Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή εργασία

Δημιουργία Online-Φαρμακείου

Εμμανουήλ Τσατσάκης (Α.Μ. 832)

Επιβλέπων Καθηγητής: Νικόλαος Παπαδάκης

ΗΡΑΚΛΕΙΟ

2016

Abstract

The rapid development of technologies related to the creation of dynamic web gives developers many new possibilities to design and implement databases and demanded information systems, since nowadays advanced functionality is a given.

The health sector is quite a fertile ground for the use of computer applications, given the large volume of data required to monitor the progress of the business and the accuracy and the validation of the drugs.

This thesis describes the steps of developing an application for an online pharmacy and through them analyzes the possibilities provided by the new technologies. A computer application using a relational database for data storage and modern internet technologies is developed to create a user interface management and data entry. It provides the opportunity to orders drugs through the online store for the simple users but also the admin to view and manage the storage and the demands of the pharmacy.

Σύνοψη

Η ραγδαία εξέλιξη των τεχνολογιών που σχετίζονται με την δημιουργία δυναμικών ιστοσελίδων, σχεσιακών βάσεων δεδομένων και πληροφοριακών συστημάτων δίνει στους προγραμματιστές πολλές νέες δυνατότητες αλλά ταυτόχρονα αλλάζει και τα κριτήρια βάση των οποίων σχεδιάζονται.

Ο τομέας της υγείας είναι ένα αρκετά πρόσφορο έδαφος για την χρήση εφαρμογών μηχανογράφησης, δεδομένου του μεγάλου όγκου δεδομένων που απαιτούνται για την παρακολούθηση της κίνησης της επιχείρησης, των αποθεμάτων των φαρμάκων και την σωστή διαχείριση αυτών.

Σε αυτή την εργασία αναπτύσσουμε μία εφαρμογή online φαρμακείου και συστήματος μηχανογράφησης με τη χρήση μίας σχεσιακής βάσης για την αποθήκευση των δεδομένων και σύγχρονων τεχνολογιών διαδικτύου για την δημιουργία ενός εύχρηστου περιβάλλοντος διαχείρισης και εισαγωγής των δεδομένων. Επίσης, παρέχεται η δυνατότητα παραγγελιών απλών χρηστών και ανάλογη διαχείριση από την πλευρά του διαχειριστή του συστήματος.

Contents

1 Εισαγωγή

1.1 Περίληψη

Σε αυτή την εργασία θα μελετήσουμε την διαδικασία ανάπτυξης μίας εφαρμογής μηχανογράφησης ενός φαρμακείου. Συγκεκριμένα θα αναπτυχθεί μία πολυχρηστικής εφαρμογής χρησιμοποιώντας τις πιο σύγχρονες και ευρέως διαδεδομένες τεχνικές (state of the art) της Java 2 Enterprise Edition και της Java 8.

Η εφαρμογή θα εκτελεί τις βασικές λειτουργίες ενός εξειδικευμένου συστήματος Enterprise Resource Management, με διαχείριση των προϊόντων και των αποθεμάτων τους, διαχείριση των συνταγογραφημένων και ελεύθερων πωλήσεων καθώς και στατιστικών αναφορών. Επίσης παρέχεται η δυνατότητα online διεκπεραίωση παραγγελιών από χρήστες / πελάτες του φαρμακείου και ανάλογη χρήση και διαχείριση δεδομένων από πλευράς διαχειριστή.

Η ανάπτυξη θα βασιστεί στο μοντέλο client - server, με τους χρήστες να έχουν πρόσβαση στις λειτουργίες τους μέσω ενός γραφικού περιβάλλοντος. Η αρχιτεκτονική της εφαρμογής θα βασίζεται στην αρχιτεκτονική MVC, ενώ η αποθήκευση των δεδομένων θα γίνεται σε μία σχεσιακή βάση δεδομένων.

1.2 Κίνητρο για την διεξαγωγή της εργασίας

Το internet είναι μία τεχνολογία που χρησιμοποιείται κατά κόρων στις μέρες μας από πάρα πολλούς ανθρώπους. Οι σελίδες σε αυτό χρησιμοποιούνται για πάρα πολλούς στόχους και επιτρέπουν πέρα από την προβολή πληροφοριών και την ανάπτυξη συστημάτων διαχείρισης και μηχανογράφησης.

Βασικό κίνητρο είναι να ελέγξουμε πόσο μπορεί αυτή η τάση να επηρεάσει και τον κόσμο της υγείας μέσω σωστής διαχείρισης και πρόσβαση χρηστών σε online αγορές φαρμάκων και προϊόντων υγείας.

Εκπαιδευτικό Κίνητρο για την διεξαγωγή αυτής της εργασίας ήταν η εξοικείωση μας με τις σύγχρονες τεχνολογίες κατασκευής δυναμικών ιστοσελίδων καθώς οι τεχνολογίες που χρησιμοποιούνται για την υλοποίηση της πτυχιακής εργασίας παρέχουν αρκετές δυνατότητες στους προγραμματιστές και τους σχεδιαστές που ασχολούνται με το διαδίκτυο ώστε να δημιουργήσουν εφαρμογές οι οποίες είναι πέρα για πέρα από λειτουργικές και στο τέμπο των σημερινών απαιτήσεων γνωσιακού επιπέδου.

1.3 Σκοπός και στόχοι εργασίας

Σκοπός της εργασίας είναι η δημιουργία μίας δυναμικής ιστοσελίδας για ένα φαρμακείο με ηλεκτρονική υπόσταση. Μέσα από την ανάπτυξη της εφαρμογής στοχεύουμε να παρουσιάσουμε κάποια βασικά στοιχεία των τυπικών τεχνολογιών που χρησιμοποιούνται στο διαδίκτυο και να αξιολογήσουμε τις δυνατότητές τους και την ευκολία χρήσης τους.

Με την εργασία μας θέλουμε να δείξουμε πως φτιάξαμε την εφαρμογή μας και πώς χρησιμοποιήσαμε τις γλώσσες προγραμματισμού HTML, Java, JavaScript και SQL καθώς και τεχνολογίες που τις υποστηρίζουν για να πετύχουμε τον σκοπό μας. Θα δούμε διεξοδικά όλα τα βήματα που ακολουθήσαμε για να φτιάξουμε το τελικό αποτέλεσμα.

Έπειτα θα πούμε τα συμπεράσματά μας για αυτές τις τεχνολογίες και τι μπορεί να γίνει για να βελτιωθεί ακόμη περισσότερο η εφαρμογή μας από μελλοντικούς ερευνητές.

1.4 Δομή εργασίας

Τα περιεχόμενα της εργασίας είναι τα εξής:

Το πρώτο κεφάλαιο είναι το παρόν, με γενικές πληροφορίες και καθορισμό στόχων και κινήτρων για την εργασία αυτή.

Το δεύτερο κεφάλαιο επεξηγεί βασικές έννοιες και ορισμούς που αποτέλεσαν βασικά θεμέλια στην πληροφορική. Είναι βασικοί ορισμοί που απαιτούν κατανόηση και ενημερώνουν των αναγνώστη της εργασίας ώστε να αντιληφθεί τις απαιτήσεις που δημιουργήθηκαν από τους χρήστες και απαιτητικούς πελάτες της εποχής σχετικά με την τεχνολογία και την επίτευξη των αναγκών για άμεση και γρηγορότερη επεξεργασία πληροφοριών.

Το τρίτο κεφάλαιο αναλύει τις τεχνολογίες και γλώσσες προγραμματισμού που χρησιμοποιήθηκαν και δίνει μία σύντομη επισκόπηση για τον σκοπό και τη λειτουργία της κάθε μίας από αυτές.

Το τέταρτο κεφάλαιο αναλύει το στήσιμο της εφαρμογής δίνοντας τεχνικές λεπτομέρειες υλοποίησης και ανάλυσης του σχεδιασμού της βάσης του συστήματος και της εγκατάστασης και λειτουργίας των απαιτούμενων server και αλληλεπίδραση συστημάτων και τεχνολογιών.

Το πέμπτο κεφάλαιο αναλύει τις λειτουργίες της εφαρμογής δίνοντας με κυκλικό τρόπο τις τεχνολογίες, και τον απαιτούμενο κώδικα σε πλαίσιο υλοποιήσεων των απαιτήσεων και λειτουργιών του συστήματος εφαρμογής. Περιλαμβάνει στιγμιότυπα εγχειρίδιου χρήσης σε συσχέτιση με τον κώδικα που βασίζεται η εφαρμογή.

Το έκτο κεφάλαιο περιέχει τα συμπεράσματά μας και κάποιες προτάσεις για μελλοντική εξέταση.

Το έβδομο κεφάλαιο είναι η βιβλιογραφία.

2 Γενικές και Βασικές Έννοιες

2.1 Μηχανογράφηση

Ως ορισμό μπορούμε να δεχτούμε την χρησιμοποίηση μηχανών σύγχρονης τεχνολογίας στην οργάνωση επιχειρήσεων και οργανισμών, για τη συγκέντρωση, ταξινόμηση και επεξεργασία στοιχείων.

Στην εποχή που ζούμε, η σπουδαιότητα μηχανογράφησης παίζει καθοριστικό ρόλο ως ανταγωνιστικό πλεονέκτημα απέναντι τόσο μέσα στην ίδια την εταιρία είτε οργανισμό όσο και εξωτερικά. Η σωστή διαχείριση της πληροφορίας αλλά και ο συνυφασμός της φύσης μας με την τεχνολογία μπορούν να δώσουν το καλλίτερο αποτέλεσμα σχετικά με το πλάνο πορείας οργάνωσης και εξέλιξης. Η τεχνολογία προχωράει με γοργούς ρυθμούς και δεν επιτρέπει λάθος χειρισμούς τόσο για την σωστή ενημέρωση όσο και για την σωστή επεξεργασία των πόρων που υπάρχουν. Δεν χρειάζεται να γίνεται σπατάλη πόρων αλλά και καταπόνηση πνευματικής υγείας.

Η σωστή λύση είναι η σωστή μηχανογράφηση με την σωστή μηχανοργάνωση κάθε φορέα, οργανισμού και εταιρείας.

2.2 Πληροφοριακό Σύστημα

Αρχικά σε επιχειρήσεις και άλλους οργανισμούς ,ο εσωτερικός έλεγχος γινόταν χειροκίνητα και μόνο περιοδικά ως παράγωγο του λογιστικού συστήματος και με κάποιες πρόσθετες στατιστικές πληροφορίες που δινόταν για την απόδοση της διαχείρισης καθυστερημένα και περιορισμένα. Τα δεδομένα οργανώνονταν με μη αυτόματο τρόπο και σύμφωνα με τις απαιτήσεις και τις ανάγκες του οργανισμού. Με την ανάπτυξη της πληροφορικής , η πληροφορία άρχισε να διαχωρίζεται από τα δεδομένα και αναπτύχθηκαν συστήματα για την παραγωγή και την οργάνωση λήψεων ,περιλήψεων ,σχέσεων και γενικεύσεων βασισμένων στα δεδομένα.

Πληροφοριακά συστήματα (*Information Systems* ή *IS*) ονομάζεται ένα σύνολο διαδικασιών, ανθρώπινου δυναμικού και αυτοματοποιημένων υπολογιστικών συστημάτων, που προορίζονται για τη συλλογή, εγγραφή, ανάκτηση, επεξεργασία, αποθήκευση και ανάλυση πληροφοριών. Τα συστήματα αυτά μπορούν να περιλαμβάνουν λογισμικό, υλικό και τηλεπικοινωνιακό σκέλος.

Τα πληροφοριακά συστήματα αποτελούν το μέσο για την αρμονική συνεργασία ανθρώπινου δυναμικού, δεδομένων, διαδικασιών και τεχνολογιών πληροφορίας και επικοινωνιών. Προέκυψαν ως γέφυρα μεταξύ των πρακτικών εφαρμογών της επιστήμης υπολογιστών και του επιχειρηματικού κόσμου. Κάθε ειδικό πληροφοριακό σύστημα έχει ως στόχο την υποστήριξη των επιχειρήσεων, τη διαχείριση και λήψη αποφάσεων. Σε μια ευρεία έννοια, ο όρος χρησιμοποιείται για να αναφερθεί όχι μόνο στην τεχνολογία της πληροφορίας και της επικοινωνίας (ΤΠΕ), που ένας οργανισμός χρησιμοποιεί, αλλά στο τρόπο με τον οποίο οι άνθρωποι αλληλοεπιδρούν με αυτή την τεχνολογία για την υποστήριξη των επιχειρηματικών διαδικασιών.

Ως εκ τούτου, τα πληροφοριακά συστήματα σχετίζονται με τα συστήματα διαχείρισης βάσης δεδομένων από τη μία πλευρά και με τα συστήματα δραστηριότητας από την άλλη. Ένα πληροφοριακό σύστημα είναι μια μορφή επικοινωνίας του συστήματος στο οποίο τα δεδομένα αντιπροσωπεύουν και υποβάλλονται σε επεξεργασία ως μια μορφή κοινωνικής μνήμης. Ένα πληροφοριακό σύστημα μπορεί επίσης να θεωρηθεί ως ημι-επίσημη γλώσσα που υποστηρίζει τις ανθρώπινες λήψεις αποφάσεων και δράσης.

2.3 Πληροφοριακό Σύστημα Διοίκησης

Ένα Πληροφοριακό Σύστημα Διοίκησης παρέχει πληροφορίες που χρειάζονται για να διαχειρίζονται οι οργανισμοί αποδοτικά και αποτελεσματικά. Τα Πληροφοριακά Συστήματα Διοίκησης περιλαμβάνουν τρεις βασικές πηγές: ανθρώπους ,τεχνολογία και πληροφορία. Τα Πληροφοριακά Συστήματα Διοίκησης είναι διακριτά από τα άλλα πληροφοριακά συστήματα τα οποία χρησιμοποιούνται για την ανάλυση λειτουργικών λειτουργιών στον οργανισμό. Ακαδημαϊκά, ο όρος χρησιμοποιείται συνήθως για να αναφερθεί στην ομάδα των μεθόδων διαχείρισης πληροφοριών που είναι συνδεδεμένες με την αυτοματοποίηση ή στηρίζουν την ανθρώπινη λήψη αποφάσεων.

2.4 Σύστημα Διαχείρισης Επιχειρησιακών Πόρων (ERP)

Τα συστήματα Διαχείρισης Επιχειρησιακών πόρων (ERP) αποτελούν την καρδιά των πληροφοριακών συστημάτων κάθε επιχείρησης ή οργανισμού και τη βάση για την ομαλή λειτουργία τους, αλλά και τη λήψη αποφάσεων της διοίκησης.

Τα ERP βασίζονται σε ιδιοπαραγόμενα προϊόντα ή σε διεθνείς πλατφόρμες και συνοδεύονται από υπηρεσίες υψηλής ποιότητας, με στόχο να παραδοθεί μία απόλυτα λειτουργική λύση προσαρμοσμένη στις ιδιαίτερες ανάγκες κάθε επιχειρησιακού περιβάλλοντος.

2.5 Βάση Δεδομένων

Βάση δεδομένων ονομάζουμε μία συλλογή από συστηματικά μορφοποιημένα σχετιζόμενα δεδομένα στα οποία είναι δυνατή η ανάκτηση δεδομένων μέσω αναζήτησης κατ' απαίτηση. Ο Αμερικανός επιστήμονας υπολογιστών Jim Gray έχει γράψει για τις βάσεις δεδομένων: «Όταν οι άνθρωποι χρησιμοποιούν τις λέξεις βάση δεδομένων, διατυπώνουν στην ουσία ότι τα δεδομένα πρέπει να αυτοπροσδιορίζονται και να έχουν μια σχηματική δομή. Αυτό ακριβώς περιγράφουν οι λέξεις βάση δεδομένων».

Ειδικότερα, στην επιστήμη της πληροφορικής και στην καθημερινή χρήση των ηλεκτρονικών υπολογιστών, με τον όρο βάσεις δεδομένων αναφερόμαστε σε οργανωμένες, διακριτές συλλογές σχετιζόμενων δεδομένων ηλεκτρονικά και ψηφιακά αποθηκευμένων, στο λογισμικό που χειρίζεται τέτοιες συλλογές (Σύστημα Διαχείρισης Βάσεων Δεδομένων, ή DBMS) και στο γνωστικό πεδίο που το μελετά. Πέρα από την εγγενή της ικανότητα να αποθηκεύει δεδομένα, η βάση δεδομένων παρέχει μέσω του σχεδιασμού και του τρόπου ιεράρχησης των δεδομένων, τα αποκαλούμενα συστήματα διαχείρισης περιεχομένου, δηλαδή τη δυνατότητα γρήγορης άντλησης και ανανέωσης των δεδομένων

2.6 Σύστημα Διαχείρισης Βάσης Δεδομένων

Με τον όρο γνωστό ως Database Management system (DBMS) εννοείται είτε κάποιο λογισμικό μέσω του οποίου γίνεται η δημιουργία, η διαχείριση, η συντήρηση και η χρήση μιας ηλεκτρονικής βάσης δεδομένων, ανάλογα με τον τύπο βάσης δεδομένων που επιλέγεται ή ένα σύνολο αλληλοσχετιζόμενων προγραμμάτων που τρέχουν και διαχειρίζονται τα δεδομένα μιας τέτοιας βάσης. Το λογισμικό χρησιμοποιεί στερεότυπες (Standard) μεθόδους καταλογοποίησης, ανάκτησης, και εκτέλεσης ερωτημάτων σχετικών με τα δεδομένα. Το σύστημα διαχείρισης οργανώνει τα εισερχόμενα δεδομένα με τρόπους χρησιμοποιήσιμους από εξωτερικούς χρήστες

Ιδωμένο από μία άλλη οπτική γωνία, το σύστημα διαχείρισης βάσης δεδομένων είναι ένας διαχειριστής αρχείων (file manager) που διαχειρίζεται δεδομένα σε βάσεις δεδομένων παρά αρχεία σε συστήματα αρχείων, τα οποία είναι μία άλλη μορφή βάσης δεδομένων.

2.7 Στατικές ιστοσελίδες

Στατική ιστοσελίδα (static ή flat web-page) ονομάζεται μια ιστοσελίδα της οποίας το περιεχόμενο παραδίδεται στον χρήστη όπως ακριβώς αποθηκεύεται στον εξυπηρετητή σελίδων (web Server). Τα περιεχόμενα μιας στατικής ιστοσελίδας εμφανίζονται με την ίδια μορφή σε όλους τους χρήστες, με τη μορφή με την οποία βρίσκονται αποθηκευμένα στο σύστημα αρχείων του εξυπηρετητή σελίδων. Οι στατικές ιστοσελίδες είναι συνήθως αποθηκευμένες σε μορφή HTML και μεταφέρονται χρησιμοποιώντας το πρωτόκολλο HTTP.

Οι στατικές ιστοσελίδες είναι κυρίως κατάλληλες για μικρού μεγέθους ιστοχώρους καθώς παρουσιάζουν δυσκολίες στη συντήρηση όταν χρησιμοποιούνται σε μεγάλους ιστοχώρους με πολλές υποσελίδες. Σε αυτή την περίπτωση, συνήθως χρησιμοποιούνται δυναμικές ιστοσελίδες οι οποίες ενημερώνονται εύκολα με μια απλή τροποποίηση στη βάση δεδομένων.

2.7.1 Πλεονεκτήματα στατικών ιστοσελίδων

Εξαιτίας της απλής λογικής τους οι στατικές ιστοσελίδες έχουν κάποια πλεονεκτήματα:

- Η αρχική τους κατασκευή είναι πιο φτηνή και πιο εύκολη (ιδίως όταν πρόκειται για μικρού μεγέθους)
- Οι σελίδες τους συνήθως παραδίδονται γρήγορα επειδή ο κώδικας html για κάθε σελίδα είναι πολύ απλός
- Δε χρειάζεται ειδικό λογισμικό στον εξυπηρετητή ιστοσελίδων για τη δημοσίευση στατικών ιστοσελίδων

2.7.2 Μειονεκτήματα στατικών ιστοσελίδων

Όμως το γεγονός πως αποτελούνται από λίγα τμήματα οδηγεί μοιραία και σε κάποια μειονεκτήματα.

- Οι πληροφορίες που περιέχουν μπορούν γρήγορα να γίνουν ξεπερασμένες και να μην ανταποκρίνονται στην πραγματικότητα τη χρονική στιγμή που ο χρήστης θα επισκεφτεί τη σελίδα. Αυτό θα μπορούσε να βλάψει μια επιχείρηση περισσότερο απ' ότι να βελτιώσει την εικόνα της.
- Δεν ενθαρρύνει τους επισκέπτες να επιστρέφουν για τις τελευταίες πληροφορίες και μπορεί να αναρωτιούνται αν η επιχείρηση που παρουσιάζεται στη σελίδα είναι ενεργή
- Δεν είναι εύκολη η δια δραστικότητα με τον χρήστη
- Εάν υπάρχουν πολλές σελίδες η αρχική επένδυση μπορεί να είναι υψηλότερη απ' ότι για μια δυναμική ιστοσελίδα
- Υψηλότερο κόστος συντήρησης, καθώς θα χρειαστεί κάποιος προγραμματιστής ακόμα και για κάποιες μικρές αλλαγές ή ενημερώσεις. Αυτό μπορεί επίσης να είναι ένα πρόβλημα αν οι αλλαγές πρέπει να γίνουν σε σύντομο χρονικό διάστημα, ενώ με μια δυναμική ιστοσελίδα μπορούν να γίνουν αλλαγές στο περιεχόμενο οποιαδήποτε στιγμή.
- Δεν υπάρχουν καταχωρήσεις χρηστών, ή προηγμένες λειτουργίες που συνήθως συνδέονται με δυναμικές ιστοσελίδες.
- Αδυναμία προσαρμογής του περιεχομένου της ιστοσελίδας στις ανάγκες του κάθε επισκέπτη.

2.8 Δυναμικές ιστοσελίδες

Μια δυναμική ιστοσελίδα (dynamic web-page) περιέχει ιστοσελίδες οι οποίες παράγονται δυναμικά. Κάθε φορά που ένας χρήστης αποκτά πρόσβαση σε έναν δυναμικό ιστότοπο, ο κώδικας html παράγεται σε πραγματικό χρόνο και στέλνεται στον περιηγητή του χρήστη. Το περιεχόμενό τους συνήθως αντλείται από μια βάση δεδομένων, έτσι κάθε φορά που χρειάζεται να γίνει κάποια προσθαφαίρεση σε αυτό δεν είναι ανάγκη να επεξεργάζεται κανείς την ιστοσελίδα, αλλά απλά διαχειρίζεται έμμεσα το περιεχόμενο στη βάση δεδομένων και οι αλλαγές γίνονται αυτόματα στη σελίδα.

Η δυναμική ιστοσελίδα δημιουργείται δυναμικά από ένα σενάριο εντολών το οποίο λαμβάνεται από τον εξυπηρετητή και εκτελείται τοπικά στον πελάτη ή στον εξυπηρετητή ή και στον πελάτη και στον εξυπηρετητή.

Ανάλογα με τη λειτουργικότητά τους οι δυναμικές ιστοσελίδες μπορούν να κατηγοριοποιηθούν σε:

- Συστήματα Διαχείρισης Περιεχομένου (CMS)
- Ηλεκτρονικά καταστήματα
- Forums
- Blogs

Υπάρχουν πολλές περισσότερες κατηγορίες αλλά οι περισσότεροι δυναμικοί ιστότοποι που υπάρχουν αυτή τη στιγμή στο διαδίκτυο ανήκουν στις παραπάνω κατηγορίες ή αποτελούν συνδυασμό αυτών.

2.8.1 Δυναμική ιστοσελίδα τοπικά στον χρήστη

Στην περίπτωση αυτή η δυναμική ιστοσελίδα εκτελείται τοπικά στην πλευρά του πελάτη από τον περιηγητή του χρήστη και όχι στον εξυπηρετητή. Το σενάριο εντολών για μια δυναμική ιστοσελίδα με client – side scripting συνήθως περιέχεται σε ένα αρχείο HTML ή XHTML αλλά μπορεί επίσης να περιέχεται σε ένα ξεχωριστό αρχείο στο οποίο το έγγραφο (ή έγγραφα) που το χρησιμοποιούν κάνουν αναφορά. Μετά από αίτημα τα απαραίτητα αρχεία στέλνονται στον υπολογιστή του χρήστη από τον διακομιστή web (server) στον οποίο είναι αποθηκευμένα. Το πρόγραμμα περιήγησης του χρήστη εκτελεί το σενάριο εντολών και στη συνέχεια εμφανίζει το έγγραφο συμπεριλαμβανομένων οποιονδήποτε ορατών παραμετροποιήσεων περιέχονται στο σενάριο.

Τα σενάρια εντολών που εκτελούνται στην πλευρά του χρήστη μπορούν επίσης να περιέχουν οδηγίες για τον περιηγητή του ώστε να τις ακολουθεί ως ανταπόκριση σε κάποιες ενέργειες του χρήστη (πχ το πάτημα ενός κουμπιού, mouse over κ.α.). Συχνά αυτές οι οδηγίες μπορούν να εφαρμόζονται χωρίς περαιτέρω επικοινωνία με τον διακομιστή.

2.8.2 Δυναμική ιστοσελίδα στον εξυπηρετητή

Στην περίπτωση αυτή το σενάριο εντολών της ιστοσελίδας, γραμμένο σε γλώσσες όπως η Perl, PHP, ASP.NET , Java κ.α., εκτελείται από τον web server όταν ο χρήστης ζητά ένα έγγραφο. Παράγει έξοδο σε μορφή κατανοητή από web browsers (συνήθως HTML), η οποία στη συνέχεια στέλνεται στον υπολογιστή του χρήστη. Τα έγγραφα που παράγονται από server – side scripts μπορούν επίσης να περιέχουν client – side scripts.

