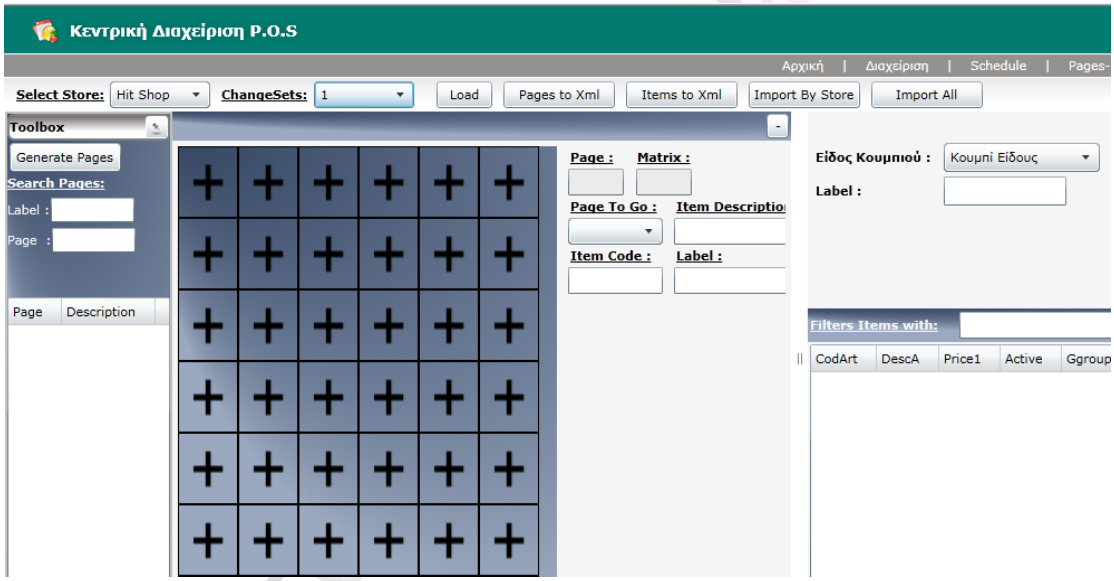
Εικόνα 57:Διαδικασία κλήσης ReqNewOrder

2.Οθόνες για το Storemanager

Παρακάτω φαίνονται οι οθόνες του διαχειριστικού εργαλείου που κάνουν κλήση αυτού του WebService.

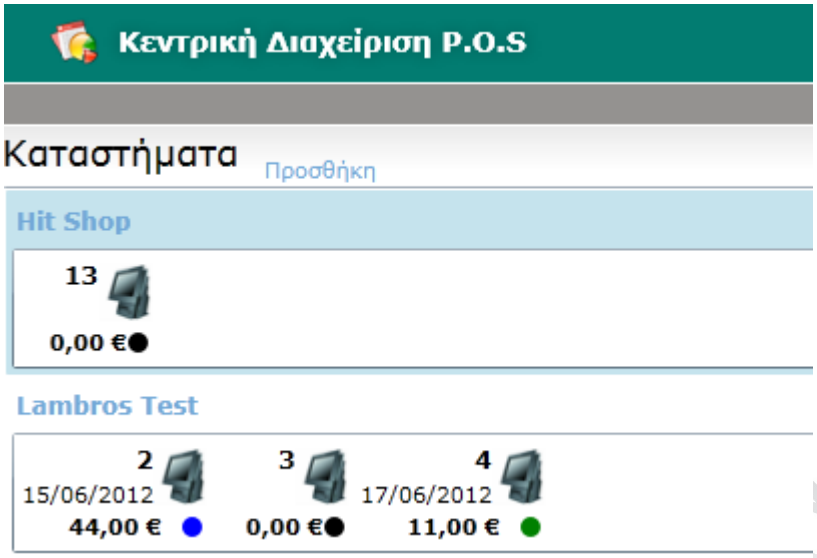
2.1 Κλήση της διαδικασίας HasPosNewPriceList

Στην παρακάτω οθόνη ορίζονται ο κατάλογος και η εμφάνιση των ειδών για το κατάστημα που έχει επιλεχθεί προς αναβάθμιση. Με την ολοκλήρωση των αλλαγών καλείται η HasPosNewPriceList που ενημερώνει το εκάστοτε σημείο πώλησης ότι έχει αλλαγές τιμών/ειδών.



2.2 Κλήση της διαδικασίας RegisterAsAlive

Για να εμφανιστούν εδώ μηχανήματα καλείται η μέθοδος RegisterAsAlive για να φανεί ποια σημεία ώλησης της αλυσίδας είναι ανοιχτά την παρούσα στιγμή.



2.3 Κλήση της διαδικασίας GetHourlyAudit

Ζητώντας τις ωριαίες πωλήσεις για κάποιο σημείο πώλησης καλείται και η μέθοδος GetHourlyAudit που μεταφέρει τις ωριαίες πωλήσεις στην κεντρική ΒΔ της εταιρίας.