Πολλές φορές οι δυναμικές ιστοσελίδες αποθηκεύονται προσωρινά στην κρυφή μνήμη του server όταν αναμένονται λίγες ή καθόλου αλλαγές σε αυτές, και αποστέλλονται κατευθείαν στον πελάτη όταν τις ζητήσει. Με τον τρόπο αυτό αποφεύγεται η υπερφόρτωση του εξυπηρετητή και έχουμε καλύτερους χρόνους φόρτωσης της σελίδας.

2.8.3 Δημιουργία δυναμικής ιστοσελίδας στον πελάτη και στον εξυπηρετητή

Η τεχνολογία AJAX (asynchronous JavaScript & XML) είναι μια τεχνολογία με την οποία οι web εφαρμογές μπορούν να ανταλλάσσουν δεδομένα με έναν web server ασύγχρονα, χωρίς να αλλάζει η εμφάνιση και η συμπεριφορά της σελίδας. Για τη δημιουργία της δυναμικής ιστοσελίδας η τεχνολογία AJAX χρησιμοποιεί σενάρια εντολών και στον πελάτη και στον εξυπηρετητή. Στην περίπτωση αυτή ο server επιστρέφει τα δεδομένα που ζητήθηκαν τα οποία μορφοποιούνται από ένα σενάριο εντολών που εκτελείται στην πλευρά του πελάτη. Η τεχνολογία αυτή μειώνει το χρόνο φόρτωσης δεδομένων από τον διακομιστή, αφού ο πελάτης δε ζητάει επαναφόρτωση ολόκληρης της σελίδας αλλά μεταδίδεται μονάχα το περιεχόμενο το οποίο θα αλλάξει. Ένα ενδιαφέρον παράδειγμα δικτυακής εφαρμογής η οποία χρησιμοποιεί την τεχνολογία AJAX είναι οι χάρτες της Google.

2.8.4 Πλεονεκτήματα δυναμικών ιστοσελίδων

Εδώ παρουσιάζουμε τα πλεονεκτήματα των δυναμικών ιστοσελίδων:

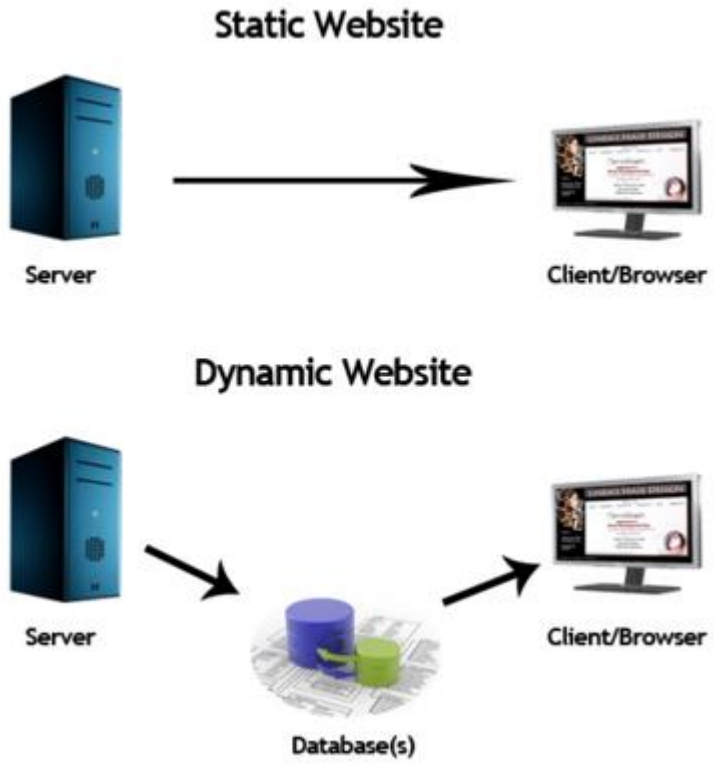
- Δυνατότητα άμεσης επέμβασης και τροποποίησης περιεχομένου ιστοσελίδας από τον ιδιοκτήτη ή τον διαχειριστή της
- Δεν απαιτούνται ιδιαίτερες υπολογιστικές γνώσεις για την συντήρηση υλικού
- Αλληλεπίδραση με τους χρήστες
- Εξοικονόμηση χρημάτων και πόρων
- Δεν υπάρχουν περιορισμοί στον όγκο που μπορεί να αποκτήσει μια ιστοσελίδα
- Εγκατεστημένη τεχνογνωσία σε παγκόσμιο επίπεδο, αφού οι πλατφόρμες που χρησιμοποιούνται είναι συνήθως ανοικτού κώδικα
- Περισσότεροι από ένας χρήστες μπορούν να αναπτύσσουν ταυτόχρονα την ιστοσελίδα.

2.8.5 Μειονεκτήματα

Τα επιπλέον τμήματα που απαιτούνται για την σωστή λειτουργία των δυναμικών ιστοσελίδων δημιουργούν κάποια προβλήματα:

- Μεγάλη εξάρτηση λειτουργίας της ιστοσελίδας με ένα πλήθος ιδιοτήτων του διακομιστή (server) στον οποίο πραγματοποιείται η φιλοξενία της ιστοσελίδας
- Δυσκολότερη αντιμετώπιση προβλημάτων και τεχνικών δυσκολιών
- Υψηλότερο κόστος αρχικής κατασκευής

2.9 Επιλογή στατικής ή δυναμικής ιστοσελίδας



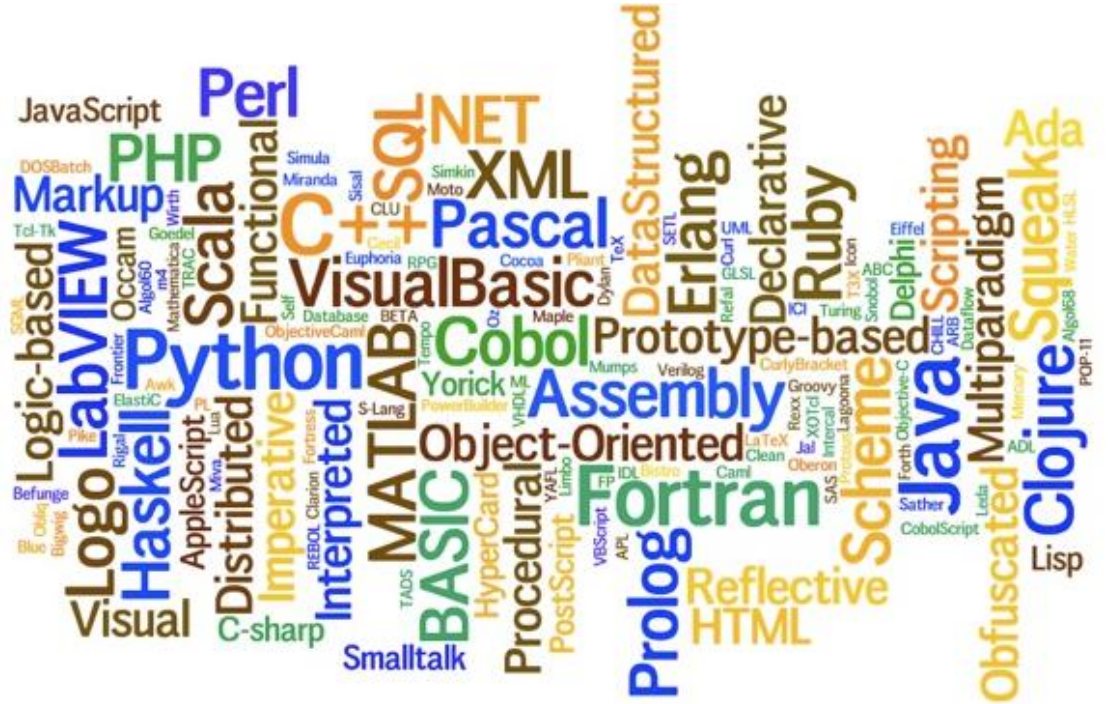
Αν αυτό που θέλουμε είναι η συχνή αναβάθμιση του περιεχομένου της ιστοσελίδας μας και η προσθήκη νέου, η δυναμική ιστοσελίδα είναι η σωστή επιλογή. Με αυτήν μας

παρέχονται δυνατότητες διαχείρισης μελών, ηλεκτρονικής ενημέρωσης των χρηστών (newsletter), online κρατήσεων ή πωλήσεων προϊόντων κ.α. Η δυναμική ιστοσελίδα παρέχει περισσότερες δυνατότητες στους διαχειριστές / ιδιοκτήτες της σελίδας και πολύ μεγαλύτερη αλληλεπίδραση με το χρήστη. Οι δυνατότητες αυτές όμως δεν είναι απαραίτητες για όλους. Αν το ζητούμενο είναι απλές υπηρεσίες όπως η διαφήμιση μιας επιχείρησης ή η απλή παρουσίαση κάποιου προϊόντος τότε η στατική ιστοσελίδα καλύπτει τις ανάγκες μας.

Όσον αφορά το κόστος, η στατική ιστοσελίδα είναι συνήθως πιο φτηνή λόγω ευκολίας στην κατασκευή ενώ οι δυναμικές ιστοσελίδες ως πιο περίπλοκες κοστίζουν περισσότερο. Εάν έχει αποφασιστεί η δυναμική ιστοσελίδα τότε η χρήση ενός open source συστήματος διαχείρισης περιεχομένου (CMS) είναι μια καλή λύση (ακόμα και αν αναθέσουμε τη δημιουργία σε κάποια εταιρεία) αφού οι εφαρμογές αυτές διατίθενται δωρεάν μέσω internet και επιπλέον είναι εύκολο να συντηρηθεί η σελίδα αργότερα, ακόμα και από άτομα που δεν έχουν προγραμματιστικές γνώσεις.

3 Τεχνολογίες και Γλώσσες Προγραμματισμού που χρησιμοποιήθηκαν

Οι Τεχνολογίες που θα παρουσιαστούν παρακάτω κρίθηκαν απαραίτητες για την υλοποίηση του πληροφοριακού συστήματος για το φαρμακείο.



3.1 HTML

Αρχικά η δομή της σελίδας που βλέπει ο χρήστης περιγράφεται με την γλώσσα HTML. Ο στόχος της HTML είναι να περιγράψει κάθε ένα στοιχείο που περιέχει η σελίδα βάση του τύπου του, δηλαδή αν είναι μία επικεφαλίδα, μία παράγραφος ή ένα πεδίο κειμένου. Τα στοιχεία είναι ιεραρχικά τοποθετημένα, δηλαδή ένα στοιχείο μενού (nav) μπορεί να περιέχει μία επικεφαλίδα (που θα περιγράφει το είδος του μενού) και μία λίστα με επιλογές.

Πέρα από μία βασική δομή που πρέπει να έχουν όλες οι σελίδες και κάποιους γενικούς κανόνες δεν υπάρχει κάποιος περιορισμός στη σειρά των στοιχείων ή στο ποιο στοιχείο μπορεί να περιέχεται μέσα σε κάποιο άλλο. Αυτό εξαρτάται από το πώς θέλουμε να φαίνεται και τι να περιέχει η σελίδα. Για παράδειγμα το περιεχόμενο του στοιχείου header περιγράφει τα περιεχόμενα ενός άλλου στοιχείου. Το header μπορεί να βρίσκεται μέσα στο στοιχείο που περιγράφει, να προηγείται ή να έπεται.

Επιπλέον το ίδιο το περιεχόμενο του header δεν είναι δεδομένο. Μπορεί να περιέχει ένα στοιχείο με κείμενο επικεφαλίδας h1, το οποίο περιέχει το μεγαλύτερο κείμενο τίτλου. Όμως αν το header αναφέρεται σε κάποιο εσωτερικό στοιχειό της σελίδας μπορεί αντί για h1 να περιέχει h2. Επίσης μπορεί να υπάρχει ένα στοιχείο img που δείχνει μία εικόνα, πχ το λογότυπο της σελίδας.

Από τα παραπάνω φαίνεται πως δεν υπάρχει σωστός ή λάθος τρόπος για την συγγραφή του κώδικα HTML, απλά ακολουθείται η λογική. Αυτό αναφέρεται βέβαια στη δομή της σελίδας και τη διάρθρωση των στοιχείων, καθώς οι συντακτικοί κανόνες είναι απαραίτητο να ακολουθούνται.

3.1.1 Εισαγωγή

Η βασική ανάγκη που κάλυψε η γλώσσα HTML και η αιτία δημιουργίας της ήταν η προβολή απλών σελίδων κειμένου και την σύνδεση τους με άλλες. Στην συνέχεια με την εξάπλωση του διαδικτύου, η ανάγκη κατασκευής πιο εντυπωσιακών σελίδων που θα ήταν και πιο φιλικές προς το χρήστη καθώς και τα τελευταία χρόνια ολόκληρων εφαρμογών είτε πληροφοριακών συστημάτων, δυνάμωσε την ανάγκη χρήσης την γλώσσας αυτής και λόγω απαιτήσεων και αναγκών φυσικά υλοποίησης εξελίχθηκε σε νέες εκδόσεις ώστε να μπορεί να υποστηρίζει τις απαιτήσεις υλοποίησης των προγραμματιστών και τις ανάγκες χρηστών.

Το πρότυπο της HTML αναθεωρήθηκε αρκετά με την τελευταία έκδοσή του 4.01 να έχει οριστικοποιηθεί το 1999. Καθώς οι απαιτήσεις από την γλώσσα αυξανόντουσαν αναπτύχθηκαν πολλές τεχνολογίες με σκοπό να καλύψουν τις ελλείψεις της γλώσσας οι οποίες με τη σειρά τους δημιούργησαν νέα προβλήματα, καθώς χρειάζεται να υποστηρίζονται από τα αντίστοιχα προγράμματα που χρησιμοποιούν οι χρήστες, κάτι που δεν ήταν δεδομένο.

Εξαιτίας ότι οι απαιτήσεις από την HTML αυξανόντουσαν όπως προαναφέρθηκε και παραπάνω, το πρότυπο της αναθεωρήθηκε αρκετά μετά την τελευταία έκδοση του 4.01 πριν το 1999 που οριστικοποιήθηκε. Η τεχνολογία και οι προγραμματιστικές απαιτήσεις επίτειναν τη ν κατάσταση για επόμενη έκδοση ακόμα περισσότερο καθώς νέες τεχνολογίες αναπτύχθηκαν με σκοπό να καλύψουν τις ελλείψεις που δημιουργόντουσαν ώστε να γίνεται σωστά η υποστήριξη προγραμμάτων και εμφάνισης δεδομένων. Η επόμενη έκδοση ήταν η 5 και γνωστή σε όλους μας HTML5, , η οποία άρχισε να αναπτύσσεται το 2004.

Η HTML5 είναι η νέα γλώσσα που προσφέρει έναν όγκο δυνατοτήτων με κύριο σκοπό να μην είναι απαραίτητη η χρήση ξένων εργαλείων για την δημιουργία ιστοσελίδας και για την υποστήριξη αυτών από φυλλομετρητές ιστοσελίδων .Έτσι η νέα έκδοση κάνει την δημιουργία ιστοσελίδων πολύ πιο εύκολη και πολλές από τις δυνατότητες της έχουν ήδη ενσωματωθεί στους υπάρχοντες φυλλομετρητές για την ευκολότερη υποστήριξη και εμφάνιση διαδικτυακών εφαρμογών και ιστοσελίδων.

3.1.2 Ιστορία της HTML

Το ENQUIRE, το οποίο επινόησε ο φυσικός Τιμ Μπέρνερς Λι το 1980, αποτέλεσε τον μακρινό πρόγονο της HTML και ήταν ένα σύστημα που χρησιμοποιούσαν οι ερευνητές του CERN, και ήταν σύστημα χρήσης και διαμοιρασμού εγγράφων. Αργότερα, ο Μπέρνερς Λι, το 1989, πρότεινε ένα σύστημα βασισμένο στο διαδίκτυο που θα χρησιμοποιούσε υπερκείμενο. Έτσι, στα τέλη του 1990, φτιάχτηκε η προδιαγραφή της HTML και γράφτηκε ο browser και το λογισμικό εξυπηρετητή από τον ίδιο. Επίσης, τον ίδιο χρόνο, ο Μπέρνερς Λι σε συνεργασία με τον Robert Cailliau, μηχανικός συστημάτων πληροφορικής του CERN, προσπάθησαν να βρούνε χρηματοδότηση από το CERN αλλά δυστυχώς το έργο δεν υιοθετήθηκε ποτέ επίσημα. Ο Μπέρνερς Λι αριθμεί «μερικές από τις πολλές χρήσεις του υπερκειμένου» στις προσωπικές του σημειώσεις από το 1990, και αναφέρει πρώτα από όλες μια εγκυκλοπαίδεια.

Ένα έγγραφο με το όνομα Ετικέτες HTML (HTML tags) ήταν η πρώτη δημόσια διαθέσιμη περιγραφή της HTML, καθώς πρωτοαναφέρθηκε, στα τέλη του 1991, ο Μπέρνερς Λι στο Διαδίκτυο. Το έγγραφο αυτό έδινε την περιγραφή των 20 στοιχειών τα οποία αποτελούσαν τον αρχικό και παράλληλα απλό σχεδιασμό της HTML. Όλες οι ετικέτες ήταν έντονα επηρεασμένες από την SGMLguid, μια μορφή δημιουργίας τεκμηρίωσης, φτιαγμένη στο CERN και βασισμένη στην SGML εκτός από την ετικέτα υπερσυνδέσμου. Αξιοσημείωτο αποτελεί οτι δεκατρία από εκείνα τα αρχικά στοιχεία υπάρχουν ακόμα σήμερα στην HTML 4.

Το πρότυπο SGML αποτελεί εκτός από μια απλή μίμηση της τυπογραφίας και αναπαραγωγή μερικών τεχνικών που χρησιμοποιούσαν οι τυπογράφοι και προσθέτει μια γενικευμένη σήμανση βασισμένη σε στοιχεία, τα οποία έχουν την δυνατότητα να μπορούν να εμφωλεύονται το ένα μέσα στο άλλο και να εμφωλεύουν τις ιδιότητες του «αντικειμένου» και ετικέτας. Ακόμα, Ο διαχωρισμός της δομής από το περιεχόμενο αποτελεί βασική λογική που είναι βασισμένο το SGML. Η HTML, με τα CSS ακολούθησαν αυτή την αρχή ώς κύριο γνώμονα στους. Η τεχνική αναφορά ISO TR 9537, Techniques for using SGML (τεχνικές χρήσης της SGML) αποτελεί την έμπνευση πολλών στοιχείων του προτύπου. Τα TYPSET και RUNOFF είχαν αναπτυχθεί στις αρχές της δεκαετίας του 1960 για το λειτουργικό σύστημα CTSS μέσω αυτής της τεχνικής και καλύπτει τα χαρακτηριστικά των πρώιμων γλωσσών μορφοποίησης κειμένου που χρησιμοποιούνταν από αυτά.

Ως μια υλοποίηση του SGML θεωρήθηκε η HTML από τον Μπέρνερς Λι. Με τη δημοσίευση της πρώτης πρότασης για μια προδιαγραφή της HTML, στα μέσα του 1993, επισημοποιήθηκε και ορίστηκε από το Internet Engineering Task Force (IETF). Η πρόταση αυτή περιλάμβανε και έναν ορισμό τύπου εγγράφου (DTD, Document Type Definition) της SGML, ο οποίος όριζε την γραμματική που ήταν βασισμένη. Μετά την πάροδο έξι μηνών, το πρόχειρο (draft) αυτό έληξε αλλά κάτι αξιοσημείωτο περιέχεται το οποίο είναι ότι η αναγνώριση της ετικέτας του NCSA Mosaic για την ενσωμάτωση εικόνων μέσα στο κείμενο. Είναι αξιοσημείωτο γιατί αντικατοπτρίζει την φιλοσοφία του IETF για ενσωμάτωση μέσα στα πρότυπα επιτυχημένων πρωτότυπων. Το «HTML+ (Hypertext Markup Format)», το ανταγωνιστικό πρόχειρο του Dave Raggett, περιείχε κάτι παρόμοιο, πρότεινε την προτυποποίηση μερικών ήδη υλοποιημένων δυνατοτήτων, όπως οι πίνακες και οι φόρμες.

Στις αρχές του 1994,το IETF δημιούργησε την Ομάδα Εργασίας για την HTML ,μετά που τα πρόχειρα HTML και HTML+ είχαν λήξει, η οποία το 1995 ολοκλήρωσε την «HTML 2.0», με την πρόθεση να αποτελέσει την πρώτη προδιαγραφή όπου οι μελλοντικές υλοποιήσεις θα βασίζονταν πάνω σε αυτή. Η HTML 2.0 περιείχε ιδέες από τα πρόχειρα HTML και HTML+. Δημοσιεύτηκε ως RFC 1866. Ο σκοπός της αρίθμησης 2.0 ήταν απλά για να ξεχωρίσει την νέα έκδοση από τα πρόχειρα που προηγήθηκαν.

Η HTML 3.0 προτάθηκε ως πρότυπο από το IETF. Περιείχε πολλές δυνατότητες, όπως τη ροή κειμένου γύρω από εικόνες, την υποστήριξη για πίνακες και την προβολή πολύπλοκων μαθηματικών τύπων. Πολλές δυνατότητες συμπεριλαμβάνονταν στην πρόταση του Raggett για την HTML+. Η πρόταση αυτή έληξε πέντε μήνες αργότερα χωρίς άλλη ενέργεια.

Η ανάπτυξη του Arena browser ως δοκιμαστική πλατφόρμα για την HTML 3 και για τα CSS ξεκίνησε από το W3C αλλά η HTML 3.0 δεν πέτυχε, για διάφορους λόγους. Αρχικά, αφενός το πρόχειρο θεωρήθηκε υπερβολικά μεγάλο καθώς αποτελούταν από 150 σελίδες αφεταίρου, ο ρυθμός ανάπτυξης του browser, καθώς και ο αριθμός των ενδιαφερομένων μερών υπερέβαιναν τις δυνατότητες του IETF. Έτσι οι εταιρείες που διέθεταν browser, όπως η Microsoft και η Netscape εκείνο τον καιρό, αποφάσισαν να εισάγουν τις δικές του επεκτάσεις στο πρόχειρο της HTML 3 καθώς επέλεξαν να υλοποιήσουν διαφορετικά υποσύνολα των δυνατοτήτων του πρόχειρου της . Οι επεκτάσεις αυτές επικεντρωνόταν σε στοιχεία που είχαν να κάνουν με τον έλεγχο εμφάνισης και παρουσίασης των εγγράφων, όσο και αν εναντιωνόταν με την ακαδημαϊκή κοινότητα μηχανικών ότι στοιχεία όπως το χρώμα, το μέγεθος, ο τύπος της γραμματοσειράς και το παρασκήνιο, ήταν οπωσδήποτε έξω από το στόχος μιας γλώσσας και λεπτομέρειες καθώς η

μοναδική πρόθεση ύπαρξης της γλώσσας αυτής ήταν να καθορίσει πώς οργανώνεται ένα έγγραφο. Βέβαια οι εταιρίες ΄και πιο ειδικά η Microsoft σχολιάστηκε για αυτή της την τροπή επέκτασης και πιο συγκεκριμένα ο Dave Raggett, συνεργάτης του W3C για πολλά χρόνια, σχολίαζε ότι «Μέχρι ενός σημείο, η Microsoft έκτισε την επιχειρηματική της δραστηριότητα στον Ιστό επεκτείνοντας τις δυνατότητες της HTML.»

Λόγω σύγκρουσης συμφερόντων που δημιουργήθηκαν, η περαιτέρω ανάπτυξη κάτω από την επίβλεψη του IETF καθυστέρησε. Οι προδιαγραφές της HTML τηρούνται, μαζί με ανάδραση από τους δημιουργούς λογισμικού, από το World Wide Web Consortium (W3C) από το 1996 και μετά. Εν το μεταξύ, το 2000 η HTML έγινε επίσης παγκόσμιο πρότυπο (ISO/IEC 15445:2000). Η τελευταία προδιαγραφή της HTML, η HTML 4.01 δημοσιεύτηκε από το W3C το 1999, και το 2001 δημοσιεύτηκαν επίσης και τα λάθη και οι παραλείψεις της (errata).

3.1.3 Εκδόσεις της XHTML

Η XHTML είναι ξεχωριστή γλώσσα η οποία ως αναδιαμόρφωση της HTML 4.01 με χρήση της XML 1.0 και η οποία συνεχίζει να αναπτύσσεται.

Η XHTML 1.0, δημοσιεύτηκε στις 26 Ιανουαρίου 2000, ως Σύσταση του W3C, μετά αναθεωρήθηκε και επανεκδόθηκε στις 1 Αυγούστου 2002. Προσφέρει τις ίδιες τρεις εκδοχές όπως η HTML 4.0 και 4.01, αναδιαμορφωμένες ως XML, με μικρούς περιορισμούς.

Η XHTML 1.1, δημοσιεύτηκε στις 31 Μαΐου 2001, ως Σύσταση του W3C. Βασίζεται στην XHTML 1.0 Strict, αλλά περιέχει μικρές αλλαγές, μπορεί να παραμετροποιηθεί, μπορεί να αναμορφωθεί χρησιμοποιώντας αρθρώματα της XHTML, τα οποία δημοσιεύτηκαν στις 10 Απριλίου 2001, ως Σύσταση του W3C.

Για την XHTML 2.0 δεν υπάρχει πρότυπο, και αυτή τη στιγμή είναι ένα πρόχειρο έγγραφο και θεωρείται ακόμα έργο σε εξέλιξη. Η XHTML 2.0 δεν είναι συμβατή με την XHTML 1.x και επομένως μπορεί πιο σωστά να χαρακτηριστεί ως μια νέα γλώσσα που είναι εμπνευσμένη από την XHTML παρά ως αναβάθμιση της υπάρχουσας XHTML 1.x.

Τον Οκτώβριο του 2009 το W3C σταμάτησε την εξέλιξη της XHTML με σκοπό να επικεντρωθεί στην ανάπτυξη της HTML5.

3.1.4 Σχετικά με τα πρότυπα

Οι εκδόσεις παίζουν καθοριστικό ρόλο καθώς με το πέρας των εκδόσεων γίνεται και η υιοθέτηση διαφόρων πρωτοτύπων ανάλογα με τις απαιτήσεις που πρέπει να καλυφθούν.

3.1.5 Διαδικασία υιοθέτησης προτύπων

Η έννοια του προτύπου αφορά μία τεχνολογία η οποία πρέπει να προσαρτηθεί στην πράξη και να καθορίσει όλες τις λεπτομέρειες τόσο συμβατότητας με τις υπάρχουσες τεχνολογίες όσο και της λειτουργικότητας που προσφέρει η ίδια. Σε περίπτωση υιοθέτησης ενός προτύπου τότε έχουμε την εγγύηση συνεργασίας προϊόντων από διαφορετικές εταιρίες.

Το Internet Engineering Task Force (IETF) είναι ο φορέας που είναι υπεύθυνος για την δημιουργία πρωτοτύπων σχετικά με τις τεχνολογίες που έχουν σχέση με το internet. Η δημιουργία ενός προχείρου (working draft) ξεκινάει όταν ξεκινήσει η συζήτηση για μία νέα

έκδοση μίας τεχνολογίας και έτσι ανοίγεται ένα νέο πρόχειρο που αναφέρεται σε αυτή. Έγγραφα RFC (Request for Comments) καταθέτονται από εταιρείες, από ειδικούς που ασχολούνται με τον τομέα, καθώς και από άλλους σχετικούς φορείς, αυτά περιέχουν προτάσεις σχετικά με την τεχνολογία που τίθεται υπό συζήτηση. Κάθε έγγραφο RFC έχει έναν μοναδικό αριθμό σαν πρωτόκολλο, όπως για παράδειγμα η έκδοση 2.0 της HTML είναι το έγγραφο RFC 1866. Τα έγγραφα RFC αυτά δεν καταλήγουν όλα να γίνονται πρότυπα, αφού κάποια από αυτά απορρίπτονται ή απορροφούνται από άλλα έγγραφα.

Στην συνέχεια, πραγματοποιείται μια συζήτηση σχετική με το έγγραφο που έχει κατατεθεί για το περιεχόμενο του και πιθανές βελτιώσεις. Ανάλογα με την τεχνολογία που περιέχεται οι φορείς που συμμετέχουν κάθε φορά στη συζήτηση διαφέρουν. Συνήθως για την νέα έκδοση της HTML αναλαμβάνει την συζήτηση το World Wide Web Consortium (W3C). Οποιοσδήποτε μπορεί να καταθέσει μία πρόταση ή να σχολιάσει μία ήδη υπάρχουσα, ενώ όταν οι συμμετέχοντες στη συζήτηση συμφωνήσουν, η πρόταση συνήθως εγκρίνεται ως πρότυπο. Όσο αφορά την έκδοση 5 της γλώσσας αυτής υπήρξε συνεργασία και με το Web Hypertext Application Technology Working Group (WHATWG), το οποίο είναι μία άλλη επιτροπή που ιδρύθηκε από στελέχη εταιρειών που ασχολούνται με την γλώσσα.

3.1.6 HTML5

3.1.6.1 Η «ιστορία» και η τυποποίηση της HTML5

Η HTML5 είναι μια υπό ανάπτυξη γλώσσα σήμανσης για τον Παγκόσμιο Ιστό που όταν ετοιμαστεί θα είναι η επόμενη μεγάλη έκδοση της HTML (Γλώσσα Υπερκειμένου, HyperText Markup Language). Με το όνομα Web Applications 1.0, η ομάδα Web Hypertext Application Technology Working Group (WHATWG) άρχισε δουλειά σε αυτή την έκδοση τον Ιούνιο του 2004. Σε κατάσταση "Last Call" στο WHATWG, ήταν ακόμη το πρότυπο, το Φεβρουάριο του 2010.

Η HTML5 πρόκειται να αντικαταστήσει της HTML 4.01, της XHTML 1.0, και της DOM Level 2 HTML. Η μείωση της ανάγκης για ιδιόκτητα plug-in και πλούσιες διαδικτυακές εφαρμογές (RIA) όπως το Microsoft Silverlight, το Adobe Flash, το Apache Pivot και η Sun JavaFX είναι ο βασικός σκοπός. Η HTML5 εμπεριέχει το πρότυπο Web Forms 2.0 που είναι επίσης της WHATWG. Οι ιδέες πίσω από την HTML5 άρχισαν να εμφανίζονται αρχικά το 2004 από την ομάδα WHATWG.

Το πρότυπο HTML5 υιοθετήθηκε ως αρχικό βήμα για τις εργασίες της νέας ομάδας εργασίας HTML του W3C το 2007. Αυτή η ομάδα εργασίας δημοσίευσε το Πρώτο Δημόσιο Working Draft του προτύπου στις 22 Ιανουαρίου 2008. Το πρότυπο είναι ακόμη υπό ανάπτυξη, μέρη της HTML5 θα τελειώσουν και θα υποστηριχτούν από περιηγητές πριν το όλο πρότυπο φτάσει στη τελική κατάσταση Recommendation και αναμένεται να παραμείνει έτσι για πολλά χρόνια.

3.1.6.2 Εύρος του προτύπου και δύναμη της HTML5

Η HTML5 περιλαμβάνει πολλά τμήματα με τα οποία θα μπορούν να δημιουργηθούν ανεξάρτητες και αρκετά πολύπλοκες εφαρμογές που απαιτούν οι συνθήκες εξέλιξης της παρούσας εποχής σε αντίθεση με τις προηγούμενες εκδόσεις της HTML οι οποίες ως στόχο είχαν την δημιουργία μίας γλώσσας κατάλληλης για την περιγραφή της εμφάνισης των σελίδων και την βασική αλληλεπίδρασή τους με τον χρήστη.

Η HTML5, εξαιτίας του εύρους των θεμάτων που μπορεί να καλύψουν οι προτάσεις που αναπτύσσονται μπορεί να χαρακτηριστεί και σαν μία πλήρης γλώσσα προγραμματισμού. Η δύναμη της γλώσσας πλέον HTML, οφείλεται στο ότι παρέχει πλέον δυνατότητες που θα μπορεί να αντικαταστήσει ένα πλήθος τρίτων τεχνολογιών που σήμερα χρησιμοποιούνται στις υπάρχουσες ιστοσελίδες χωρίς να υπάρξει απώλεια δυνατότητων της ιστοσελίδας είτε τεχνολογίας. Επιπλέον πολλές από τις αλλαγές που έχουν προταθεί έχουν σαν βασικό μέλημα να διορθώσουν κενά της γλώσσας και άμεσες ελλείψεις.

Οι αλλαγές προφανώς επηρεάζουν και προσθέτουν πολλά καινούργια στοιχεία και στη Javascript και στο CSS εκτός από την γλώσσα HTML. Το πρότυπο διατυπώνεται με μεγάλη ακρίβεια και λεπτομέρεια ώστε να είναι ξεκάθαρος ο τρόπος υλοποίησής τους και να μην επαναληφθούν προβλήματα που παρουσιάστηκαν στο παρελθόν.

3.1.7 Περιορισμοί της HTML

Η τυποποίηση της HTML αναγκάστηκε να γίνεται παράλληλα με την χρήση της και την υλοποίηση της είτε μέχρι τελικής μορφής είτε τμημάτων της καθώς εάν ακολουθούνταν το τυπικό πρωτόκολλο υιοθέτησης προτύπων θα έπρεπε να ολοκληρωθεί το 2020 όμως είναι πράγμα αναπόφευκτο. Οι εταιρείες και οι οργανισμοί αναλαμβάνουν την ευθύνη παράλληλης στήριξης και χρήσης και υλοποίησης καθώς φυσικά χρειάζεται να γίνει και κάλυψη δικών τους αναγκών.

Όπως παραδείγματος χάριν συμβαίνει με τους browsers όπου κάθε διαφορετική έκδοση του ίδιου browser μπορεί να υποστηρίζει διαφορετικά στοιχεία σε διαφορετικό βαθμό

ή να τα υλοποιεί με διαφορετικό τρόπο. Οι περισσότεροι σύγχρονοι browser αναβαθμίζονται συνεχώς, υιοθετώντας όλο και περισσότερα χαρακτηριστικά με κάθε νέα έκδοση.

Τα προβλήματα που υπάρχουν σήμερα για την κατασκευή σελίδων με τις προηγούμενες εκδόσεις της HTML χωρίζονται σε δύο βασικές κατηγορίες. Οι κατηγορίες αυτές είναι η πληθώρα των τρίτων τεχνολογιών και η δεύτερη οι αποκλίσεις από το πρότυπο της.

Η αιτία δημιουργίας και ύπαρξης της πρώτης είναι η έλλειψη χαρακτηριστικών από την HTML, όπως για παράδειγμα η έλλειψη την δυσκολία εμφάνισης αρκετά πολύπλοκων γραφικών ή animations.

Στην προσπάθεια εξάλειψης αυτού του κενού που υπήρχε δημιουργήθηκε μια πληθώρα τρίτων τεχνολογιών με σκοπό την ενσωμάτωσή τους σε μία σελίδα. Η πιο γνωστή τέτοια τεχνολογία είναι το Flash της εταιρείας Adobe, το οποίο χρησιμοποιείται κατά κύριο λόγο μεταξύ άλλων για την δημιουργία animations, παιχνιδιών και την προβολή βίντεο. Εκτός από το flash, η Silverlight από την εταιρεία Microsoft, το JavaFX που είναι βασισμένο στη γλώσσα Java έχουν επιπροσθεθεί σε σελίδες μαζί και άλλες μικρότερες εφαρμογές.

Σε όλους τους χρήστες είναι γνωστό ότι για την εφικτή λειτουργία των τεχνολογιών αυτών είναι απαραίτητο να υπάρχει εγκαταστημένο πρόγραμμα που να ρυθμίζει να υποστηρίζει και αναγνωρίζει την τεχνολογία αυτή εκτός από το πρόγραμμα περιήγησης που πρέπει να είναι συνεργάσιμο με την τεχνολογία αυτή.

Αυτό είναι προβληματικό από πολλές απόψεις καθώς αυτά τα προγράμματα αναπτύσσονται από τρίτες εταιρείες και δεν προσφέρουν εγγυήσεις σχετικά με την διαθεσιμότητα τους σε όλες τις πλατφόρμες, όπως για παράδειγμα το Silverlight το οποίο ποτέ δεν κυκλοφόρησε για λειτουργικά συστήματα Linux. Από την άλλη οπτική και διαθέσιμα να είναι για την πλατφόρμα που χρησιμοποιεί ο χρήστης, ο χρήστης είναι αναγκασμένος να εγκαταστήσει το πρόγραμμα και σε περίπτωση όπου δεν κατέχει τις απαραίτητες γνώσεις πάλι καταλήγουμε στο ίδιο παρονομαστή. Επιπλέον, οι τεχνολογίες αυτές μπορεί να δημιουργούν προβλήματα τόσο κενών ασφάλειας όσο και απαιτήσεις σε επεξεργαστική ισχύ και μνήμη και ειδικά όταν πρόκειται για φορητές συσκευές αυτό προκαλεί μεγάλη κατανάλωση ενέργειας, όπως για παράδειγμα το πρόγραμμα Flash Player που είναι υπεύθυνο για την αναπαραγωγή αντικειμένων τύπου Flash.

Σε περίπτωση που το δούμε από την οπτική γωνία του προγραμματιστή που είναι υπεύθυνος για την κατασκευή ιστοσελίδων που εμπεριέχουν γραφικά και animation είτε videos και ο χρήστης που θα επισκεφθεί την ιστοσελίδα που δημιούργησε να μην μπορεί να δει το περιεχόμενο της ,αφού δεν υπάρχει κάποια εγγύηση πως θα έχει εγκαταστημένο το αντίστοιχο πρόσθετο, αυτόματα περιορίζει το κοινό της σελίδας του ή δεν χρησιμοποιεί την συγκεκριμένη τεχνολογία, άρα πάλι στην αβεβαιότητα της στιγμής.

Όσο αφορά την δεύτερη και σημαντική κατηγορία είναι από την στιγμή που δεν υπάρχει σταθεροποίηση προτύπου έχουμε και αποκλίσεις υλοποίησης του ανάλογα με το browser και έτσι πολλές σελίδες υποστηρίζονται με διαφορετικό τρόπο και εμφανίζονται ανάλογα με τον browser και όχι με την υλοποίηση της ιστοσελίδας που προήλθε από τον προγραμματιστή είτε λόγο διαφορετικής ερμηνείας των προδιαγραφών είτε λόγω εσωτερικών προβλημάτων (bugs).

Αυτό έχει σαν αποτέλεσμα ο κώδικας που λειτουργεί σωστά σε ένα πρόγραμμα περιήγησης να μην εμφανίζεται σε κάποιο άλλο. Έτσι ο προγραμματιστής πρέπει να κάνει δοκιμές σε όλους τους πιθανούς browsers τον κώδικα μίας σελίδας και να παρεμβαίνει όπου δεν λειτουργούν σωστά. Κάτι τέτοιο όμως αυξάνει κατακόρυφα το κόστος και τον χρόνο ανάπτυξης. Σαν αποτέλεσμα πολλές σελίδες γράφονται στοχεύοντας μόνο τις πιο πρόσφατες εκδόσεις των browsers και δεν λειτουργούν σωστά στις παλιότερες, και συνήθως παροτρύνουν τον χρήστη να αναβαθμίσει το πρόγραμμα περιήγησης, πράγμα το οποίο για άλλη μία φορά καταλήγουμε στο ότι ο χρήστης πρέπει κάτι να κάνει που ίσως και να αποτελεί μόνο στο άκουσμα του κάτι σαν σπαζοκεφαλιά.

Δυστυχώς είναι λάθη τα οποία επαναλαμβάνονται καθώς χαρακτηριστικό παράδειγμα κακής χρήσης των προτύπων παρουσιάστηκε κατά την έκδοση της HTML 3.0, η οποία ποτέ δεν επισημοποιήθηκε και τότε οι μεγάλοι κατασκευαστές browsers της εποχής ήταν οι εταιρείες Microsoft και Netscape, οι οποίες υιοθέτησαν διαφορετικά τμήματα της προδιαγραφής με αποτέλεσμα αρκετά πράγματα που λειτουργούσαν στα προϊόντα της μίας να μην λειτουργούν στα προϊόντα της άλλης. Προς αποφυγήν κάτι τέτοιου κατά την διαδικασία ανάπτυξης της HTML5 δίνεται αρκετά μεγάλο βάρος στην ακριβή διατύπωση της προδιαγραφής και στην παροχή διευκρινήσεων για όλες τις λεπτομέρειες που μπορεί να παρερμηνευτούν σε περίπτωση επέκτασης πρωτοτύπου και βαρύτητας γνώσεων.

3.1.8 Λειτουργία

Συνήθως οι ετικέτες HTML λειτουργούν ανά ζεύγη όπως για παράδειγμα <h1>Επικεφαλίδα1</h1>, με την πρώτη ετικέτα να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης. Ανάμεσα στις ετικέτες τοποθετούνται αρχεία όπως κείμενα, βίντεο, εικόνες κτλ. όπου ο browser ερμηνεύει το περιεχόμενο, χωρίς όμως να εμφανίζει τις ετικέτες HTML. Οι σελίδες που γράφονται σε αυτή τη μορφή είναι κείμενα και μπορούν να διαβαστούν από οποιονδήποτε χρησιμοποιεί απλό κείμενο. Δεν επηρεάζονται οι εντολές από το αν έχουν γραφτεί με κεφαλαία ή πεζά γράμματα. Τα αρχεία αυτά αποθηκεύονται ως *.html ή *.htm**.**

3.1.9 Μερικές συντάξεις και εντολές των στοιχείων της HTML

- Περιγραφή της ιστοσελίδας:

```
<html>

…

</html>
```

- Κεφαλίδα εγγράφου HTML:

```
<head>
...
</head>
```

- Καθορισμός του τίτλου της ιστοσελίδας, ο οποίος εμφανίζεται στο πάνω μέρος του browser:

```
<title>
 The title
 </title>
```

- Κύριο μέρος του αρχείου:

```
<body>
 …
</body>
```

- Επικεφαλίδες:

```
<h1>
Επικεφαλίδα1
</h1>
<h2>
Επικεφαλίδα2
</h2>
<h3>
Επικεφαλίδα3
</h3>
```

- Πίνακες δεδομένων:

```
<table>

…
```

Οι γραμμές του πίνακα καθορίζονται από την ετικέτα:

```
 <tr>

 …

</tr>
```

Οι στήλες του πίνακα καθορίζονται από την ετικέτα:

```
  <th>

 …
```

Τα δεδομένα στηλών καθορίζονται από την ετικέτα:

```
<td>

 …

</td>

</th>

</table>
```

- Πλαίσια (frames):

  <**frame**>
  …

  </**frame**>

- Μη – αριθμημένη λίστα:
  <**ul**>

  <**li**> Πρώτο στοιχείο

  <**li**> Δεύτερο στοιχείο

  </**ul**>

- Αριθμημένη λίστα:

  <**ol**>
  <**li**> Πρώτο στοιχείο

  <**li**> Δεύτερο στοιχείο

  </**ol**>

- Φόρμα:

  <**form**>

  …

  Σε περίπτωση εισαγωγής κειμένου:
  <**input** name="**..**" ...>

  Σε περίπτωση αποστολής δεδομένων συμπλήρωσης φόρμας μέσω κουμπιού:
  <**input** type="**submit**" value="**....**">
  </**form**>

3.1.10 Τα Πλεονεκτήματα της HTML :

- Εύκολο στη χρήση.
- Υποστηρίζεται από κάθε πρόγραμμα περιήγησης.
- Είναι δωρεάν. Δε χρειάζεται η αγορά κάποιου λογισμικού.
- Είναι εύκολο στη μάθηση και στη δημιουργία κώδικα.
- Χρησιμοποιείται ευρέως.

3.2 CSS

Η CSS (Cascading Style Sheets) είναι μια γλώσσα φύλλων στυλ που χρησιμοποιείται για την περιγραφή της παρουσίασης ενός εγγράφου που είναι γραμμένο σε μια γλώσσα σήμανσης. Συνήθως χρησιμοποιείται για τον έλεγχο της εμφάνισης εγγράφων που έχουν γραφτεί στις γλώσσες HTML και XHMTL. Είναι ουσιαστικά σαν να έχει τον ρόλο του ζωγράφου που αποφασίζει με τι χρώματα, τι γράμματα, τι εμφάνιση θέλει να έχει στο πορτραίτο του και σε περίπτωση που θέλει να τα αλλάξει να μπορεί δημιουργήσει στον ίδιο καμβά το πορτραίτο του αλλαγμένο απλά πειράζοντας τους παραμέτρους που όρισε.

Η CSS έχει σχεδιαστεί για να επιτρέπει κυρίως τον διαχωρισμό του περιεχομένου ενός εγγράφου από την παρουσίαση του εγγράφου και για να προσφέρει περισσότερες δυνατότητες για μορφοποίηση από την HTML, συμπεριλαμβανομένων στοιχείων, όπως η διάταξη, τα χρώματα και τις γραμματοσειρές. Η πιο κοινή της εφαρμογή είναι σε ιστοσελίδες γραμμένες σε HTML και XHTML, αλλά η CSS μπορεί επίσης να εφαρμοστεί σε οποιοδήποτε είδος εγγράφου XML, συμπεριλαμβανομένου του απλού XML, SVG και XUL.

Η CSS δημιουργεί έναν δομημένο τρόπο παρακολούθησης του εγγράφου και των τροποποιήσεων της παρουσίασης του. Η χρήση της CSS και ο διαχωρισμός που προκύπτει του περιεχομένου από τη μορφοποίηση, παρέχει μεγαλύτερη ευελιξία και έλεγχο στην εξειδίκευση των χαρακτηριστικών παρουσίασης, βελτιώνει την προσβασιμότητα του περιεχομένου, προσφέρει τη δυνατότητα πολλές σελίδες να μοιράζονται την ίδια ακριβώς μορφοποίηση και μειώνει την πολυπλοκότητα και τις επαναλήψεις στον κώδικα που αφορά τη δόμηση του εγγράφου – ιστοσελίδας.

Ένα φύλλο στυλ αποτελείται από μια λίστα κανόνων. Κάθε κανόνας ή σετ κανόνων αποτελείται από έναν ή περισσότερους επιλογείς και ένα μπλοκ δήλωσης. Οι επιλογείς χρησιμοποιούνται για να δηλώσουν σε ποιο μέρος της σήμανσης εφαρμόζεται το στυλ. Το τμήμα δήλωσης αποτελείται από μια λίστα δηλώσεων οι οποίες περικλείονται σε αγκύλες. Κάθε δήλωση αποτελείται από μια ιδιότητα, άνω και κάτω τελεία (:) και μια τιμή. Εάν υπάρχουν πολλές δηλώσεις σε ένα μπλοκ τότε εισάγεται ένα ερωτηματικό για το διαχωρισμό κάθε δήλωσης.

Η CSS επιτρέπει στην ίδια σελίδα σήμανσης (πχ μια HTML σελίδα) σε μια ιστοσελίδα να παρουσιαστεί με διαφορετικό τρόπο ανάλογα με το μέγεθος της οθόνης ή τη συσκευή στην οποία προβάλλεται να παρουσιάζεται με διαφορετικό στυλ ανάλογα με τη μέθοδο επεξεργασίας, όπως σε οθόνη υπολογιστή, σε συσκευές αφής ή σε έντυπη μορφή. Επιτρέπει επίσης.

Ο δημιουργός ενός εγγράφου σήμανσης συνήθως συνδέει το έγγραφο με ένα αρχείο CSS, όμως όσοι το διαβάζουν μπορούν να χρησιμοποιήσουν ένα διαφορετικό φύλλο στυλ και να παρακάμψουν αυτό που έχει καθορίσει ο δημιουργός.

Τα φύλλα στυλ ονομάστηκαν επικαλυπτόμενα ή αλλιώς διαδοχικά (cascading style sheets) γιατί η CSS έχει ορίσει ένα σύστημα προτεραιότητας για να καθορίζει ποιος κανόνας στυλ πρέπει να εφαρμοστεί αν περισσότεροι από ένας κανόνας ταιριάζουν με ένα συγκεκριμένο στοιχείο. Προτεραιότητες ή αλλιώς βάρη υπολογίζονται και εκχωρούνται στους κανόνες έτσι ώστε να υπάρχει μια σειρά και να μπορεί να προβλεφθεί η εμφάνιση της σελίδας.

3.2.1 Μειονεκτήματα

Ένα βασικό μειονέκτημα αποτελεί ότι οι επιλογείς δεν είναι σε θέση να καταλάβουν την ιεραρχία που προσφέρει η γλώσσα καθώς δεν υπάρχει τρόπος για να επιλεχθεί ένα γονέα ή

πρόγονο ενός στοιχείου που πληροί ορισμένα κριτήρια. Έχουν απορριφθεί διάφορες προτάσεις εξαιτίας ότι προκαλούν ζητήματα απόδοσης. Πιο εξειδικευμένοι επιλογείς χρησιμοποιούνται από την γλώσσα XPath για να υποστηρίζουν πιο πολύπλοκα έγγραφα.

Υπάρχουν κάθετοι περιορισμοί ελέγχου στην τοποθέτηση στοιχείων καθώς είναι δύσκολο να διαχειριστούν είτε αδύνατον ορισμένες φορές ενώ αντιθέτως η οριζόντια τοποθέτηση είναι διαχωρίσιμη. Απλές εργασίες όπως κεντράρισμα είναι ορισμένες φορές ανώφελο να ορισθούν μέσα από την γλώσσα για το έγγραφο.

Δυστυχώς εξαιτίας ότι δεν έχουν ακόμα οριστεί από το πρότυπο εκφράσεις δεν μπορούν να υποστηριχτούν και απουσιάζουν εντελώς ακόμα και απλές εκφράσεις όπως το περιθώριο κέρδους. Ενέργειες που είχαν γίνει σχετικά με αυτό σταμάτησαν να υποστηρίζονται γιατί πρέπει να υπάρχει συνάφεια των πρωτοτύπων με τους browsers ώστε να μην υπάρχουν προβλήματα επίδοσης και ασφάλειας.

Η έλλειψη της δήλωσης στήλη είναι ένα άλλο θέμα που ταλαιπωρεί τον σχεδιασμό και αν και είναι δυνατόν στη σημερινή CSS 3 (με τη χρήση της μονάδας στήλης-count), σχεδιαγράμματα με πολλαπλές στήλες μπορεί να είναι περίπλοκο να εφαρμοστεί στην CSS 2.1. Στην CSS 2.1, η διαδικασία γίνεται συχνά με τη χρήση πλωτών στοιχείων, τα οποία συχνά αποδίδονται με διαφορετικό τρόπο από διαφορετικούς browsers, διαφορετικά σχήματα ανάλογα με την οθόνη του υπολογιστή και τις αναλογίες της.

Δεν μπορεί να δηλωθεί καινούργιο πεδίο, ανεξάρτητα από τη θέση του.

Δεν γίνεται υποστήριξη και έλεγχος ψευδό- δυναμικής συμπεριφοράς. Η CSS υλοποιεί ψευδό-κλάσεις που επιτρέπουν ως ένα βαθμό τα σχόλια των χρηστών με όρους εφαρμογής εναλλακτικών μορφών. Όπως για παράδειγμα, μια CSS ψευδό-class, «: hover», είναι δυναμική (ισοδύναμη με το «onmouseover» της JavaScript) και έχει τη δυνατότητα κατάχρησης- υπερκάλυψης (π.χ., στην εφαρμογή αναδυόμενα παράθυρα), αλλά η CSS δεν έχει καμία δυνατότητα για έναν πελάτη για να το απενεργοποιήσει. Δεν υπάρχει δυνατότητα απενεργοποίησης ή να περιορίσει τα αποτελέσματά της.

Δεν επιτρέπεται να γίνει ονομασία κανόνων και δεν υφίσταται να συμπεριληφθεί κάποιο στυλ από έναν κανόνα σε έναν άλλο κανόνα. Η επανάληψη αρκετών κανόνων σε style CSS πρέπει να πραγματοποιηθεί ώστε να επιτευχθεί το επιθυμητό αποτέλεσμα, προκαλώντας πρόσθετη συντήρηση και απαίτηση ενός ενδελεχή έλεγχού.

3.2.2 Πλεονεκτήματα

Αρχικά, επιτυγχάνεται ο διαχωρισμός του περιεχομένου από την παρουσίαση. Η CSS διευκολύνει τη δημοσίευση του περιεχομένου σε πολλαπλές μορφές παρουσίασης με βάση τις ονομαστικές παραμέτρους. Μερικές από τις ονομαστικές παραμέτρους αποτελούν οι προτιμήσεις του χρήστη, τα διαφορετικά προγράμματα περιήγησης στο Web, ο τύπο της συσκευής που χρησιμοποιείται για να δει το περιεχόμενο είτε μέσω ενός επιτραπέζιου υπολογιστή είτε ενός κινητού), η γεωγραφική θέση του χρήστη και πολλές άλλες μεταβλητές.

Προσφέρει συνοχή αφενός με την χρήση κύριων άρθρων όπου ο διαχωρισμός της παρουσίασης, του περιεχομένου και των style sheets για τον σχεδιασμό ιστοσελίδων είναι ευδιάκριτος και αφετέρου όταν η CSS χρησιμοποιεί αποτελεσματικά, την έννοια της κληρονομικότητας και υπερχείλισης. Ένα παγκόσμιο φύλλο στυλ μπορεί να χρησιμοποιηθεί για να επηρεάσει τα στοιχεία style σε όλη την εμβέλεια. Εάν σύμφωνα με την περίσταση, προκύπτει ότι το style sheet των στοιχείων θα πρέπει να αλλάξει ή να προσαρμοστεί, κάτι τέτοιο πριν την CSS, ήταν πιο δύσκολο, δαπανηρό και χρονοβόρο, τώρα όμως εύκολο, γρήγορο και ευέλικτο.

Παρέχει την δυνατότητα εύκολης και γρήγορης επεξεργασίας και αλλαγής παρουσίασης της ιστοσελίδας. Με μια απλή αλλαγή σε μία γραμμή, ένα διαφορετικό stylesheet μπορεί να χρησιμοποιηθεί για την ίδια σελίδα. Αυτό έχει πλεονεκτήματα σχετικά με την προσβασιμότητα αλλά παρέχει και τη δυνατότητα να προσαρμόζει μια σελίδα ή ένα ολόκληρο site σε διαφορετικές συσκευές.

Το βασικότερο όλων όμως είναι η προσβασιμότητα καθώς χωρίς το CSS, οι σχεδιαστές ιστοσελίδων έπρεπε τυπικά να ορίζουν τις σελίδες τους με τεχνικές όπως πίνακες HTML που εμπόδιζαν την προσβασιμότητα σε χρήστες με προβλήματα όρασης χρήστες, πράγμα το οποίο είναι τώρα εφικτό.

3.2.3 Προγραμματιστικός Συνδυασμός HTML και CSS

Ο συνδυασμός HTML και CSS**,** αν και γίνεται είναι γενικά πολύ κακή πρακτική. Γενικός στόχος είναι να ξεχωρίζουμε το περιεχόμενο από την εμφάνιση, ώστε οι αλλαγές στο ένα να μην επηρεάζουν το άλλο. Για αυτό το λόγο τυπικά οι οδηγίες CSS δίνονται σε ένα ξεχωριστό αρχείο CSS, το οποίο συμπεριλαμβάνουμε στη σελίδα με την κατάλληλη εντολή. Για να εισάγουμε ένα αρχείο CSS προσθέτουμε μέσα στο στοιχείο head μία γραμμή που καθορίζει το αρχείο που θέλουμε να χρησιμοποιηθεί.

```
<!doctype html>

<html>

<head>

        <meta charset=”utf-8”>

        <title> Εφαρμογή Πτυχιακής: Φαρμακείο </title>

        <link href= “style.css” rel = “stylesheet” type=”text/css”>
```

Ο παραπάνω κώδικας αποτελεί την αρχή μίας σελίδας HTML που χρησιμοποιεί ένα αρχείο CSS.

Το παραπάνω παράδειγμα δείχνει ξεκάθαρα τον ρόλο του CSS σε αντιδιαστολή με την HTML. Το αρχείο style.css που εισήγαμε έχει μέγεθος περίπου διακόσιες γραμμές και αλλάζει ριζικά τον τρόπο εμφάνισης της σελίδας. Όμως όλες οι πληροφορίες για την εμφάνιση είναι περιορισμένες σε αυτό, και για την χρήση του δεν χρειάζεται καμία επιπλέον αλλαγή στην HTML. Ο κώδικας της σελίδας παραμένει ξεκάθαρος και διαβάζοντάς τον είναι εύκολο να καταλάβουμε το περιεχόμενό της και τον ρόλο του κάθε στοιχείου.

3.3 JavaScript

Η δομή μιας σελίδας περιγράφεται από τη γλώσσα HTML και η εμφάνισή της από τη γλώσσα CSS. Είναι φανερό πως και οι δύο γλώσσες είναι στατικές, δηλαδή απλά δίνουν οδηγίες για το τι και πώς θα φαίνεται, και όχι για το πώς θα λειτουργεί, εκεί έρχεται ο ρόλος της Javascript.

Καθώς οι σελίδες HTML παρουσιάζουν περιεχόμενο στον χρήστη, αρκετές φορές αυτό αρκεί. Όμως υπάρχουν περιπτώσεις που θέλουμε να κάνουμε μία σελίδα να αντιδράει στις επιλογές του χρήστη με διάφορους τρόπους. Κάτι τέτοιο δεν είναι απαραίτητο, όμως μπορεί να κάνει την σελίδα πολύ πιο εύκολη στη χρήση άλλες πάλι φορές είναι απαραίτητο καθώς πρέπει να γίνουν και ενέργειες που αφορούν και λειτουργικά κομμάτια όπως σύνδεση σε ένα server . Ο ρόλος της γλώσσας Javascript εμπίπτει ακριβώς εκεί.

Η JavaScript είναι μια δυναμική γλώσσα προγραμματισμού των ηλεκτρονικών υπολογιστών. Χρησιμοποιείται πιο συχνά ως μέρος των web browsers, των οποίων οι υλοποιήσεις επιτρέπουν client-side scripts για να αλληλοεπιδράσει με το χρήστη, ελέγχουν το πρόγραμμα περιήγησης, να επικοινωνούν ασύγχρονα, και μεταβάλλει το περιεχόμενο του εγγράφου που εμφανίζεται. Επίσης, χρησιμοποιείται για τον προγραμματισμό του δικτύου server-side με πλαίσια όπως Node.js, την ανάπτυξη παιχνιδιών και τη δημιουργία των desktop και mobile εφαρμογών.

Στην κατανόηση της γλώσσας αυτής θα εξηγήσουμε τον σημαντικό ρόλο που παίζει μέσω ενός παραδείγματος που σίγουρα όλοι μας έχουμε αντιμετωπίσει ως χρήστες του διαδικτύου. Το παράδειγμα μας είναι η ορθή συμπλήρωση φόρμας και αυτόματος έλεγχος των πεδίων συμπλήρωσης σε κάποια ιστοσελίδα. Ας το κοιτάξουμε αναλυτικά.

Ένα τυπικό σενάριο είναι μία φόρμα εγγραφής νέου χρήστη. Σε αυτή ο χρήστης συμπληρώνει τα στοιχεία του και υποβάλει την φόρμα, δηλαδή στέλνει αυτά τα στοιχεία στον server. Πριν γίνει η αποθήκευση των στοιχείων του νέου χρήστη χρειάζεται να γίνει έλεγχος των στοιχείων που στάλθηκαν, για παράδειγμα αν έχουν δοθεί έγκυρα στοιχεία και αν έχουν συμπληρωθεί όλα τα υποχρεωτικά πεδία, πχ σε ένα πεδίο που ζητάει το email του χρήστη εάν είναι σωστά γραμμένο. Αν τα στοιχεία δεν είναι έγκυρα τότε δεν μπορεί να γίνει αποθήκευση, οπότε είναι απαραίτητο να εμφανιστεί πάλι η φόρμα και να εμφανιστεί το

κατάλληλο μήνυμα που να ειδοποιεί τον χρήστη για τα λάθη. Παρόλο που η παραπάνω προσέγγιση λειτουργεί, είναι κουραστική και δυσνόητη για τον χρήστη. Αφού αυτός συμπληρώνει την φόρμα εγγραφής περιμένει να δει το αποτέλεσμά της ενέργειάς του. Το να εμφανίζεται πάλι η ίδια φόρμα μπορεί να τον αποπροσανατολίσει, ειδικά αν δεν έχει εμπειρία, ενώ χρειάζεται να καταβάλει προσπάθεια για καταλάβει τι λάθος έκανε και ότι πρέπει να την συμπληρώσει ξανά.

Στο παραπάνω παράδειγμα η φόρμα θα μπορούσε να ελέγχεται με JavaScript πριν την υποβολή της. Με αυτόν τον τρόπο τα λάθη θα εμφανίζονταν μόλις ο χρήστης πατούσε το κουμπί υποβολής, χωρίς να ξαναφορτωθεί η σελίδα. Με αυτόν τον τρόπο μπορεί να καταλάβει αμέσως τι έχει συμβεί, αφού το λάθος εμφανίστηκε σαν αντίδραση στο πάτημα του κουμπιού, κάνοντας την εφαρμογή πιο φιλική, είτε θα μπορούσε με το που συμπληρώνει το κάθε κουτάκι να το τσεκάρει και να μην το κοκκινίζει όπως έχουμε δει σε πολλές σελίδες.

3.3.1 Περιορισμοί

Δυστυχώς θα πρέπει να παραθέσουμε κάποιους περιορισμούς που μας ορίζει η γλώσσα αυτή που δυστυχώς μας προκαλεί την εντύπωση και όχι μόνο, μιας γλώσσας περίπλοκης και δύσκολης συγκριτικά με τις συμβατικές γλώσσες που γνωρίζουμε αν και παρέχει παντοδύναμα πράγματα στην τροποποίηση περιεχομένου μιας σελίδας.

Ο πρώτος είναι πως για την εκτέλεση πολύπλοκων λειτουργιών ο κώδικας JavaScript που απαιτείται είναι αρκετά πιο μπερδεμένος σε σχέση με κάποιες παραδοσιακές γλώσσες προγραμματισμού.

Επιπλέον υπάρχουν αρκετές μικρές διαφοροποιήσεις στον ακριβή τρόπο λειτουργίας της JavaScript ανάμεσα σε διαφορετικούς browsers, κάτι το οποίο κάνει απαραίτητο τον έλεγχο της σελίδας σε κάθε έναν από αυτούς ώστε να τσεκάρουμε την συμβατότητα υποστήριξης της.

Ο βασικότερος περιορισμός της JavaScript είναι η αδυναμία άμεσης επικοινωνίας με βάση δεδομένων. Αν και είναι δυνατή η επικοινωνία με τον server μέσω AJAX, δεν γίνεται με κανέναν τρόπο να επικοινωνήσει με την βάση δεδομένων, εκτός και αν παρεμβάλλεται μία σελίδα γραμμένη σε κάποια γλώσσα που εκτελείται στον server.

Έτσι, αυτό που θα πρέπει να θυμόμαστε είναι ότι ο ρόλος της είναι βοηθητικός, στο να προσφέρει κάποια επιπλέον λειτουργικότητα, αλλά η δημιουργία της βασικής σελίδας γίνεται στον server.

3.3.2 JavaScript Frameworks

Όπως αναφέραμε προηγουμένως κάθε browser έχει διαφορετικές υλοποιήσεις της JavaScript, οι οποίες μπορούν να διαφέρουν μεταξύ τους. Για να εξασφαλιστεί η απρόσκοπτή λειτουργία μίας σελίδας που χρησιμοποιεί JavaScript χρειάζεται να ελεγχθεί σε όλους τους πιθανούς browsers που θα την τρέξουν, κάτι το οποίο είναι εξαιρετικά επίπονο και πολλές φορές πρακτικά αδύνατο, καθώς δεν γίνεται να καλυφθούν όλες οι εκδόσεις του κάθε browser, ιδιαίτερα αν λάβουμε υπόψη μας και τους διαφορετικούς browsers που υπάρχουν σε ένα πλήθος κινητών τηλεφώνων και tablets.

Αν και υπάρχουν μικρές ασυμβατότητες στην υλοποίηση και τον τρόπο εμφάνισης των στοιχείων HTML και των κανόνων CSS, το πρόβλημα είναι μεγαλύτερο στην JavaScript γιατί πχ, μία διαφορά σε έναν κανόνα CSS συνήθως οδηγεί σε διαφορετική απεικόνιση ενός στοιχείου, ενώ ένα κομμάτι κώδικα JavaScript μπορεί να πάψει να λειτουργεί. Γι' αυτόν τον λόγο σπάνια γίνεται άμεσα χρήση εντολών μόνο της JavaScript που περιέχεται σε έναν browser, αλλά συνήθως χρησιμοποιείται ένα επιπλέον Framework.

Ένα framework είναι μία βιβλιοθήκη που περιέχει κάποια ήδη ορισμένα σύμβολα (μεταβλητές, κλάσης και συναρτήσεις) τα οποία παρέχουν κάποιες λειτουργίες, οι οποίες μπορεί να αντικαθιστούν τις λειτουργίες που περιέχει η JavaScript ή και να προσφέρουν επιπλέον δυνατότητες. Είναι και αυτό γραμμένο σε JavaScript, όμως με τέτοιο τρόπο ώστε να εγγυάται την συμβατότητα με αρκετούς browsers και πολλές φορές να προσφέρει έτοιμες κάποιες ενέργειες, συνήθως με την μορφή έτοιμων συναρτήσεων, οι οποίες αν και μπορούν να υλοποιηθούν με JavaScript είναι είτε τετριμμένες είτε αρκετά δύσκολες στην υλοποίηση.

Ένα βασικό framework μπορεί να προσφέρει κάποιες έτοιμες δυνατότητες, όμως αρκετά από αυτά είναι αρκετά πιο εξελιγμένα και έχουν συγκεκριμένους κανόνες χρήσης. Η εκμάθηση ενός framework χρειάζεται κάποια προσπάθεια, όμως αποδίδει γιατί στην πορεία μειώνει τον κόπο για την υλοποίηση των επιμέρους λειτουργιών που χρησιμοποιούνται και καθιστά περιττό τον έλεγχο σε κάθε browser.

Το βασικό μειονέκτημα που έχει η χρήση ενός framework, πέρα από τον κόπο που χρειάζεται για την εκμάθησή του, είναι πως τυπικά για την εκτέλεση μίας λειτουργίας απαιτείται η χρήση πιο πολύπλοκου κώδικα, τον οποίον ο προγραμματιστής δεν βλέπει. Αυτό μπορεί να έχει επιπτώσεις στην ταχύτητα εκτέλεσης των λειτουργιών σε μία σελίδα.

Η πτώση των επιδόσεων ήταν παλιότερα μεγάλο πρόβλημα, όμως με το πέρασμα του χρόνου τα περισσότερα frameworks έχουν βελτιστοποιηθεί αρκετά σε αυτόν τον τομέα, και σε συνδυασμό με τη συνεχή αύξηση των επιδόσεων των υπολογιστών έχει πάψει να είναι αποτρεπτικός παράγοντας.

3.4 MySQL

Το MySQL προέρχεται από το My, το όνομα της κόρης του ενός από τους ιδρυτές Michael 'Monty' Widenius (Μόντυ Βιντένιους), και το SQL αντιστοιχεί στο “Structured Query Language”, που σημαίνει «Δομημένη Γλώσσα Ερωτημάτων».

Οι σχεσιακές βάσεις δεδομένων, δηλαδή αυτές που χρησιμοποιούν την γλώσσα SQL για την επεξεργασία και την ανάπτυξη δεδομένων είναι η προφανής επιλογή για την αποθήκευση μίας πληθώρας δεδομένων, διευκολύνοντας την ανάκτηση τους ανάλογα με πολλά κριτήρια συνδυάζοντας δεδομένα από διαφορετικούς πίνακες.

Σαν βάση δεδομένων χρησιμοποιήσαμε την MySQL. Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (Relational Database Management System - RDBMS), δηλαδή ένας server που μπορεί να περιέχει πολλές διαφορετικές μεταξύ τους βάσεις δεδομένων. Κάθε βάση έχει ένα μοναδικό όνομα και μπορεί να περιέχει πολλούς πίνακες, ενώ τα περιεχόμενα δύο διαφορετικών βάσεων δεν σχετίζονται μεταξύ τους.

Πρόκειται για την περισσότερο διαδεδομένη βάση δεδομένων, με εκατομμύρια εγκαταστάσεις σε όλο τον κόσμο. Προσφέρει αρκετά καλές επιδόσεις και υποστηρίζει ένα μεγάλο εύρος της γλώσσας SQL, αν και έχει κάποιες ελλείψεις σε σχέση με άλλες βάσεις, με πιο εμφανή στο θέμα της υποστήριξης των ξένων κλειδιών.

Με τις κατάλληλες εντολές συνδεόμαστε στη βάση και μετά χρησιμοποιούμε κάποιες συναρτήσεις που παίρνουν για παραμέτρους ερωτήματα SQL σαν string. Οποιοδήποτε ερώτημα μπορεί να εκτελεστεί, είτε πρόκειται απλή επερώτηση είτε εάν σχετίζεται με δημιουργία νέων εγγραφών, τροποποίηση υπαρχόντων ή διαγραφή.

Το πρόγραμμα τρέχει έναν εξυπηρετητή (server) ο οποίος παρέχει πρόσβαση πολλών χρηστών σε ένα σύνολο βάσεων δεδομένων. Είναι γνωστή για διαδικτυακά προγράμματα και ιστοσελίδες και χρησιμοποιείται στις πιο διαδεδομένες ιστοσελίδες όπως το Twitter, το Google, τη Wikipedia και το YouTube. Μία βάση επιτρέπει να αποθηκεύσουμε, να ταξινομήσουμε και να αναζητήσουμε αποτελεσματικά τα δεδομένα. Η MySQL ελέγχει την πρόσβαση στα δεδομένα μας ώστε να μπορούν να δουλεύουν πολλοί χρήστες ταυτόχρονα, αλλά και να ελέγχει ότι μόνο πιστοποιημένοι χρήστες θα μπορούν να έχουν πρόσβαση.

3.4.1 Ορισμένες Εντολές της MySQL

- **Διαχείριση χρηστών.**
  Δημιουργία:
  create user username@hostname *identified by* password;

  Διαγραφή:
  drop user username@hostname;

- **Διαχείριση βάσης δεδομένων.**
  Δημιουργία βάσης δεδομένων:
  create database όνομα_βάσης;

  Διαγραφή βάσης δεδομένων:
  drop database όνομα_βάσης;

  Εμφάνιση των βάσεων που έχει δικαιώματα ο χρήστης:
  show databases;

- **Διαχείριση Δικαιωμάτων σε χρήστη.**
  Εκχώρηση δικαιωμάτων σε χρήστη:
  *grant [*privilege*] on[*db*][*.table*]to* user@host [*identified by* password] *[with grant option]*;

  Ανάκληση:
  revoke [privilege] *on* [db] [.table] [column] *from* user@host;

3.4.2 Πλεονεκτήματα της MYSQL

- Είναι πολύ γρήγορο και δυνατό σύστημα διαχείρισης βάσεων δεδομένων.
- Μπορούν πολλές συνδέσεις με τη βάση να υπάρχουν ταυτόχρονα χωρίς να υπάρχουν πολλαπλά αντίγραφα της.
- Η απόδοση της είναι καλύτερη σε μεγαλύτερο όγκο βάσεων δεδομένων.
- Είναι οικονομική.
- Είναι λογισμικό ανοιχτού κώδικα.
- Είναι γρήγορη στην ανάκτηση δεδομένων.
- Παρέχει ευκολίες στο backup.

3.4.3 Μειονεκτήματα της MySQL

- Οι συναλλαγές δεν αντιμετωπίζονται αρκετά αποτελεσματικά.
- Δεν υποστηρίζει ένα μεγάλο μέγεθος της βάσης δεδομένων.
- Δεν υποστηρίζει ξένα κλειδιά.

3.5 Java

Η Java είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού που σχεδιάστηκε από την εταιρεία πληροφορικής Sun Microsystems. Ένα από τα βασικά πλεονεκτήματα της Java έναντι των περισσότερων άλλων γλωσσών είναι ότι τα προγράμματα που είναι γραμμένα σε Java τρέχουνε ακριβώς το ίδιο σε Windows, Linux, Unix και Macintosh (χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα δηλαδή προσφέρει η ανεξαρτησία του λειτουργικού συστήματος και πλατφόρμας.

3.5.1 Τα χαρακτηριστικά της Java

Ένα από τα βασικά πλεονεκτήματα της Java εν συγκρίσει με των υπολοίπων περισσοτέρων γλωσσών είναι η ανεξαρτησία που διαθέτει όπως και προαναφέραμε εισαγωγικά προ λίγου. Για να επιτευχθεί ο στόχος αυτός όμως, χρειάστηκε να βρεθεί κάποιος τρόπος έτσι ώστε τα προγράμματα που ήταν γραμμένα σε Java να μπορούν να είναι «κατανοητά» από κάθε υπολογιστή ανεξάρτητα από το είδος του επεξεργαστή (Intel x86, IBM, Sun SPARC, Motorola) αλλά και του λειτουργικού συστήματος (Windows, Unix, Linux, BSD, MacOS). Ο λόγος που συμβαίνει αυτό είναι ότι κάθε κεντρική μονάδα επεξεργασίας κατανοεί διαφορετικό κώδικα μηχανής. Κατά συνέπεια ο συμβολικός κώδικας (assembly) που μεταφράζεται και εκτελείται σε Windows είναι διαφορετικός από αυτόν που μεταφράζεται και εκτελείται σε έναν υπολογιστή Macintosh. Η λύση προήλθε με την ανάπτυξη της Εικονικής Μηχανής (VirtualMachine ή VM ή ΕΜ στα ελληνικά).Σύμφωνα με την Εικονική Μηχανή όταν γραφτεί κάποιο πρόγραμμα σε Java, στη συνέχεια μεταγλωττίζεται μέσω του μεταγλωττιστή javac, ο οποίος παράγει έναν αριθμό από αρχεία .class (κώδικας byte ή bytecode). Ο κώδικας byte είναι η μορφή που παίρνει ο πηγαίος κώδικας της Java όταν μεταγλωττιστεί. Όταν χρειαστεί να εκτελεστεί η εφαρμογή σε ένα μηχάνημα, το JavaVirtualMachine που πρέπει να είναι εγκατεστημένο σε αυτό θα αναλάβει να διαβάσει τα αρχεία .class. Στη συνέχεια τα μεταφράζει σε γλώσσα μηχανής που να μπορούν να υποστηριχτεί από το λειτουργικό σύστημα και τον επεξεργαστή, έτσι ώστε να εκτελεστεί (να σημειωθεί εδώ ότι αυτό συμβαίνει με την παραδοσιακή Εικονική Μηχανή (VirtualMachine). Οι πιο σύγχρονες εφαρμογές της Εικονικής Μηχανής μπορούν και μεταγλωττίζουν εκ των προτέρων τμήματα bytecode απευθείας σε κώδικα μηχανής (εγγενή κώδικα ή nativecode) με αποτέλεσμα να βελτιώνεται η ταχύτητα. Χωρίς αυτό δε θα ήταν δυνατή η εκτέλεση λογισμικού γραμμένου σε Java. Αξίζει να σημειωθεί ότι η JVM είναι λογισμικό που εξαρτάται από την πλατφόρμα, δηλαδή για κάθε είδος λειτουργικού συστήματος και αρχιτεκτονικής επεξεργαστή υπάρχει διαφορετική έκδοση του. Έτσι υπάρχουν διαφορετικές JVM για Windows, Linux, Unix, Macintosh, κινητά τηλέφωνα, παιχνιδομηχανές κλπ.

Οτιδήποτε θέλει να κάνει ο προγραμματιστής (ή ο χρήστης) γίνεται μέσω της εικονικής μηχανής. Εξαιτίας αυτού υπάρχει μεγαλύτερη ασφάλεια στο σύστημα γιατί η εικονική μηχανή είναι υπεύθυνη για την επικοινωνία χρήστη - υπολογιστή. Γενικά ισχύει ότι ο προγραμματιστής δεν μπορεί να γράψει κώδικα ο οποίος θα έχει καταστροφικά αποτελέσματα για τον υπολογιστή γιατί η εικονική μηχανή θα τον ανιχνεύσει και δε θα επιτρέψει να εκτελεστεί. Παρομοίως, στην άλλη μεριά ούτε ο χρήστης μπορεί να κατεβάσει «κακό» κώδικα από το δίκτυο και να τον εκτελέσει. Αυτό είναι ιδιαίτερα χρήσιμο για μεγάλα κατανεμημένα συστήματα όπου πολλοί χρήστες χρησιμοποιούν το ίδιο πρόγραμμα συγχρόνως.

Η ύπαρξη του συλλέκτη απορριμμάτων (GarbageCollector) αποτελεί ακόμα μία ιδέα που βρίσκεται πίσω από τη Java. Συλλογή απορριμμάτων είναι μία κοινή ονομασία που χρησιμοποιείται στον τομέα της πληροφορικής για να δηλώσει την ελευθέρωση τμημάτων μνήμης από δεδομένα που δε χρειάζονται και δε χρησιμοποιούνται άλλο. Στη Java η απελευθέρωση μνήμης είναι αυτόματη και γίνεται μέσω του συλλέκτη απορριμμάτων. Η εικονική μηχανή είναι και πάλι υπεύθυνη για αυτό και μόλις «καταλάβει» ότι ο σωρός (heap) της μνήμης (στη Java η συντριπτική πλειοψηφία των αντικειμένων αποθηκεύονται στο σωρό σε αντίθεση με τη C++ όπου αποθηκεύονται κυρίως στη στοίβα) κοντεύει να γεμίσει ενεργοποιεί το συλλέκτη απορριμμάτων. Συνεπώς ο προγραμματιστής δε χρειάζεται να ανησυχεί για το πότε και αν θα ελευθερώσει ένα συγκεκριμένο τμήμα της μνήμης, ούτε και για σφάλματα δεικτών. Αυτό είναι ιδιαίτερα σημαντικό γιατί είναι κοινά τα σφάλματα προγραμμάτων που οφείλονται σε λανθασμένο χειρισμό της μνήμης.

Παρόλο που η εικονική μηχανή προσφέρει όλα αυτά τα πλεονεκτήματα, η Java αρχικά ήταν πιο αργή σε σχέση με άλλες προγραμματιστικές γλώσσες υψηλού επιπέδου

(high-level) όπως η C και η C++. Στο παρελθόν εμπειρικές μετρήσεις είχαν δείξει ότι η C++ μπορούσε να είναι αρκετές φορές γρηγορότερη από την Java. Παρόλα αυτά γίνονται προσπάθειες από τη Sun για τη βελτιστοποίηση της εικονικής μηχανής, ενώ υπάρχουν και άλλες υλοποιήσεις της εικονικής μηχανής από διάφορες εταιρίες (όπως της IBM), οι οποίες μπορεί σε κάποια σημεία να προσφέρουν καλύτερα και σε κάποια άλλα χειρότερα αποτελέσματα. Επιπλέον με την καθιέρωση των μεταγλωττιστών JIT (JustInTime), οι οποίοι μετατρέπουν τον κώδικα byte απευθείας σε γλώσσα μηχανής, η διαφορά ταχύτητας από τη C++ έχει μικρύνει κατά πολύ.

3.5.2 Java servlet

Ένα Java servlet είναι ένα πρόγραμμα γραμμένο στην γλώσσα προγραμματισμού Java το οποίο επεκτείνει τις δυνατότητες ενός server. Αν και οι server μπορούν να ανταποκρίνονται σε πολλούς τύπους αιτημάτων- απαιτήσεων, μπορούν γενικότερα να υλοποιούν εφαρμογές που φιλοξενούνται σε servers διαδικτυακούς. Τα servlets παρέχουν την δυνατότητα στην java να μπορέσει να υλοποιήσει και να ανταγωνιστεί τις υπόλοιπες γλώσσες προγραμματισμού και διαδικτύου σε τεχνολογίες όπως την PHP και ASP.NET.

3.5.2.1 Εισαγωγή στα Servlets

Τα Servlets είναι ευρέως γνωστά για την χρήση τους σε διάφορους τομείς για τις εξής ικανότητες τους:

Για την επεξεργασία και αποθήκευση μιας Java class σε Java EE η οποία επιβεβαιώνεται και καθορίζεται μέσω λειτουργιών και μεθόδων που παρέχονται από το εκάστοτε Java Servlet API που αποτελεί μια στάνταρ αλληλουχία διαδικασιών για την υλοποίηση Java κλάσεων που απαιτείται να ανταποκρίνονται σε αιτήματα. Τα Servlets μπορούν κατά κόρων να επικοινωνούν πάνω σε πρωτόκολλα client-server, αλλά και να χρησιμοποιούνται ευρέως σε HTTP πρωτόκολλα. Επομένως, εν συντομία ένα "servlet" χρησιμοποιείται σαν μια συντόμευση ονόματος για το "HTTP servlet". Κατά συνέπεια, ένας προγραμματιστής λογισμικού μπορεί να χρησιμοποιήσει ένα servlet για την προσθήκη δυναμικού περιεχομένου σε ένα server διαδικτύου που χρησιμοποιεί και προγραμματίζει σε Java platform. Το παραγόμενο περιεχόμενο που επιστρέφει είναι κατά κύριο λόγο HTML αλλά μπορεί να δώσει και αποτελέσματα δεδομένων και σε άλλες μορφές όπως είναι σε XML. Τα Servlets μπορούν να διατηρούν την κατάσταση σε επίπεδο session μεταβλητών καθώς εξελίσσονται συναλλαγές – δοσοληψίες μεταξύ των server με την χρήση HTTP cookies, η επαναπληκτρολόγιση διευθύνσεων.

Η χρήση ενός διαδικτυακού container είναι απαραίτητο για το στήσιμο, τον προγραμματισμό, τον σχεδιασμό και το τρέξιμο ενός servlet. Το διαδικτυακό container (γνωστό και ως servlet container) είναι ουσιαστικά ένα εξάρτημα- τμήμα του διαδικτυακού server που αλληλοεπιδρά με τα servlets και είναι υπεύθυνο για να διαχειριστεί τον κύκλο ζωής των servlets καθώς και να χαρτογραφήσει ένα URL σε ένα συγκεκριμένο servelt και να έχει την δυνατότητα να επιβεβαιώσει στον server ότι το αίτημα για το εκάστοτε URL έχει πρόσβαση με σωστά και επαληθευμένα δικαιώματα

Το Servlet API περιέχεται ως ένα επιπλέον package της Java με ιεραρχία javax.servlet, το οποίο ορίζει τις απαιτούμενες μεθόδους για την δυνατότητα υλοποίησης σωστών αλληλεπιδράσεων μεταξύ των διαδικτυακών container και των servlet.

Ένα Servlet είναι ένα αντικείμενο το οποίο λαμβάνει αιτήματα και αποστέλλει απαντήσεις βάση του αιτήματος που έλαβε. Το βασικό Servlet package ορίζει αντικείμενα της Java που αναπαριστούν τα αιτήματα και τις απαντήσεις των servlet με τέτοιο τρόπο ώστε να ανταποκρίνονται και να αναπαριστούν σωστές παραμετροποιήσεις σε περιβάλλοντα

εκτέλεσης. Το pakage javax.servlet.http ορίζει συγκεκριμένες υποκλάσεις του HTTP που παράγουν στοιχεία τύπου generic servlet, τα οποία περιλαμβάνουν αντικείμενα διαχείρισης session που ανιχνεύουν πολλαπλές αιτήσεις και απαντήσεις μεταξύ server διαδικτύου και client. Τα Servlets μπορούν να πακεταριστούν σε ένα WAR αρχείο ως μια εφαρμογή διαδικτύου

Τα servlets μπορούν να παραχθούν αυτόματα από τις σελίδες τύπου server της Java, τα λεγόμενα Java Server Pages (JSP) από τον JavaServer Pages compiler. Η διαφορά μεταξύ servlets και JSP είναι ότι τυπικά τα servlets εμπεριέχουν HTML μέσα σε κώδικα Java, ενώ τα JSPs εμπεριέχουν κώδικα Java code σε HTML. Ενώ η άμεση χρησιμότητα των servlets να παράγουν HTML μπορεί να δημιουργείται σπάνια, το πιο υψηλό επίπεδο MVC web framework σε Java EE (JSF) ακόμα χρησιμοποίει την τεχνολογία των servlet για χαμηλού επιπέδου αιτήματα και απαιτήσεις που χειρίζονται μέσω FacesServlet. Μια παλιότερη χρησιμότητα είναι η χρήση των servlets σε συνδυασμό με JSPs σε ένα μοτίβο γνωστό και ως "Model 2", το οποίο αποτελεί μια μικρή γεύση του model–view–controller.

3.5.2.2 Ιστορία

Η δημιουργία εξειδίκευσης των Servlet δημιουργήθηκε από την SUN Microsystems, με την έκδοση 1.0 και οριστικοποιήθηκε τον Ιούνιο του 1997. Ξεκινώντας με την έκδοση 2.3, εξειδίκευση αναπτύχθηκε κάτω από την επίβλεψη της Κοινότητα Επεξεργασίας της Java. Η JSR 53 ορίστηκε από τις εξειδικεύσεις τόσο των Servlet 2.3 όσο και των JavaServer Page 1.2 εξειδικεύσεων. Η έκδοση JSR 154 εξειδικεύεται στα Servlet 2.4 και 2.5 εξειδικεύσεις. Η τρέχουσα έκδοση των servlets τρέχει από τον Μάρτιο του 2010 ως Servlet εξειδίκευση 3.0.

Ο Jim Driscoll ήταν ο πρώτος που σκέφτηκε και ανέλυσε λεπτομερώς την τεχνολογία των servlet. Ο James Gosling ήταν αυτός που προσπάθησε να προωθήσει και να υλοποιήσει σε πράξη την θεωρία των servlets αλλά αυτό το κατάφερε και το έκδωσε σαν προϊόν στην αγορά η Sun ως Java Web Server

Servlet API history

Servlet API version	Released	Platform	Important Changes
Servlet 3.1	May 2013	JavaEE 7	Non-blocking I/O, HTTP protocol upgrade mechanism (WebSocket)[5]
Servlet 3.0	December 2009	JavaEE 6, JavaSE 6	Pluggability, Ease of development, Async Servlet, Security, File Uploading
Servlet 2.5	September 2005	JavaEE 5, JavaSE 5	Requires JavaSE 5, supports annotation
Servlet 2.4	November 2003	J2EE 1.4, J2SE 1.3	web.xml uses XML Schema
Servlet 2.3	August 2001	J2EE 1.3, J2SE 1.2	Addition of `Filter`
Servlet 2.2	August 1999	J2EE 1.2, J2SE 1.2	Becomes part of J2EE, introduced independent web applications in .war files
Servlet 2.1	November 1998	Unspecified	First official specification, added `RequestDispatcher`, `ServletContext`
Servlet 2.0		JDK 1.1	Part of Java Servlet Development Kit 2.0
Servlet 1.0	June 1997		

3.5.2.3 Servlets σε σύγκριση με άλλα μοντέλα εφαρμογών διαδικτύου

Τα πλεονεκτήματα χρήσης των Servlets είναι η γρήγορη επίδοση τους και η ευκολία χρήσης σε συνδυασμό με περισσότερη δυναμική πέρα από τις παραδοσιακές CGI διεπαφές (Common Gateway Interface) τους. Παραδοσιακά σενάρια CGI γραμμένα σε Java δυστυχώς έχουν έχει μια σειρά από μειονεκτήματα απόδοσης που είναι τα ακόλουθα:

Όταν ένα αίτημα HTTP πραγματοποιηθεί, μια νέα διαδικασία δημιουργείται κάθε φορά που το σενάριο CGI καλείται. Η κάλεση του σχετίζεται με τη δημιουργία της διαδικασίας που μπορεί να δημιουργήσει καθυστερήσεις στον φόρτο εργασίας και

δυνατοτήτων της εφαρμογής, ιδίως όταν το σενάριο κάνει σχετικά γρήγορα πράξεις. Επομένως, η δημιουργία της διαδικασίας θα χρειαστεί περισσότερος χρόνος για την εκτέλεση του script CGI παρά την σχετική απόδοση και λειτουργικότητα που μπορεί να παρέχει. Αντίθετα, για τα servlets, κάθε αίτηση γίνεται από ένα ξεχωριστό νήμα Java στο πλαίσιο της διαδικασίας του web server, αποφεύγοντας έτσι την επιβάρυνση που συνδέεται με την δημιουργία εικονικών διεργασιών πανομοιότυπες με την ίδια μέσα στον δαίμονα HTTP.

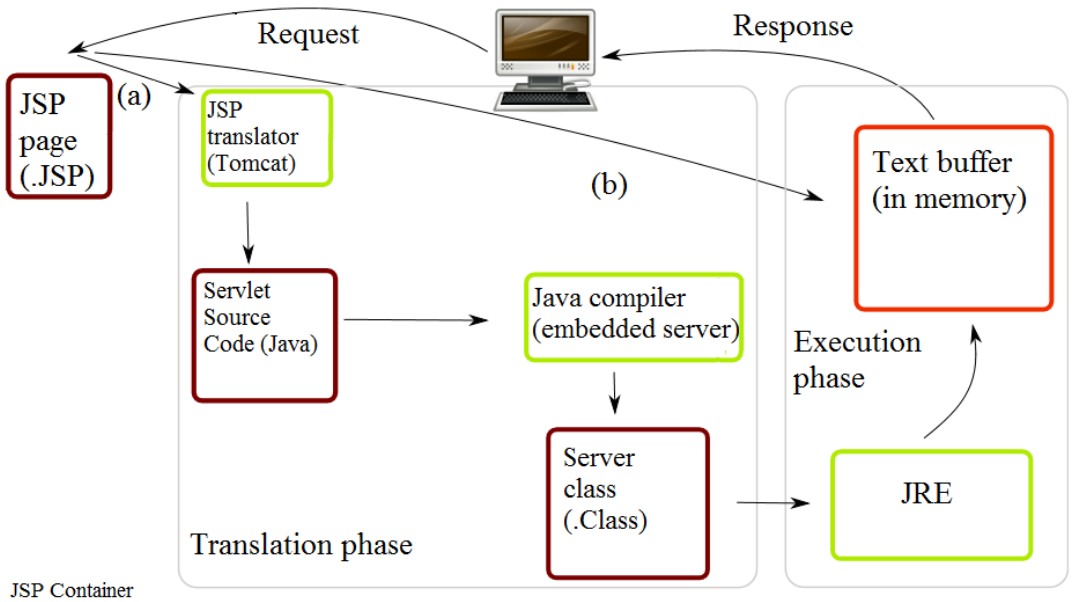
Η διαδικασία ταυτόχρονης αίτησης CGI είναι χρονοβόρα και βαριά καθώς θα πρέπει να φορτώσει το σενάριο CGI που θα αντιγραφεί στη μνήμη μία φορά ανά αίτηση ενώ με τα servlets, υπάρχει μόνο ένα αντίγραφο, που επιμένει σε όλη αιτήματα και μοιράζεται μεταξύ των νημάτων που ενεργούν για την ολοκλήρωση των αιτημάτων προς απάντηση από τον server διαδικτύου.

Η αναπαράσταση του αιτήματος με ένα μόνο στιγμιότυπο δίνει την δυνατότατα απάντησης σε όλα τα αιτήματα ταυτόχρονα. Αυτό έχει ως αποτέλεσμα να μειώνει τη χρήση της μνήμης και διευκολύνει τη διαχείριση των δεδομένων.

Ένα servlet μπορεί να εκτελεστεί από ένα servlet container σε ένα περιοριστικό περιβάλλον, που ονομάζεται sandbox. Αυτό είναι παρόμοιο με ένα applet που τρέχει στο sandbox του web browser. Αυτό επιτρέπει περιορισμένη χρήση των δυνητικά επιβλαβών servlets. Τα CGI προγράμματα μπορούν από μόνα τους να αποτελέσουν sandbox και τα ίδια, δεδομένου ότι είναι απλά OS διαδικασίες.

Τεχνολογίες όπως η FastCGI και τα παράγωγά του (συμπεριλαμβανομένης της SCGI, WSGI) δεν παρουσιάζουν τα μειονεκτήματα των επιδόσεων των CGI που προκύπτουν από τη συνεχή επεξεργασία. Είναι, ωστόσο, κατά προσέγγιση τόσο απλό όπως τα CGI. Επομένως, είναι σε μόνιμη αντίθεση με τα servlets τα οποία είναι ουσιαστικά πιο περίπλοκα.

3.5.2.4 Ο κύκλος ζωής ενός servlet



(a) Translation occurs at this point, if JSP has been changed or is new.
(b) If not, translation is skipped.

Τρεις μέθοδοι παίζουν καθοριστικό ρόλο για τη διάρκεια του κύκλου ζωής ενός servlet. Αυτές είναι η init(), service(), και destroy(). Χρησιμοποιούνται σε κάθε εφαρμογή από ένα servlet και γίνεται επίκληση αυτών των μεθόδων σε συγκεκριμένες ώρες από το διακομιστή - server.

Κατά το πρώτο στάδιο αρχικοποίησης του κύκλου ζωής ενός servlet, το container διαδικτύου προετοιμάζει ένα στιγμιότυπο του servlet, καλώντας τη μέθοδο init (), περνώντας ένα αντικείμενο που υλοποιεί το javax.servlet.ServletConfiginterface. Αυτό το αντικείμενο επιτρέπει την διαμόρφωση του servlet για να αποκτήσει πρόσβαση σε παραμέτρους αρχικοποίησης από την διαδικτυακή εφαρμογή.

Μετά την αρχικοποίηση , το στιγμιότυπου του servlet μπορεί να εξυπηρετήσει τα αιτήματα των πελατών - client. Κάθε αίτηση εξυπηρετείται στο δικό της ξεχωριστό νήμα. Το web container καλεί τη μέθοδο service () του servlet για κάθε αίτηση. Η μέθοδος service () προσδιορίζει το είδος του αιτήματος που γίνεται και τις αποστολές του με ένα κατάλληλο τρόπο έτσι ώστε να γίνει η σωστή διαχείριση της αίτησης. Ο προγραμματιστής που γράφει το servlet πρέπει να παρέχει μια υλοποίηση υποστήριξης αυτών των μεθόδων. Αν μια αίτηση γίνεται για μια μέθοδο που δεν υλοποιείται από το servlet, τυπικά η μέθοδος της γονικής κλάσης αναλαμβάνει ώστε ως αποτέλεσμα να επιστραφεί ένα σφάλμα στον αιτούντα και ο προγραμματιστής να μεριμνήσει για αυτό σχετικά με το error που παρουσιάστηκε.

Τέλος, το container διαδικτύου καλεί τη μέθοδο destroy () που θέτει το servlet εκτός λειτουργίας μετά την ολοκλήρωση του αιτήματος που είχε να ασχοληθεί. Η μέθοδος destroy (), όπως και η init (), καλείται μόνο μία φορά κατά τη διάρκεια του κύκλου ζωής ενός servlet.

Το ακόλουθο είναι ένα τυπικό σενάριο χρήσης των μεθόδων αυτών.

1. Ας υποθέσουμε ότι ένας χρήστης ζητά να επισκεφθεί ένα URL.
    a. Το πρόγραμμα περιήγησης στη συνέχεια, δημιουργεί μια αίτηση HTTP για αυτό το URL.
    b. Το αίτημα αυτό αποστέλλεται στη συνέχεια στο κατάλληλο διακομιστή.
2. Η αίτηση HTTP λαμβάνεται από τον server διαδικτύου και προωθείται στο container με τα Servlet.
    a. Το container αντιστοιχίζει αυτό το αίτημα σε ένα συγκεκριμένο servlet ανάλογα με το αίτημα που ζητήθηκε.
    b. Το servlet δυναμικά ανακτάται και τοποθετείται στον ανάλογο χώρο διευθύνσεων του conatiner.
3. Το container επικαλείται τη μέθοδος init () του servlet.
    a. Η μέθοδος αυτή επικαλείται μόνο όταν το servlet παραφορτωθεί στη μνήμη.
    b. Υπάρχει η περίπτωση φόρτωσης ορισμάτων-παραμέτρων κατά την αρχικοποίηση του Servlet που του παρέχει την δυνατότητα να μπορεί να αυτορυθμιστεί.
4. Το δοχείο επικαλείται τη μέθοδο service () του servlet.
    a. Αυτή η μέθοδος καλείται για να επεξεργαστεί την αίτηση HTTP.
    b. Το servlet μπορεί να διαβάσει τα δεδομένα που έχουν παραχθεί στην αίτηση HTTP.
    c. Το servlet μπορεί επίσης να διατυπώνει μια απόκριση HTTP για τον πελάτη.
5. Το servlet παραμένει στο χώρο διευθύνσεων του container και είναι διαθέσιμο για την επεξεργασία οποιωνδήποτε άλλων αιτήσεων HTTP που λαμβάνονται από τους πελάτες.
    a. Η μέθοδος service () καλείται για κάθε αίτηση HTTP.
6. Το δοχείο μπορεί, οποιαδήποτε χρονική στιγμή επιθυμεί, να αποφασίσει να ξεφορτώσει το servlet από τη μνήμη του.
    a. Οι αλγόριθμοι με τους οποίους γίνεται η παρούσα απόφαση είναι ειδικές για κάθε δοχείο.
7. Το δοχείο καλεί τη μέθοδο του servlet destroy () ώστε να παραιτηθεί από τυχόν πόρους, όπως χειρισμού αρχείων που διατίθενται για την servlet
8. Η μνήμη που διατίθεται για το Servlet και τα αντικείμενά του μπορεί στη συνέχεια αυτοκαταστραφούν

3.5.2.5 Ένα παράδειγμα των Servet

Το ακόλουθο παράδειγμα με servlets εκτυπώνει πόσες φορές καλείται η μέθοδος service().

Σημειώνεται ότι η HttpServlet είναι υποκλάση της GenericServlet, και υλοποιεί μια διεπαφή του Servlet

Η μέθοδος service() είναι μέθοδος της HttpServlet κλάσης και διανέμει αιτήματα με τη χρήση μεθόδων όπως doGet(), doPost(), doPut(), doDelete(), ανάλογα με το εκάστοτε HTTP αίτημα. Στο παράδειγμα παρακάτω η μέθοδος service() υπερκαλύπτεται και δεν κατανέμει ποιο HTTP αίτημα εξυπηρετεί.

```
import java.io.IOException;

 import javax.servlet.ServletConfig;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


public class ServletLifeCycleExample extends HttpServlet {

   private int count;

   @Override

   public void init(ServletConfig config) throws ServletException {

      super.init(config);

      getServletContext().log("init() called");

      count = 0;

   }


   @Override

   protected void service(HttpServletRequest request, HttpServletResponse response)

         throws ServletException, IOException {

      getServletContext().log("service() called");
```

```
    count++;
    response.getWriter().write("Incrementing the count: count = " + count);
  }
  @Override
  public void destroy() {
    getServletContext().log("destroy() called");
  }
```

3.6 JDBC

JDBC είναι μια τεχνολογία Java σύνδεσης βάσεων δεδομένων (η γνωστή πλατφόρμα Java Standard Edition) από την Oracle Corporation . Αυτή η τεχνολογία είναι ένα API για τη γλώσσα προγραμματισμού Java που ορίζει τον τρόπο με τον οποίο ο πελάτης μπορεί να έχει πρόσβαση σε μια βάση δεδομένων . Παρέχει μεθόδους για την αναζήτηση και την ενημέρωση των δεδομένων σε μια βάση δεδομένων . JDBC συγκεκριμενοποιείται σε σχεσιακές βάσεις δεδομένων . Μια γέφυρα JDBC -προς- ODBC επιτρέπει συνδέσεις σε οποιαδήποτε πηγή δεδομένων ODBC προσβάλαμε στο περιβάλλον υποδοχής JVM .

3.6.1 Ιστορία και υλοποίηση

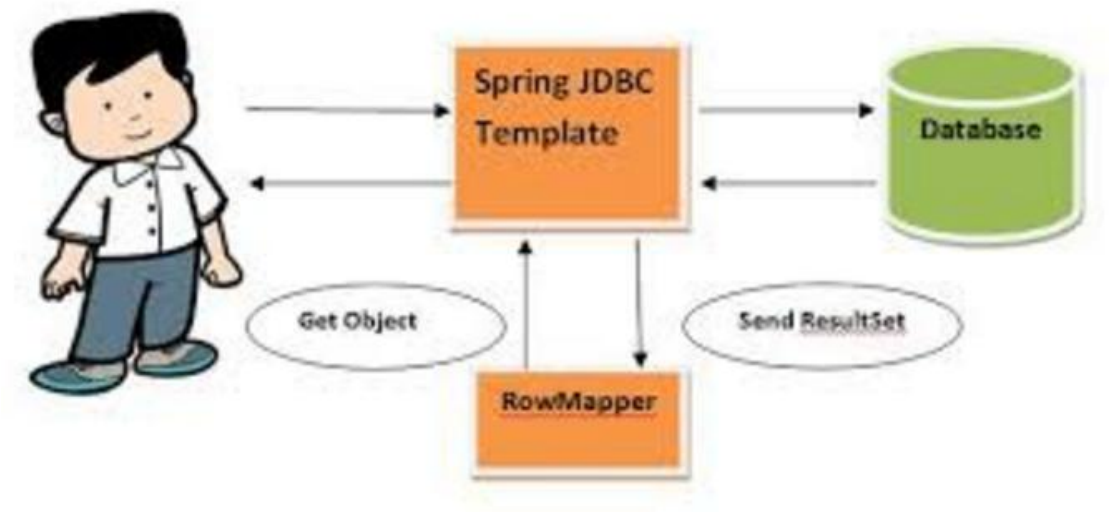
Η Sun Microsystems κυκλοφόρησε την JDBC ως μέρος του JDK 1.1 στις 19 Φεβρουαρίου , 1997. Από τότε είναι μέρος της Java Standard Edition .

Οι JDBC κλάσεις περιέχονται στο pakage java.sql και javax.sql Java .

Ξεκινώντας με την έκδοση 3.1 ,η JDBC έχει αναπτυχθεί στο πλαίσιο της κοινότητας Επεξεργασίας της Java . JSR 54 εξειδικεύεται στην JDBC 3.0 (που περιλαμβάνεται στο J2SE 1.4) , JSR 114 καθορίζει τις προσθήκες των JDBC rowset , και η JSR 221 αποτελεί τον προσδιορισμός του JDBC 4,0 (συμπεριλαμβανομένου της Java SE 6)

Η JDBC 4.1 , προσδιορίζεται με δελτίο συντήρησης 1 του JSR 221 και περιλαμβάνεται στην Java SE 7.

Η τελευταία έκδοση , JDBC 4.2 , προσδιορίζεται με δελτίο συντήρησης 2 του JSR 221 [5] και περιλαμβάνεται στην Java SE 8. [6]



3.6.2 Λειτουργικότητα

JDBC επιτρέπει σε πολλαπλές υλοποιήσεις να υπάρχουν και να χρησιμοποιούνται από την ίδια εφαρμογή. Το API παρέχει ένα μηχανισμό για τη δυναμική φόρτωση των σωστών πακέτων Java και την εγγραφή τους με το πρόγραμμα JDBC Driver Manager. Ο Manager Driver χρησιμοποιείται ως ένα εργοστάσιο σύνδεσης για τη δημιουργία JDBC συνδέσεις.

Οι JDBC συνδέσεις υποστηρίζουν τη δημιουργία και την εκτέλεση δηλώσεων. Αυτά μπορεί να είναι δηλώσεις όπως SQL CREATE, INSERT, UPDATE και DELETE, ή μπορεί να είναι δηλώσεις ερώτημα, όπως SELECT. Επιπλέον, αποθηκευμένες διαδικασίες μπορεί να επανακληθούν μιας σύνδεσης JDBC. Η JDBC αντιπροσωπεύει δηλώσεις που χρησιμοποιούν τις ακόλουθες κλάσεις:

• Statement - η δήλωση αποστέλλεται στον διακομιστή της βάσης δεδομένων κάθε φορά.

• PreparedStatement - η δήλωση είναι προσωρινά αποθηκευμένη και στη συνέχεια η πορεία εκτέλεσης της προκαθορίζεται στο διακομιστή της βάσης δεδομένων που επιτρέπει να εκτελεστεί πολλές φορές με αποτελεσματικό τρόπο.

• CallableStatement - που χρησιμοποιούνται για την εκτέλεση αποθηκευμένων διαδικασιών στη βάση δεδομένων.

Οι Δηλώσεις Ενημέρωσης όπως INSERT, UPDATE και DELETE επιστρέφουν ένα καταμετρητή ενημερωμένη έκδοσης που δείχνει πόσες γραμμές επιλέγηκαν στη βάση δεδομένων. Αυτές οι δηλώσεις δεν επιστρέφουν οποιαδήποτε άλλη πληροφορία.

Οι Δηλώσεις ερωτημάτων QUERY επιστρέφουν ένα σύνολο αποτελεσμάτων στην JDBC. Οι μεμονωμένες στήλες σε μια σειρά ανακτώνται είτε με βάση το όνομα ή τον αριθμό της στήλης. Μπορεί να υπάρχει οποιοσδήποτε αριθμός των γραμμών στο σύνολο αποτελεσμάτων. Το σύνολο των αποτελεσμάτων έχει τα μεταδεδομένα που περιγράφουν τα ονόματα των στηλών και τους τύπους τους.

Υπάρχει μια επέκταση του βασικού JDBC API στην javax.sql.

3.6.3 Παραδείγματα

Η μέθοδος Class.forName(String) χρησιμοποιείται για να φορτώσει την JDBC driver κλάση. Η εντολή παρακάτω προκαλεί την δημιουργία ενός στιγμιότυπου της JDBC driver σε μία εφαρμογή. (Κάποια JVMs χρειάζονται και την μέθοδο .newInstance().)

```
Class.forName( "com.somejdbcvendor.TheirJdbcDriver" );
```

Στην JDBC 4.0 , δεν είναι πλέον απαραίτητο να φορτωθούν οι JDBC οδηγοί καθώς χρησιμοποιείται η Class.forName ( ) .

Όταν μια κλάση φορτώνεται το πρόγραμμα οδήγησης δημιουργεί ένα στιγμιότυπο που καταγράφεται με την DriverManager . Αυτό μπορεί να γίνει συμπεριλαμβανομένου ενός κώδικα που χρειάζεται στο στατικό μπλοκ της κλάσης οδηγού. Π.χ. , DriverManager.registerDriver (Driver driver)

Τώρα, όταν απαιτείται η σύνδεση , μια από τις DriverManager.getConnection ( ) μεθόδους χρησιμοποιείται για να δημιουργήσει μια σύνδεση JDBC .

```
Connection conn = DriverManager.getConnection(
    "jdbc:somejdbcvendor:other data needed by some jdbc vendor",
    "myLogin",
    "myPassword" );
try {
    /* you use the connection here */
} finally {
   //It's important to close the connection when you are done with it
   try { conn.close(); } catch (Throwable ignore) { /* Propagate the original exception
instead of this one that you may want just logged */ }
}
```

Ξεκινώντας με την Java SE 7 ο οποιασδήποτε μπορεί να χρησιμοποιήσει try-with-resources δηλώσεις της Java για να κάνει τον κώδικα πιο ευανάγνωστο και ξεκάθαρο:

```
try (Connection conn = DriverManager.getConnection(
    "jdbc:somejdbcvendor:other data needed by some jdbc vendor",
    "myLogin",
    "myPassword" ) ) {
```

```
    /* you use the connection here */

}  // the VM will take care of closing the connection
```

Η διεύθυνση URL που χρησιμοποιείται εξαρτάται από το συγκεκριμένο πρόγραμμα JDBC driver . Θα αρχίσει πάντα με το " JDBC : " « πρωτόκολλο. Μόλις καθιερωθεί μια σύνδεση , μπορεί να δημιουργηθεί μια δήλωση .

```
try (Statement stmt = conn.createStatement()) {

   stmt.executeUpdate( "INSERT INTO MyTable( name ) VALUES ( 'my name' ) " );

}
```

Σημειώνεται ότι οι συνδέσεις , δηλώσεις , και σύνολα αποτελεσμάτων συχνά δίνουν πόρους του λειτουργικού συστήματος , όπως περιγραφείς αρχείων . Στην περίπτωση των συνδέσεων σε απομακρυσμένους διακομιστές βάσεων δεδομένων , απαιτούνται περαιτέρω πόροι αλληλένδετοι με τον διακομιστή Είναι ζωτικής σημασίας η μέθοδος close ( ) οποιοδήποτε αντικείμενου στην JDBC, η οποία παίζει τον ρόλο συλλογή σκουπιδιών και δεν θα πρέπει να γίνει επίκληση . Ξεχνώντας την μέθοδο close ( ) τα πράγματα καταλήγουν σωστά σε παρασιτικά λάθη και κακή συμπεριφορά . Τα δεδομένα ανακτώνται από τη βάση δεδομένων με τη χρήση ενός μηχανισμού ερωτήματος στη βάση δεδομένων . Το παρακάτω παράδειγμα δείχνει τη δημιουργία μιας δήλωσης και την εκτέλεση ενός ερωτήματος .

```
try (Statement stmt = conn.createStatement();

   ResultSet rs = stmt.executeQuery( "SELECT * FROM MyTable" )

) {

   while ( rs.next() ) {

      int numColumns = rs.getMetaData().getColumnCount();

      for ( int i = 1 ; i <= numColumns ; i++ ) {

        // Column numbers start at 1.

        // Also there are many methods on the result set to return

        //  the column as a particular type. Refer to the Sun documentation

        //  for the list of valid conversions.

        System.out.println( "COLUMN " + i + " = " + rs.getObject(i) );

      }
```

```
   }
}
```

Συνήθως , όμως , αυτό θα ήταν σπάνιο για έναν έμπειρο προγραμματιστή της Java. Ο συνήθης προγραμματιστικός τρόπος είναι να χρησιμοποιείτε αφηρημένη λογική στην βάση δεδομένων που περιέχει SQL δηλώσεις και τη σύνδεση με τις απαιτούμενες μεθόδους . Αν αφήσουμε κατά μέρος το μοντέλο δεδομένων από τον κώδικα της εφαρμογής καθιστά πιο πιθανό ότι οι αλλαγές στο μοντέλο εφαρμογών και των δεδομένων μπορεί να γίνουν ανεξάρτητα .

Ένα παράδειγμα μιας ερώτησης PreparedStatement , χρησιμοποιώντας conn

```
try (PreparedStatement ps =
   conn.prepareStatement( "SELECT i.*, j.* FROM Omega i, Zappa j WHERE i.name = ? AND j.num = ?" )
){
   // In the SQL statement being prepared, each question mark is a placeholder
   // that must be replaced with a value you provide through a "set" method invocation.
   // The following two method calls replace the two placeholders; the first is
   // replaced by a string value, and the second by an integer value.
   ps.setString(1, "Poor Yorick");
   ps.setInt(2, 8008);

   // The ResultSet, rs, conveys the result of executing the SQL statement.
   // Each time you call rs.next(), an internal row pointer, or cursor,
   // is advanced to the next row of the result.  The cursor initially is
   // positioned before the first row.
   try (ResultSet rs = ps.executeQuery()) {
      while ( rs.next() ) {
         int numColumns = rs.getMetaData().getColumnCount();
         for ( int i = 1 ; i <= numColumns ; i++ ) {
            // Column numbers start at 1.
            // Also there are many methods on the result set to return
```

```
            // the column as a particular type. Refer to the Sun documentation
            // for the list of valid conversions.
            System.out.println( "COLUMN " + i + " = " + rs.getObject(i) );
          } // for
        } // while
    } // try
} // try
```

Αν μια λειτουργία βάσης δεδομένων αποτύχει , JDBC εγείρει ένα SQLException . H SQLException μπορεί να μεταφράσει μια εξαίρεση η οποία τελικά οδηγεί σε μια επαναφορά συναλλαγής και κοινοποίηση προς τον χρήστη .

Ένα παράδειγμα συναλλαγής με την βάση δεδομένων(database transaction)

```
boolean autoCommitDefault = conn.getAutoCommit();
try {
   conn.setAutoCommit(false);
    /* You execute statements against conn here transactionally */
    conn.commit();
} catch (Throwable e) {
   try { conn.rollback(); } catch (Throwable ignore) {}
   throw e;
} finally {
   try { conn.setAutoCommit(autoCommitDefault); } catch (Throwable ignore) {}
}
```

`setXXX()` **Methods**

Oracle Datatype	`setXXX()`
CHAR	`setString()`
VARCHAR2	`setString()`
NUMBER	`setBigDecimal()`
	`setBoolean()`
	`setByte()`
	`setShort()`
	`setInt()`
	`setLong()`
	`setFloat()`
	`setDouble()`
INTEGER	`setInt()`
FLOAT	`setDouble()`
CLOB	`setClob()`
BLOB	`setBlob()`
RAW	`setBytes()`
LONGRAW	`setBytes()`
DATE	`setDate()`
	`setTime()`
	`setTimestamp()`

Παραδείγματα of host database types which Java can convert to with a function.

3.7 Apache

Ο Apache Web Server είναι ένας εξυπηρετητής του παγκόσμιου ιστού. Κάθε φορά που ένας χρήστης επισκέπτεται κάποιον ιστότοπο, το πρόγραμμα πλοήγησης (browser) επικοινωνεί με ένα διακομιστή (server) μέσω του πρωτόκολλου HTTP, ο οποίος παράγει τις ιστοσελίδες και τις αποστέλλει στο πρόγραμμα πλοήγησης.

Ουσιαστικά ο ρόλος του Apache είναι να περιμένει αιτήσεις από κάποιους χρήστες - προγράμματα και στη συνέχεια να εξυπηρετεί τις αιτήσεις αυτές, μεταβιβάζοντας τους στις ιστοσελίδες που ζητούν μέσω μιας ηλεκτρονικής διεύθυνσης είτε άμεσα, είτε μέσω ενός συνδέσμου.

3.7.1 Χαρακτηριστικά και λειτουργίες

Δύο από τα βασικά χαρακτηριστικά του Apache, τα οποία του δίνουν πολλές δυνατότητες είναι τα παρακάτω:

1. Μπορούν να προσαρμοστούν σε αυτό προσθήκες προγραμμάτων (modules) τα οποία παρέχουν διαφορετικές λειτουργίες.

2. Ο Apache υποστηρίζει αρκετές εφαρμογές και γλώσσες προγραμματισμού όπως MySQL, PHP, Java κτλ.

Κάποια γνωστά modules του Apache HTTP είναι τα modules πιστοποίησης, δηλαδή τα mod_access, τα mod_digest κτλ.

Μέσω των mod_ssl και mod_proxy παρέχει SSL σε TLS ενώ μέσω του mod_rewrite πραγματοποιεί ανακατευθύνσεις διευθύνσεων.

3.7.2 Βασικές εντολές του Apache

Για τον έλεγχο της σύνταξης του αρχείου ρυθμίσεων χρησιμοποιούμε την εντολή: /usr/local/apache2/bin/httpd –t

Για τον έλεγχο των ρυθμίσεων του Virtual Hosts και to configuration file:

/usr/local/apache2/bin/httpd –S

Για εμφάνιση των πληροφοριών για την τρέχουσα έκδοση του apache:

/usr/local/apache2/bin/httpd –v

Για πληροφορίες εγκατάστασης:

/usr/local/apache2/bin/httpd -V

Και άλλες πολλές….

3.7.3 Apache Tomcat

Στην δική μας εφαρμογή χρησιμοποιήθηκε ο Apache Tomcat.

Ο Apache Tomcat είναι ένας server διαδικτύου και container servlet ανοικτού κώδικα που αναπτύχθηκε από την Apache Software Foundation (ASF) . Ο Tomcat εφαρμόζει αρκετές EEspecifications Java , όπως Java Servlet , JavaServer Pages (JSP) , Java EE , και WebSocket , και παρέχει μια «καθαρή Java» σε περιβάλλον server διαδικτύου για τον κώδικα της Java ώστε να τρέξει.

Ο Apache αναπτύσσεται και συντηρείται από μια ανοιχτή κοινότητα προγραμματιστών κάτω από την αιγίδα του Ιδρύματος Apache Software , και απελευθερώνεται υπό την άδεια Apache License 2.0 , και αποτελεί λογισμικό ανοιχτού κώδικα .

3.7.3.1 Components- Εξαρτήματα

Tomcat 4.x εκδόθηκε με την Catalina (ένα servlet container), τον Coyote (έναν HTTP connector) και τον Jasper (μια JSP engine).

3.7.3.1.1 Catalina

Η Catalina είναι ένα servlet container του Tomcat .Η Catalina εφαρμόζει τις προδιαγραφές της Sun Microsystems για τα servlet και τις JavaServer Pages (JSP) . Σε επίπεδο Tomcat , ένα στοιχείο Realm αντιπροσωπεύει μια «βάση δεδομένων » με ονόματα χρηστών , κωδικούς πρόσβασης , και τους ρόλους που αποδοθεί στους εν λόγω χρήστες . Διαφορετικές υλοποιήσεις Realm επιτρέπουν στην Catalina να ενσωματωθούν σε περιβάλλοντα όπου οι εν λόγω πληροφορίες ελέγχου ταυτότητας έχουν ήδη δημιουργηθεί και αναφερόμαστε σε θέματα συντήρησης

3.7.3.1.2 Coyote

Ο Coyote είναι ένα εξάρτημα σύνδεσης για τον Tomcat που υποστηρίζει το HTTP 1.1 πρωτόκολλο ως server διαδικτύου . Αυτό επιτρέπει την Καταλίνα , ονομαστικά σε ένα Java Servlet ή σε ένα JSP container , ώστε να ενεργήσει και ως ένα απλό server διαδικτύου και να εξυπηρετήσει τοπικά αρχεία και έγγραφα HTTP .

Ο Coyote ακούει σε εισερχόμενες συνδέσεις στο διακομιστή σε μια συγκεκριμένη θύρα TCP και διαβιβάζει την αίτηση στις μηχανές του Tomcat για την επεξεργασία της αίτησης και επιστρέφει μια απάντηση στο αιτούν πελάτη . Ένας άλλος τρόπος σύνδεσης είναι εκτός του Coyote Connector , είναι ο Coyote JK , που ακούει παρόμοια αλλά, αντίθετα, προωθεί τα αιτήματά του σε άλλο web server , όπως τον Apache , χρησιμοποιώντας το πρωτόκολλο JK . Αυτό συνήθως προσφέρει καλύτερη απόδοση .

3.7.3.1.3 Jasper

Ο Jasper είναι ένας JSP κινητήρα του Tomcat. Ο Jasper αναλύει τα αρχεία JSP για την κατάρτισή τους σε κώδικα Java, όπως servlets (που μπορεί να αντιμετωπιστεί από την Catalina). Κατά το χρόνο εκτέλεσης, ο Jasper ανιχνεύει τις αλλαγές στα αρχεία JSP και τους ανασύνθεση τους.

Από την έκδοση 5, Tomcat χρησιμοποιεί Jasper 2, ο οποίος είναι μια υλοποίηση του JSP με προδιαγραφές της Sun Microsystems 2.0. Από τον Jasper σε Jasper 2, σημαντικά χαρακτηριστικά προστέθηκαν:

1. Η JSP βιβλιοθήκη tag pooling - Κάθε σήμανση ετικέτας στο αρχείο JSP γίνεται από μια τάξη χρήσης ετικέτας, αντικείμενα μπορούν να συγκεντρωθούν και να επαναχρησιμοποιηθούν σε όλη την JSP Servlet.
2. Το ιστορικό της μεταγλώττισης των JSP - Ενώ με την μεταγλώττιση του τροποποιημένου κώδικα JSP Java, η παλαιότερη έκδοση είναι ακόμα διαθέσιμη για αιτήματα από τον server. Η παλαιότερη JSP Servlet διαγράφεται μόλις η νέα servlet JSP έχει τελειώσει την ανασύνθεση της.

3. Δυνατότητα ξαναμεταγλώτησης των JSP όταν περιλαμβάνονται αλλαγές σελίδας - Σελίδες μπορούν να εισαχθούν και να συμπεριληφθούν σε ένα JSP κατά το χρόνο εκτέλεσης. Η JSP δεν θα ανασυνθέτει μόνο με τις αλλαγές στο αρχείο JSP, αλλά και με τις συμπεριλαμβανόμενες αλλαγές σελίδας.
4. JDT Java compiler – Με την χρήση του Jasper 2, ένας προγραμματιστής μπορεί να χρησιμοποιήσει το Eclipse JDT (Εργαλεία Java Development) για Java compiler αντί το Ant και javac σε επίπεδο γραμμής εντολών.

Τρία νέα χαρακτηριστικά προστέθηκαν στην ανανεωμένη έκδοση του Tomcat 7:

1. Υποστήριξη Cluster: Έχει προστεθεί αυτό το συστατικό εξάρτημα ώστε να παρέχει την δυνατότητα διαχείρισης μεγάλων εφαρμογών . Χρησιμοποιείται για εξισορρόπηση φορτίου που μπορεί να επιτευχθεί μέσω πολλών τεχνικών. Υποστήριξη ομαδοποίηση απαιτείται σήμερα από την έκδοση JDK 1.5 ή οποιαδήποτε νεότερη της
2. Υποστήριξη Υψηλής διαθεσιμότητας (High availability): Έχει προστεθεί ένα χαρακτηριστικό υψηλής διαθεσιμότητας για να διευκολύνει τον προγραμματισμό των αναβαθμίσεων ενός συστήματος (π.χ. νέες κυκλοφορίες , αιτήματα αλλαγής) , χωρίς να επηρεάζεται το ζωντανό περιβάλλον . Αυτό γίνεται με την αποστολή ζωντανών αιτήματα της κυκλοφορίας σε έναν προσωρινό διακομιστή σε μια διαφορετική θύρα , ενώ ο κύριος διακομιστής έχει αναβαθμιστεί στο κεντρική θύρα . Είναι πολύ χρήσιμο για τον χειρισμό των αιτήσεων των χρηστών για διαδικτυακές εφαρμογές υψηλής κυκλοφορίας .
3. Υποστήριξη Εφαρμογών Διαδικτύου (Web Application): Επίσης, έχει προστεθεί από τον χρήστη , δυνατότητα προσθήκης υποστήριξης για την ανάπτυξη σε όλη την ποικιλία των περιβαλλόντων . Προσπαθεί επίσης να διαχειρίζεστε τα session , καθώς και εφαρμογές σε όλο το δίκτυο .

3.7.3.2 Tomcat

Ο Tomcat χτίζει πρόσθετα στοιχεία συνεχώς. Μια σειρά από πρόσθετα συστατικά μπορούν να χρησιμοποιηθούν με τον Apache Tomcat .

3.7.3.2.1 Χαρακτηριστικά

Ο Tomcat 7.x υλοποιεί τις Servlet 3.0 και JSP 2.2 προδιαγραφές . Απαιτεί την έκδοση Java 1.6 , αν και οι προηγούμενες εκδόσεις τρέχουν σε Java 1.1 με 1.5 . Στις εκδόσεις 5 έως 6 σημειώθηκαν βελτιώσεις σχετικά με την συλλογή των απορριμμάτων- garbage collection , το JSP parsing , την απόδοση και επεκτασιμότητα .

3.7.3.2.2 Ιστορικά

Ο Tomcat ξεκίνησε ως μια εφαρμογή αναφοράς servlet από τον James Duncan Davidson , ένας αρχιτέκτονας λογισμικού της Sun Microsystems . Αργότερα βοήθησε να κάνει το open source project και έπαιξε καθοριστικό ρόλο στη δωρεά του bySun Microsystems για την Apache Software Foundation . Το λογισμικό εργαλείο κατασκευής αυτοματισμού Apache Ant αναπτύχθηκε ως παρενέργεια της δημιουργίας του Tomcat ως ένα έργο ανοικτού πηγαίου κώδικα .

Ο Davidson είχε αρχικά ελπίσει ότι το έργο θα γίνει ανοικτή πηγή και , δεδομένου ότι πολλά έργα ήταν ανοικτού κώδικα , είχε και πολλά O'Reilly βιβλία που σχετίζονται με αυτά που διαθέτουν ένα ζώο στο εξώφυλλο , ήθελε να ονομάσει το έργο μετά από ένα ζώο . Ήρθε με το Tomcat αφού αιτιολογημένη το ζώο εκπροσωπείται κάτι που θα μπορούσε να τα βγάλει πέρα μόνο του . Παρά το γεγονός ότι ο γάτος ήταν ήδη σε χρήση για έναν άλλο τίτλο O'Reilly , την επιθυμία του να δει ένα ζώο τελικά έγινε πραγματικότητα όταν O'Reilly δημοσιεύονται Tomcat βιβλία τους με μια λεοπάρδαλη χιόνι στο εξώφυλλο του 2003.

Servlet API history

Servlet API version	Released	Platform	Important Changes
Servlet 3.1	May 2013	JavaEE 7	Non-blocking I/O, HTTP protocol upgrade mechanism (WebSocket)[5]
Servlet 3.0	December 2009	JavaEE 6, JavaSE 6	Pluggability, Ease of development, Async Servlet, Security, File Uploading
Servlet 2.5	September 2005	JavaEE 5, JavaSE 5	Requires JavaSE 5, supports annotation
Servlet 2.4	November 2003	J2EE 1.4, J2SE 1.3	web.xml uses XML Schema
Servlet 2.3	August 2001	J2EE 1.3, J2SE 1.2	Addition of `Filter`
Servlet 2.2	August 1999	J2EE 1.2, J2SE 1.2	Becomes part of J2EE, introduced independent web applications in .war files
Servlet 2.1	November 1998	Unspecified	First official specification, added `RequestDispatcher`, `ServletContext`
Servlet 2.0		JDK 1.1	Part of Java Servlet Development Kit 2.0
Servlet 1.0	June 1997		

3.7.3.2.3 Apache TomEE

Ο Apache TomEE (προφέρεται "Tommy") είναι η Java Enterprise Edition του Apache Tomcat (Tomcat + Java EE = TomEE) που συνδυάζει διάφορα έργα Enterprise Java, συμπεριλαμβανομένων Apache OpenEJB, Apache OpenWebBeans, Apache OpenJPA, Apache MyFaces και άλλα. [16] Τον Οκτώβριο του 2011, το έργο έλαβε πιστοποίηση από την Oracle Corporation ως μια συμβατή εφαρμογή της Java EE 6 Web Profile.

3.8 Hibernate Framework

Το Hibernate Framework είναι λογισμικό ανοιχτού κώδικα (ελεύθερο λογισμικό) που σκοπό έχει να συνδέσει τα αντικείμενα που δημιουργούνται σε μια αντικειμενοστραφή γλώσσα προγραμματισμού (Java) με τους πίνακες μιας σχεσιακής βάσης δεδομένων. Η σύνδεση αυτή επιτυγχάνεται με την χρήση επιπρόσθετης πληροφορίας (metadata) που τοποθετείται κατάλληλα (μαζί με τον κώδικα Java ή σε ξεχωριστά xml αρχεία) και περιγράφει την αντιστοιχία μεταξύ των αντικειμένων και της βάσης δεδομένων. Γενικά το Hibernate προσφέρει την αυτόματη μετατροπή της μιας μορφής (αντικείμενα) στην άλλη (σχεσιακή βάση δεδομένων).

Το Hibernate συγκαταλέγεται στην κατηγορία του λογισμικού ORM (*object/relational mapping*). Το λογισμικό ORM στοχεύει στην δημιουργία μιας διεπαφής (*interface*) μεταξύ των διαδεδομένων σχεσιακών βάσεων δεδομένων και του αντικειμενοστραφούς προγραμματισμού. Με απλά λόγια, προσφέρει την χρησιμοποίηση μιας σχεσιακής βάσης δεδομένων σαν να ήταν αντικειμενοστραφής. Για να το επιτύχει αυτό δημιουργεί αντιστοιχίες μεταξύ των εννοιών του αντικειμενοστραφούς προγραμματισμού (συσχετίσεις, κληρονομικότητα, πολυμορφισμός) - που δεν υπάρχουν σε μια σχεσιακή βάση δεδομένων - και των πινάκων και σχέσεων μεταξύ των πινάκων μιας σχεσιακής βάσης. Με αυτό τον τρόπο ο προγραμματιστής *βλέπει* τελικά μια αντικειμενοστραφή βάση δεδομένων, παρόλο που στην ουσία χρησιμοποιεί μια σχεσιακή. Έτσι ο προγραμματιστής χρησιμοποιεί τα αντικείμενα της συγκεκριμένης εφαρμογής, τα τροποποιεί σχετικά με τη λογική της εφαρμογής που αναπτύσσει και τα αποθηκεύει (τροποποιεί, διαγράφει και αναζητά) στην βάση *ως αντικείμενα*, σκεπτόμενος δηλαδή με αντικειμενοστραφείς έννοιες και όχι με βάση το σχήμα της σχεσιακής βάσης δεδομένων. Σε αυτό το σημείο είναι το Hibernate που, γνωρίζοντας την αντιστοιχία μεταξύ βάσης και λογικής της εφαρμογής, αναλαμβάνει να κατασκευάσει την κατάλληλη εντολή της SQL η οποία και στέλνεται τελικά στην βάση

δεδομένων. Έπειτα, τα αποτελέσματα που επιστρέφει η βάση το Hibernate τα επιστρέφει στον προγραμματιστή ως αντικείμενα της εφαρμογής. Είναι δηλαδή ένα ενδιάμεσο επίπεδο μεταξύ της εφαρμογής και της βάσης δεδομένων.

3.8.1 Πλεονεκτήματα και Μειονεκτήματα

Το Hibernate προσφέρει τα παρακάτω στον προγραμματιστή:

- *Παραγωγικότητα*: Στην ανάπτυξη λογισμικού ένα μεγάλο μέρος της προγραμματιστικής προσπάθειας αφιερώνεται στην διεπαφή της εφαρμογής με τη βάση δεδομένων. Το Hibernate αυτοματοποιώντας τις βασικές λειτουργίες Δημιουργία/Ανάγνωση/Τροποποίηση/Διαγραφή (CRUD – Create Read Update Delete) επιτρέπει αρχικά στον προγραμματιστή να επικεντρώνει την προσπάθειά του στη λογική της εφαρμογής (business logic). Επίσης, υπάρχει η δυνατότητα να ακολουθηθούν δύο στρατηγικές ανάπτυξης λογισμικού: είτε αρχίζοντας από το μοντέλο δεδομένων είτε από τη βάση δεδομένων. Αυτό μειώνει σε μεγάλο βαθμό το χρόνο ανάπτυξης.

- *Συντηρησιμότητα*: Με τη χρήση του Hibernate γράφονται σημαντικά λιγότερες γραμμές κώδικα και ο κώδικας είναι πιο κατανοητός και καλογραμμένος. Αυτό κάνει την συντήρηση της εφαρμογής ευκολότερη.

- *Ανεξαρτησία από τη βάση δεδομένων*: Με τη συμβατότητα του Hibernate με διαφορετικές βάσεις δεδομένων και τη δυνατότητα σύνδεσής του με τη βάση μέσω δηλώσεων οριζόμενων σε ειδικό αρχείο η αναπτυσσόμενη εφαρμογή μπορεί με ελάχιστες τροποποιήσεις να χρησιμοποιηθεί με βάσεις δεδομένων διαφορετικών κατασκευαστών. Το γεγονός αυτό στερεί μεν από το Hibernate την εκμετάλλευση των ιδιαίτερων χαρακτηριστικών της χρησιμοποιούμενης βάσης, όμως, και σε αυτή την περίπτωση, δίνεται η δυνατότητα χρήσης πηγαίας SQL μέσα στο Hibernate που εκμεταλλεύεται τα ιδιαίτερα αυτά χαρακτηριστικά. Αυτό βέβαια μειώνει την ανεξαρτησία του Hibernate.

3.9 AngularJS

AngularJS (που συνήθως αναφέρονται ως "γωνιακή" ή "Angular.js") είναι ένα πλαίσιο web εφαρμογή ανοικτού κώδικα διατηρείται κυρίως από την Google και από την κοινότητα των ατόμων και των επιχειρήσεων για την αντιμετώπιση πολλών από τις προκλήσεις που ανέκυψαν κατά την ανάπτυξη εφαρμογών μίας σελίδας. Έχει ως στόχο να απλοποιήσει τόσο την ανάπτυξη και τη δοκιμή αυτών των εφαρμογών με την παροχή ενός πλαισίου για client-side model-View-Controller (MVC) και το model-view-viewmodel (MVVM) αρχιτεκτονικές, μαζί με τα συστατικά που χρησιμοποιούνται συνήθως σε πλούσιες εφαρμογές Διαδικτύου.

Το πλαίσιο AngularJS λειτουργεί με την πρώτη ανάγνωση της σελίδας HTML, η οποία έχει ενσωματώσει σε αυτό πρόσθετα χαρακτηριστικά προσαρμοσμένα tag. Ερμηνεύει αυτά τα χαρακτηριστικά, όπως οδηγίες για να δεσμεύουν είσοδο ή έξοδο μέρη της σελίδας σε ένα μοντέλο που αντιπροσωπεύεται από το πρότυπο μεταβλητές JavaScript.

Σύμφωνα με την Javascript υπηρεσία analytics Libscore, το AngularJS χρησιμοποιείται στις ιστοσελίδες των Wolfram, NBC, Walgreens, Intel, Sprint, ABC News, και περίπου 8.400 άλλων δικτυακών τόπων από 1 εκατομμύρια δοκιμασμένα από τον Ιούλιο του 2015.

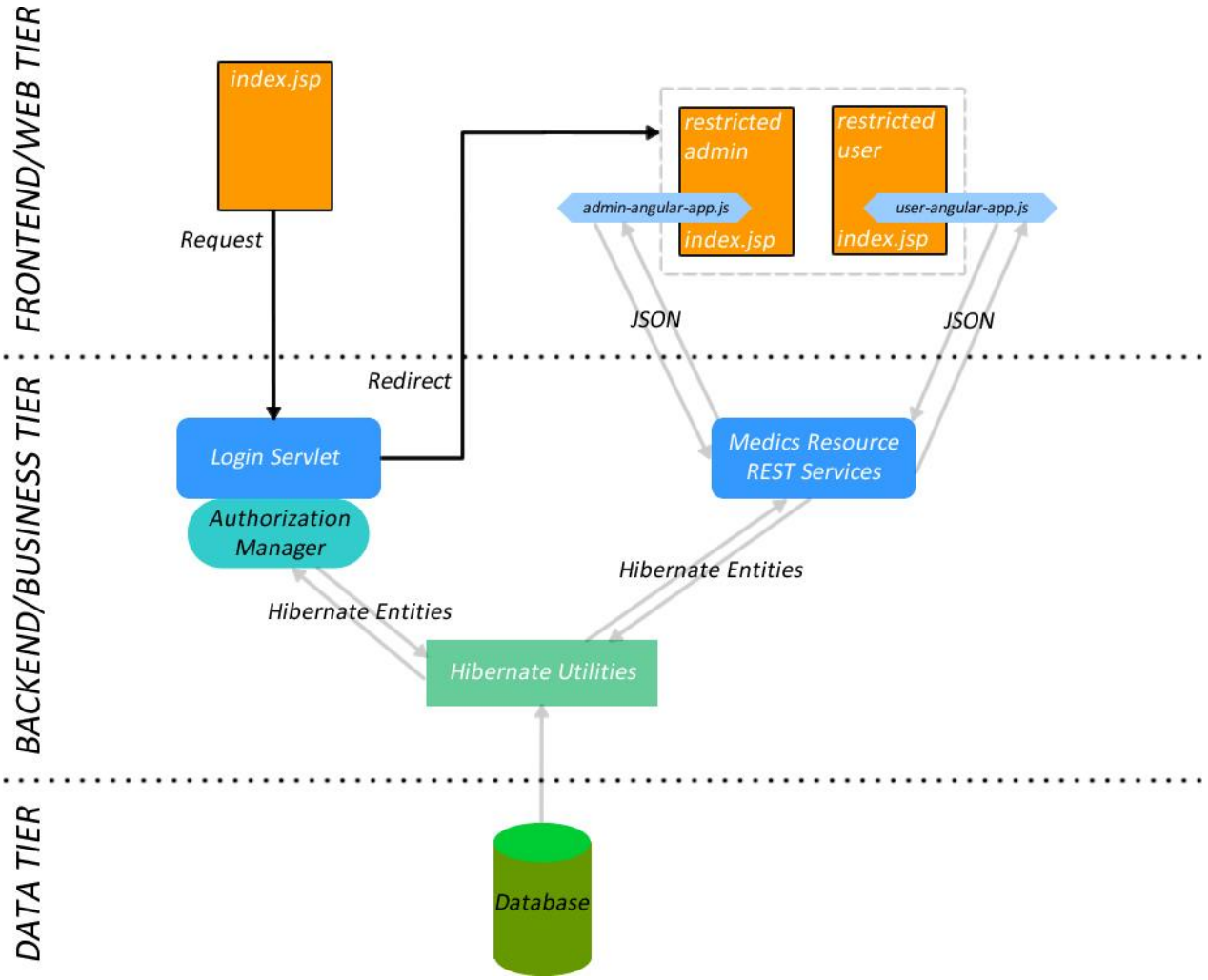
AngularJS είναι το μέρος frontend μαζί με Node.js σε χρόνο εκτέλεσης με Express.js πλαίσιο backend και τη βάση δεδομένων MongoDB.

4 Υλοποίηση Εφαρμογής

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκαν οι παρακάτω τεχνολογίες:

- Περιβάλλον εκτέλεσης εφαρμογής: JAVA JRE, Έκδοση 8
  - Χρήση Servlets και JSP
- Βάση Δεδομένων: MySQL Database, Έκδοση 5.6
- Αντιστοίχηση Πινάκων Σχεσιακής Βάσης Δεδομένων σε Java Objects (ORM): Hibernate Core, Έκδοση 3.6
- Υλοποίηση Διαχείρισης αυθεντικοποίησης χρήστη και ρόλων: Spring Components (web, context) 4.2.4
- Web Services: RESTFull WS με αντιστοίχηση JSON σε POJOs: Jersey, Έκδοση 1.19
- Javascript Framework: angular.js, Έκδοση 1.4.8 (με χρήση plugin angular-date-picker.js για επιλογή ημερομηνίας)
- WEB Σελίδες: HTML, CSS

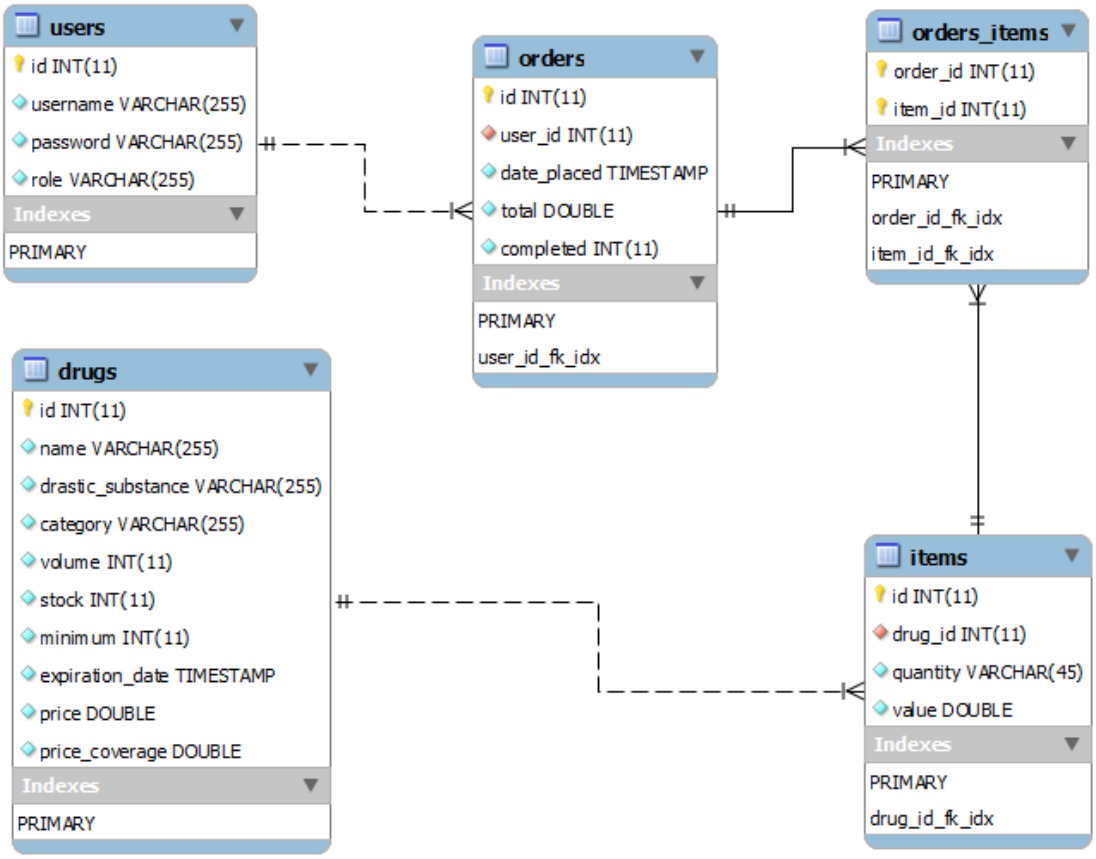
Παρακάτω ακολουθεί ένα σχήμα της εφαρμογής και πως γίνεται η αλληλεπίδραση μεταξύ των κομματιών, καθώς και τα περιεχόμενα κάθε επιπέδου:



Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε το Eclipse για Java EE, έκδοση Kepler.
Για το deployment της εφαρμογής έγινε χρήση του Maven.
Η εφαρμογή είναι προσανατολισμένη για χρήση σε Apache Tomcat 8

4.1 Βάση Δεδομένων

Για τις ανάγκες της εργασίας, δημιουργούνται και χρησιμοποιούνται πέντε πίνακες και σχετίζονται μεταξύ τους όπως φαίνεται στο παρακάτω σχήμα:



4.1.1 Οι πίνακες της βάσης

Ο πίνακας users αποθηκεύει τους χρήστες της εφαρμογής και το ρόλο τους. Για λόγους ασφάλειας, διατηρείται ο κωδικός πρόσβασης των χρηστών σε μορφή hex hash με χρήση του αλγόριθμου MD5.

Ο πίνακας drugs περιέχει πληροφορίες όπως όνομα, δραστική ουσία κλπ., αλλά και ποσοτικά χαρακτηριστικά για τα φάρμακα όπως απόθεμα (stock), ελάχιστο απόθεμα (minimum), τιμή (price) και κάλυψη τιμής(price coverage).

Ο πίνακας orders κρατάει γενικά στοιχεία για τις παραγγελίες όπως ημερομηνία πραγματοποίησης (date_placed), γενικό σύνολο (total) και κατάσταση ολοκλήρωσης (completed).

Ο πίνακας items περιέχει στοιχεία για τις λεπτομέρειες της παραγγελίας, όπως αριθμός τεμαχίων (quantity) του φαρμάκου που παραγγέλθηκε και την τιμή τους(value).

Ο πίνακας orders_items ορίζει τη σχέση μεταξύ παραγγελιών και τεμαχίων.

4.1.2 Σχέσεις μεταξύ των πινάκων

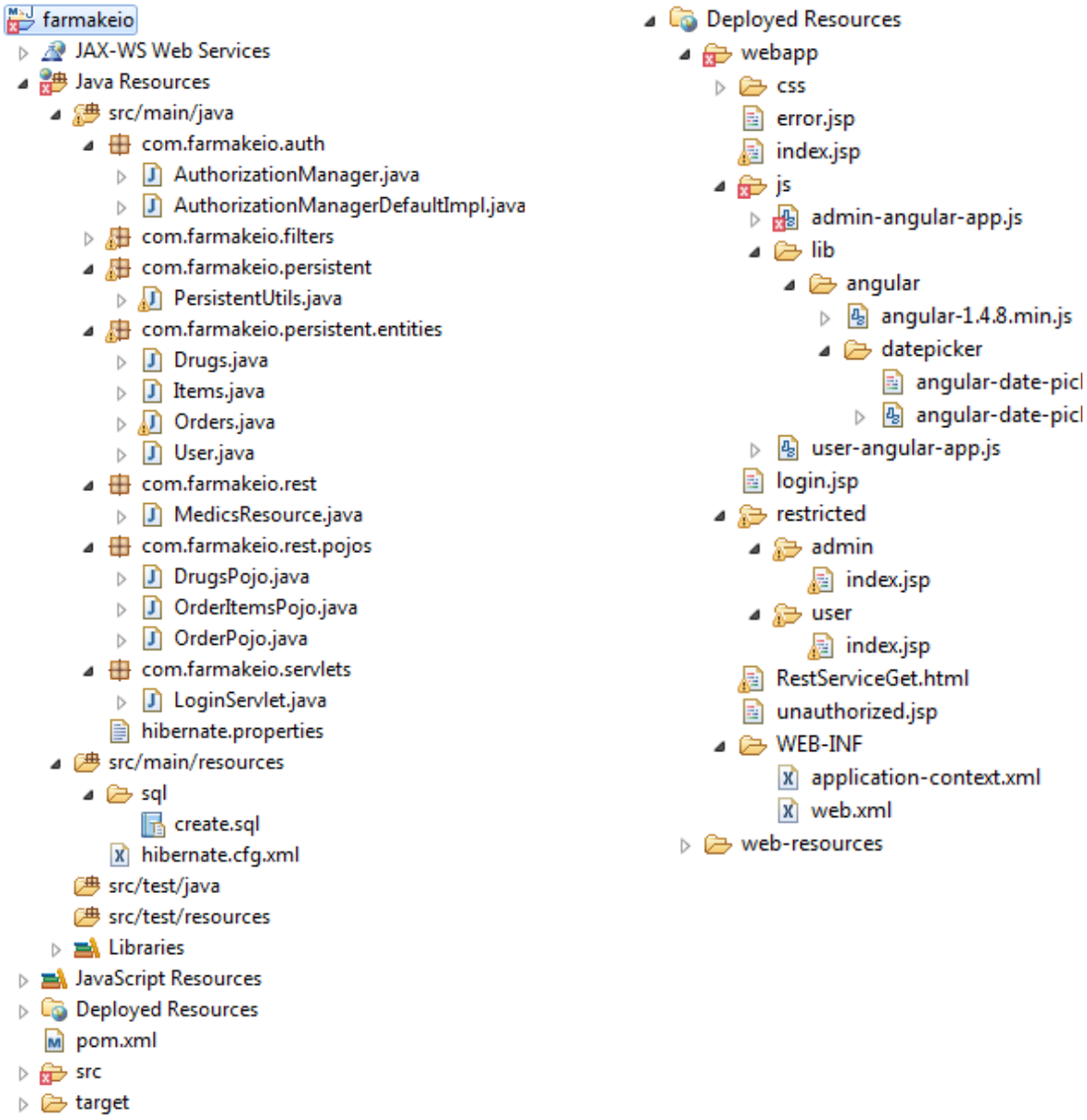
Ο πίνακας orders σχετίζεται με τον πίνακα users μέσω ενός foreign key που αναφέρεται στο id της εγγραφής του user, και έτσι έχουμε την πληροφορία για το ποιος χρήστης έκανε την παραγγελία.

Ο πίνακας items σχετίζεται με τον πίνακα drugs μέσω ενός foreign key που αναφέρεται στο id της εγγραφής του drugs, και έτσι έχουμε την πληροφορία για το ποιο(α) φάρμακο(α) έχει(έχουν) παραγγελθεί.

Τέλος, μέσω του πίνακα orders_items που αποτελείται από δυο πεδία που αναφέρονται στο id της εγγραφής του orders και στο id της εγγραφής του items μπορούμε να συσχετίσουμε ποια τεμάχια παραγγελίας αντιστοιχούνε σε ποια παραγγελία και αντιστρόφως.

4.2 Δομή αρχείων eclipse project

Η δομή των αρχείων φαίνεται παρακάτω. Στο αριστερό κομμάτι φαίνεται η οργάνωση της java packages και στο δεξιό η οργάνωση των αρχείων του web application και του frontend



4.3 Παραμετροποίηση / Εκτέλεση Εφαρμογής

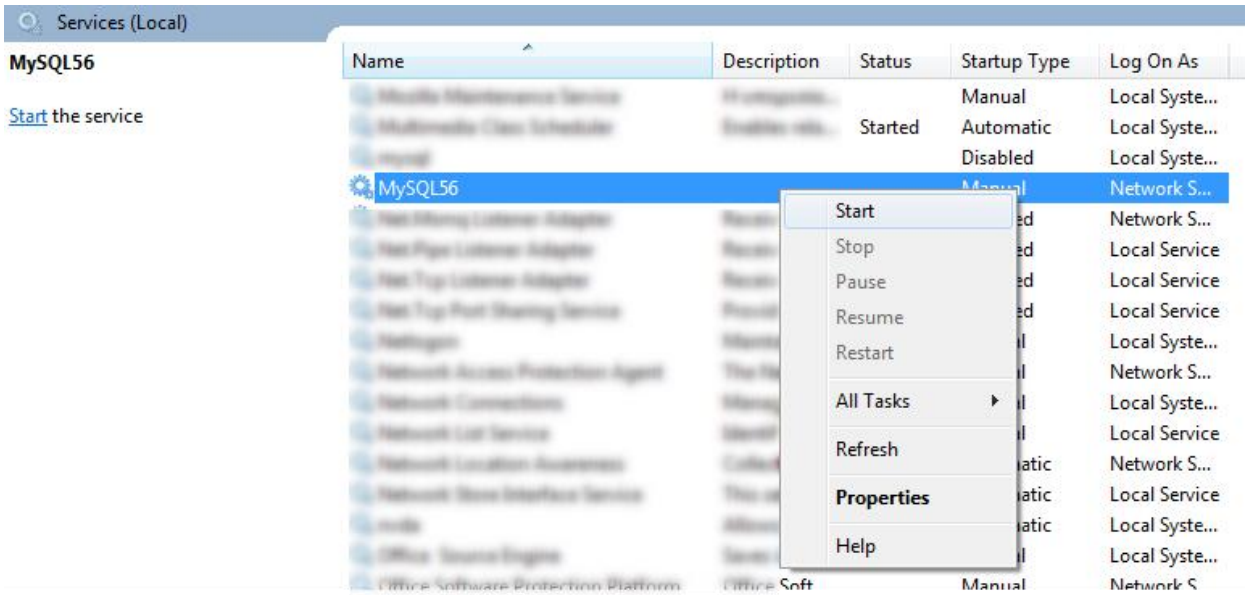
Η εφαρμογή είναι συμπτυγμένη στο αρχείο farmakeio-0.0.1-SNAPSHOT.war

4.3.1 MySQL

Για τη χρήση της εφαρμογής, πρέπει να έχουμε εγκαταστήσει τη βάση MySQL 5.6. Την παραμετροποιούμε ώστε να τρέχει στην πόρτα 3306 και δημιουργούμε μια βάση δεδομένων με το όνομα farmakeio_db. Ορίζουμε τον χρήστη root με κωδικό root123 να έχει πρόσβαση στη βάση μας, και μετά εκτελούμε τα SQL queries που βρίσκονται στο αρχείο create.sql με τα οποία δημιουργούνται οι πίνακες καθώς και:

- 3 χρήστες
    - 1 διαχειριστής (admin1/admin1pass)
    - 2 απλοί χρήστες (user1/user1pass, user2/user2pass)
- 15 ενδεικτικές εγγραφές φαρμάκων

Εφόσον έχουμε εγκαταστήσει/ παραμετροποιησει την βάση μας και εκτελέσαμε και τα SQL queries, σε περιβάλλον Windows θα πρέπει να εκκινήσουμε το service της MySQL όπως φαίνεται στην εικόνα:

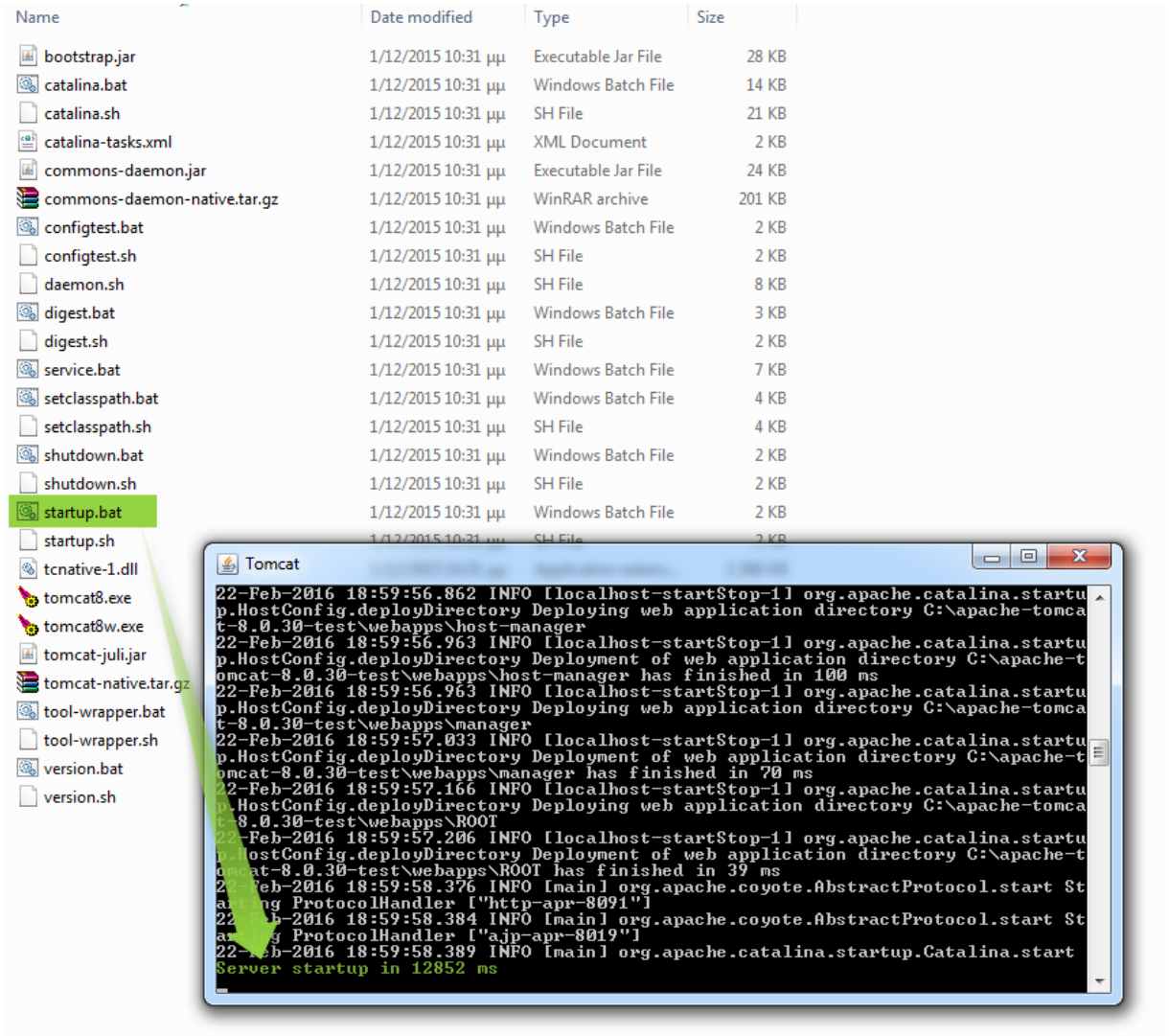


Περισσότερα για την παραμετροποιηση της MySQL καθώς και για εκτέλεση σε περιβάλλον linux: https://dev.mysql.com/doc/

4.3.2 Tomcat

Για την εκτέλεση του αρχείου της εφαρμογής κάνουμε χρήση του Apache Tomcat 8.0.30 (https://tomcat.apache.org/download-80.cgi). Για λόγους σύμβασης, θα αναφερόμαστε στο directory του tomcat installation με το όνομα <tomcat8_base>.

Μετονομάζουμε το αρχείο “farmakeio-0.0.1-SNAPSHOT.war” σε “farmakeio.war” και το τοποθετούμε στο φάκελο <tomcat8_base>/webapps. Κατόπιν πηγαίνουμε στο φάκελο <tomcat8_base>/bin και εκτελούμε το αρχείο “startup.bat” (“startup.sh” για περιβάλλον linux) και αφού εκτελεστεί σωστά θα δούμε το παρακάτω μήνυμα στο παράθυρο εκτέλεσης:


Name
Date modified
Type
Size
bootstrap.jar 1/12/2015 10:31 μμ Executable Jar File 28 KB
catalina.bat 1/12/2015 10:31 μμ Windows Batch File 14 KB
catalina.sh 1/12/2015 10:31 μμ SH File 21 KB
catalina-tasks.xml 1/12/2015 10:31 μμ XML Document 2 KB
commons-daemon.jar 1/12/2015 10:31 μμ Executable Jar File 24 KB
commons-daemon-native.tar.gz 1/12/2015 10:31 μμ WinRAR archive 201 KB
configtest.bat 1/12/2015 10:31 μμ Windows Batch File 2 KB
configtest.sh 1/12/2015 10:31 μμ SH File 2 KB
daemon.sh 1/12/2015 10:31 μμ SH File 8 KB
digest.bat 1/12/2015 10:31 μμ Windows Batch File 3 KB
digest.sh 1/12/2015 10:31 μμ SH File 2 KB
service.bat 1/12/2015 10:31 μμ Windows Batch File 7 KB
setclasspath.bat 1/12/2015 10:31 μμ Windows Batch File 4 KB
setclasspath.sh 1/12/2015 10:31 μμ SH File 4 KB
shutdown.bat 1/12/2015 10:31 μμ Windows Batch File 2 KB
shutdown.sh 1/12/2015 10:31 μμ SH File 2 KB
startup.bat 1/12/2015 10:31 μμ Windows Batch File 2 KB
startup.sh
tcnative-1.dll
tomcat8.exe
tomcat8w.exe
tomcat-juli.jar
tomcat-native.tar.gz
tool-wrapper.bat
tool-wrapper.sh
version.bat
version.sh
Tomcat
22-Feb-2016 18:59:56.862 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory C:\apache-tomcat-8.0.30-test\webapps\host-manager
22-Feb-2016 18:59:56.963 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory C:\apache-tomcat-8.0.30-test\webapps\host-manager has finished in 100 ms
22-Feb-2016 18:59:56.963 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory C:\apache-tomcat-8.0.30-test\webapps\manager
22-Feb-2016 18:59:57.033 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory C:\apache-tomcat-8.0.30-test\webapps\manager has finished in 70 ms
22-Feb-2016 18:59:57.166 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory C:\apache-tomcat-8.0.30-test\webapps\ROOT
22-Feb-2016 18:59:57.206 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory C:\apache-tomcat-8.0.30-test\webapps\ROOT has finished in 39 ms
22-Feb-2016 18:59:58.376 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-apr-8091"]
22-Feb-2016 18:59:58.384 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["ajp-apr-8019"]
22-Feb-2016 18:59:58.389 INFO [main] org.apache.catalina.startup.Catalina.start
Server startup in 12852 ms

5 Λειτουργείες εφαρμογής

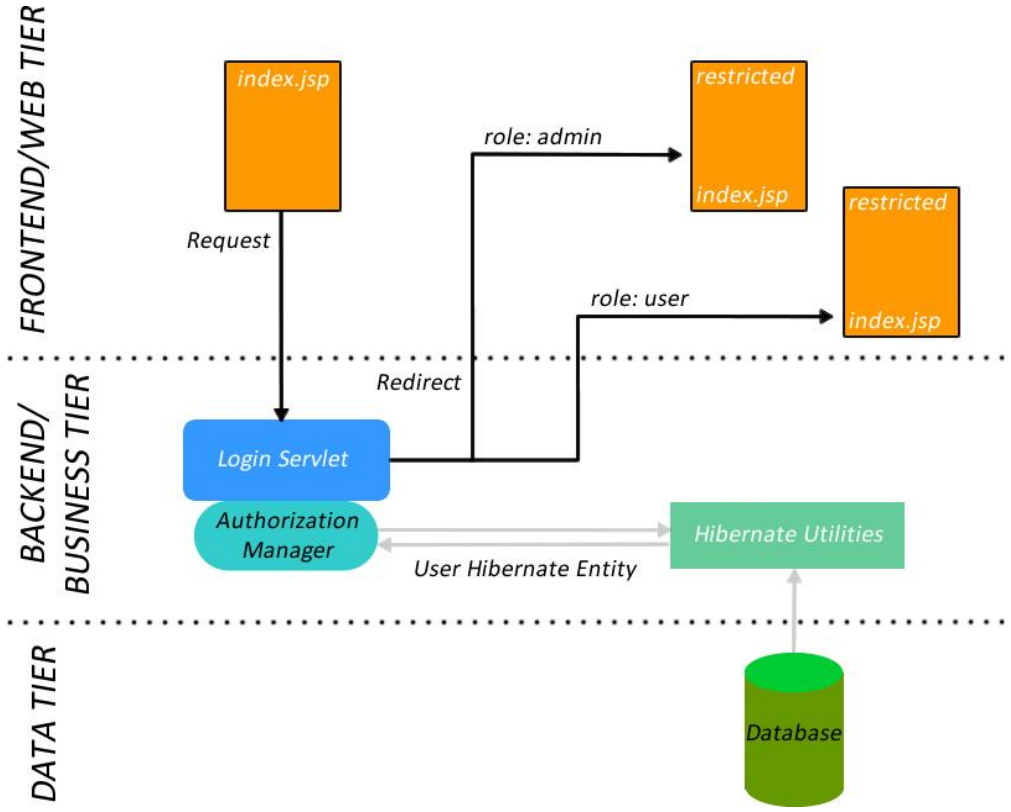
Το ηλεκτρονικό φαρμακείο είναι ένα πληροφοριακό σύστημα τόσο εσωτερικής αλλά και εξωτερικής διαχείρισης πώρων. Παρέχει δυνατότητες και λειτουργίες που περιγράφονται παρακάτω αναλυτικά αλλά θα κάνουμε μια μικρή αναφορά.

Έπειτα από τον απαραίτητο σχεδιασμό βάσης που αναφέραμε αναλυτικά και στο παραπάνω κεφάλαιο, οι λειτουργίες που διεκπεραιώνονται στο σύστημα μας είναι:

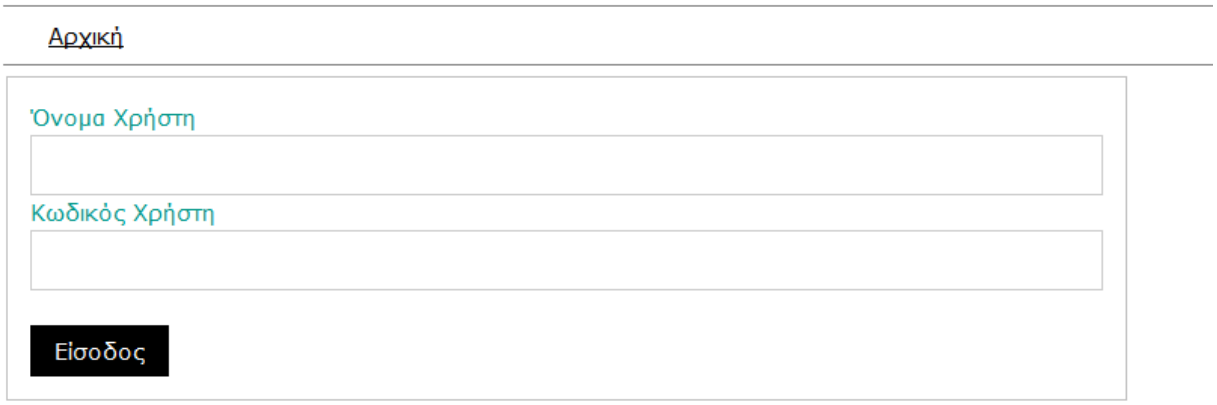
- Γνωστοποίηση χρήστη, απλού είτε διαχειριστή. Κάθε χρήστης έχει το δικό του προφίλ και τις δικές του λειτουργίες που μπορεί να διεκπεραίωση.
- Λειτουργίες Διαχειριστή:
  - Εισαγωγή / Διαγραφή / Επεξεργασία /Διαχείριση Αποθεμάτων – Διαθεσιμότητα / Προβολή φαρμάκων
  - Διαχείριση Παραγγελιών
- Λειτουργίες Χρήστη:
  - Προβολή διαθέσιμων Φαρμάκων / Προθήκη σε καλάθι
  - Διαχείριση καλαθιού/ Προθήκη και διαγραφή από αυτό
  - Παραγγελία φαρμάκων
  - Προβολή Παραγγελιών

5.1 Authentication

Η αυθεντικοποίηση χρήστη γίνεται με τη χρήση μιας υλοποίησης Authentication Manager ο οποίος είναι υπεύθυνος να ελέγξει την εγκυρότητα της ταυτότητας του χρήστη και να τον ανακατευθύνει στην κατάλληλη σελίδα ανάλογα με το ρόλο που έχει. Η διαδικασία φαίνεται σχηματικά στην εικόνα παρακάτω:



Αρχικά, εμφανίζεται η εικόνα με τη φόρμα εσόδου στην εφαρμογή.



Στη συνέχεια γίνεται ένα request στο LoginServlet με παραμέτρους τα στοιχεία που εισήγαγε ο χρήστης καθώς και το είδος του action (login) και εκτελείται η μέθοδος loginUser() του servlet.

```
private void loginUser(HttpServletRequest request, HttpServletResponse response, String user, String pass){

    HttpSession session = request.getSession();

    String path = "/index.jsp";

    try {

        User current = PersistentUtils.authenticateUser(user, pass);
        if(current != null){
            session.setAttribute("logged", current);

           //Obtain AuthorizationManager singleton from Spring ApplicationContext
           ApplicationContext ctx = WebApplicationContextUtils.getWebApplicationContext(session.getServletContext());
           AuthorizationManager authMgr = (AuthorizationManager)ctx.getBean("AuthorizationManager");

           if(authMgr.getPathMappingRole(current.getRole()) == null)
               path = request.getServletContext().getContextPath() + "/error.jsp";
           else
               path = request.getServletContext().getContextPath() + authMgr.getPathMappingRole(current.getRole()) + "index.jsp";


               request.setCharacterEncoding("UTF-8");
               request.setAttribute("logged", current.getUsername());
               response.sendRedirect(path);
//             request.getRequestDispatcher(path).include(request, response);


        }else{
            request.setAttribute("error-msg", "Authentication Failed, check username and password");
            response.sendRedirect( request.getServletContext().getContextPath() + "/index.jsp");
        }

    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Αυτή αρχικά χρησιμοποιεί τη μέθοδο authenticateUser() της κλάσης PersistentUtils που περιέχει μεθόδους που σχετίζονται με το Hibernate και την επικοινωνία με τη βάση.

```
public static User authenticateUser(String username, String password){

    SessionFactory factory = createFactory();
    // Get current session
    Session qSession = factory.getCurrentSession();

    // Begin transaction
    qSession.getTransaction().begin();
    String hashedPassword = strToMD5Digest(password);

    org.hibernate.Query query =
            qSession.createQuery("from User where username=:username and password=:password")
                    .setString("username", username)
                    .setString("password", hashedPassword);

    User user = null;
    if(query.list().size() == 1)
        user = (User) query.list().get(0);

    // Commit transaction
    qSession.getTransaction().commit();

    return user;
}
```

Η authenticateUser() δημιουργεί ένα session factory το οποίο είναι αυτό που αναλαμβάνει το transaction και την εκτέλεση του query στη βάση που θα αναζητήσει τα στοιχεία του χρήστη. Να σημειώσουμε ότι το password των χρηστών επειδή είναι αποθηκευμένο hashed όπως προείπαμε, γίνεται πρώτα ο υπολογισμός του hash με τη μέθοδο strToMD5Digest() και μετά συγκρίνουμε αυτή την τιμή με αυτή που είναι αποθυκευμένη στη βάση.

Εφόσον έχουμε επιτυχή ταυτοποίηση από τη βάση, γίνεται ανάκτηση των τιμών και το mapping τω αποτελεσμάτων σε ένα Users entity object. Αυτό το object θα αποθηκευτεί στο session της εφαρμογής για να χρησιμοποιηθεί μελλοντικά από την εφαρμογή για επιβεβαίωση ότι ο χρήστης είναι logged in, καθώς και από το μηχανισμό των παραγγελιών.

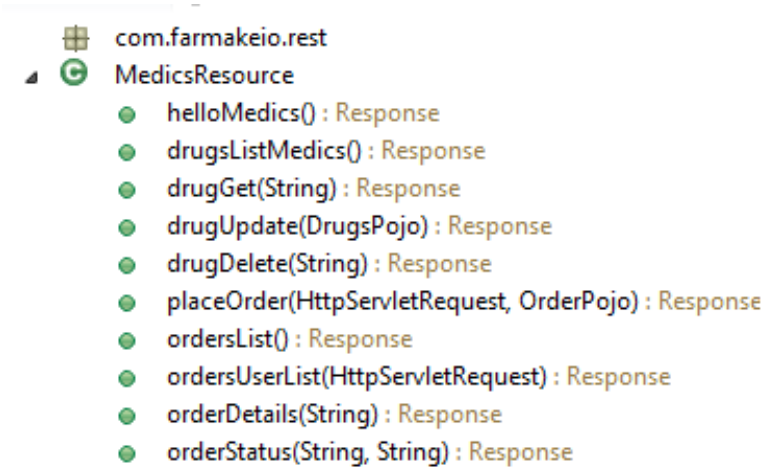
Στη συνέχεια, παίρνει το web application context το από session έτσι ώστε να πάρει το implementation του interface AuthorizationManager (αυτή η διαδικασία αναλαμβάνεται από το spring framework). Στην εφαρμογή μας η υλοποίηση του interface AuthorizationManager που χρησιμοποιούμε γίνεται από την κλάση AuthorizationManagerDefaultImpl. Εκεί αποθηκεύουμε ένα Map<String,String> roleMappings, που περιέχει key-value ζεύγη με κλειδί το όνομα του ρόλου και value το path της αυθεντικοποιοημένης σελίδας.

Μετά ο Authorization Manager καλεί τη μέθοδο getPathMappingRole() που είναι υπεύθυνη να ψάξει στο map roleMappings το path της αυθεντικοποιοημένης σελίδας που σχετίζεται με τον ρόλο του χρήστη.

Τέλος ανακατευθύνει το χρήστη στη σωστή σελίδα.

5.1.1 Rest Services

Η υλοποίηση των λειτουργιών του διαχειριστή και του χρήστη γίνεται μέσω requests του frontend με τη βοήθεια του angular.js σε Rest Endpoints της κλάσης MedicsResource. Η παρακάτω λίστα μεθόδων της κλάσης αναλαμβάνει να διεκπεραίωση τις αίτησης.



drugsListMedics() : Ανάκτηση της λίστας όλων των φαρμάκων (Διαχειριστής, Χρήστης)

drugGet(id) : Ανάκτηση του φαρμάκου δεδομένου id (Διαχειριστής)

drugUpdate(drug) : Εισαγωγή ή επεξεργασία φαρμάκου (Διαχειριστής)

drugDelete() : Διαγραφή φαρμάκου (Διαχειριστής)

placeOrder(order) : Πραγματοποίηση παραγγελίας (Χρήστης)

ordersList(): Ανάκτηση της λίστας όλων των παραγγελιών και των τεμαχίων τους (Διαχειριστής)

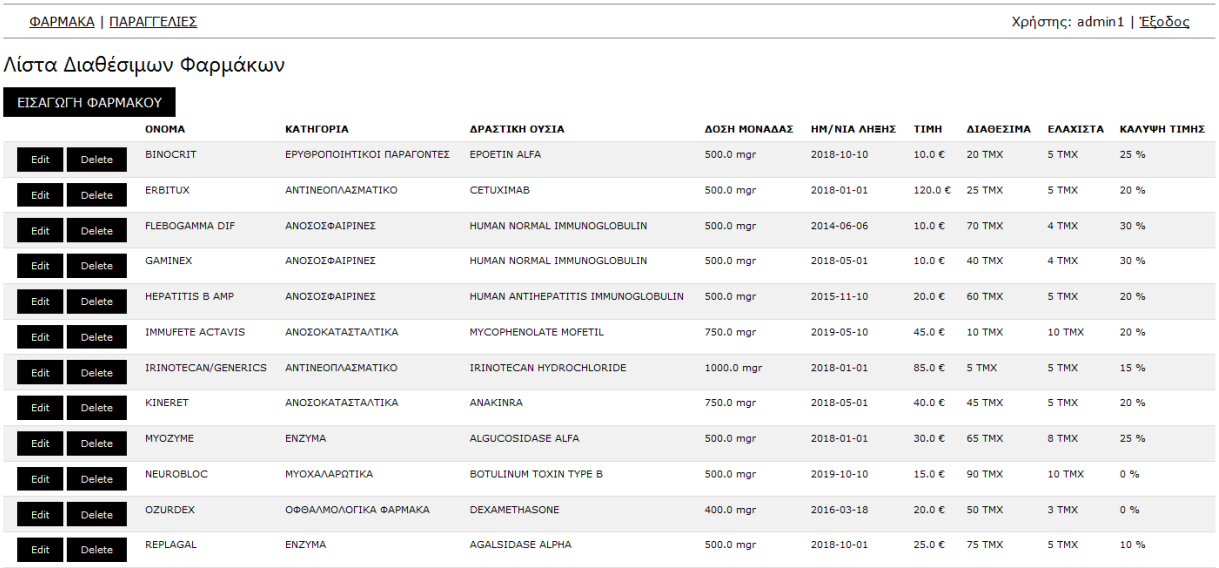
ordersUserList(): Ανάκτηση της λίστας όλων των παραγγελιών του χρήστη (Χρήστης)

orderDetails(id): Ανάκτηση της λίστας των τεμαχίων παραγγελίας δεδομένου Id (Χρήστης)

orderStatus(status): Χαρακτηρισμός κατάστασης παραγγελίας (Διαχειριστής)

5.2 Διαχειριστής / Ενέργειες

Ο διαχειριστής αφού κάνει login βλέπει την παρακάτω σελίδα που περιέχει τη λίστα όλων των φαρμάκων:



	ΟΝΟΜΑ	ΚΑΤΗΓΟΡΙΑ	ΔΡΑΣΤΙΚΗ ΟΥΣΙΑ	ΔΟΣΗ ΜΟΝΑΔΑΣ	ΗΜ/ΝΙΑ ΛΗΞΗΣ	ΤΙΜΗ	ΔΙΑΘΕΣΙΜΑ	ΕΛΑΧΙΣΤΑ	ΚΑΛΥΨΗ ΤΙΜΗΣ
Edit Delete	BINOCRIT	ΕΡΥΘΡΟΠΟΙΗΤΙΚΟΙ ΠΑΡΑΓΟΝΤΕΣ	EPOETIN ALFA	500.0 mgr	2018-10-10	10.0 €	20 TMX	5 TMX	25 %
Edit Delete	ERBITUX	ΑΝΤΙΝΕΟΠΛΑΣΜΑΤΙΚΟ	CETUXIMAB	500.0 mgr	2018-01-01	120.0 €	25 TMX	5 TMX	20 %
Edit Delete	FLEBOGAMMA DIF	ΑΝΟΣΟΣΦΑΙΡΙΝΕΣ	HUMAN NORMAL IMMUNOGLOBULIN	500.0 mgr	2014-06-06	10.0 €	70 TMX	4 TMX	30 %
Edit Delete	GAMINEX	ΑΝΟΣΟΣΦΑΙΡΙΝΕΣ	HUMAN NORMAL IMMUNOGLOBULIN	500.0 mgr	2018-05-01	10.0 €	40 TMX	4 TMX	30 %
Edit Delete	HEPATITIS B AMP	ΑΝΟΣΟΣΦΑΙΡΙΝΕΣ	HUMAN ANTIHEPATITIS IMMUNOGLOBULIN	500.0 mgr	2015-11-10	20.0 €	60 TMX	5 TMX	20 %
Edit Delete	IMMUFETE ACTAVIS	ΑΝΟΣΟΚΑΤΑΣΤΑΛΤΙΚΑ	MYCOPHENOLATE MOFETIL	750.0 mgr	2019-05-10	45.0 €	10 TMX	10 TMX	20 %
Edit Delete	IRINOTECAN/GENERICS	ΑΝΤΙΝΕΟΠΛΑΣΜΑΤΙΚΟ	IRINOTECAN HYDROCHLORIDE	1000.0 mgr	2018-01-01	85.0 €	5 TMX	5 TMX	15 %
Edit Delete	KINERET	ΑΝΟΣΟΚΑΤΑΣΤΑΛΤΙΚΑ	ANAKINRA	750.0 mgr	2018-05-01	40.0 €	45 TMX	5 TMX	20 %
Edit Delete	MYOZYME	ENZYMA	ALGUCOSIDASE ALFA	500.0 mgr	2018-01-01	30.0 €	65 TMX	8 TMX	25 %
Edit Delete	NEUROBLOC	ΜΥΟΧΑΛΑΡΩΤΙΚΑ	BOTULINUM TOXIN TYPE B	500.0 mgr	2019-10-10	15.0 €	90 TMX	10 TMX	0 %
Edit Delete	OZURDEX	ΟΦΘΑΛΜΟΛΟΓΙΚΑ ΦΑΡΜΑΚΑ	DEXAMETHASONE	400.0 mgr	2016-03-18	20.0 €	50 TMX	3 TMX	0 %
Edit Delete	REPLAGAL	ENZYMA	AGALSIDASE ALPHA	500.0 mgr	2018-10-01	25.0 €	75 TMX	5 TMX	10 %

Για την ανάκτηση αυτής της λίστας γίνεται η κλήση στο rest path /drugs/list μέσω της συνάρτησης του angular controller toy application module 'adminApp' που περιγράφεται στο αρχείο admin-angular-app.js

```
var baseUrl = window.location.protocol + "//" + window.location.hostname + ":" + window.location.port;

var app = angular.module('adminApp', [ 'mp.datePicker' ]);

app.controller('drugsCtrl', function($scope, $http) {

    $scope.showDrugsList = function(){
        $scope.hideform = true;
        $scope.hideOrders = true;
        $scope.hidelist = false;

        $scope.getDrugs();
    },

    $scope.getDrugs = function() {
        $http.get(baseUrl+"/farmakeio/rest/medics/drugs/list", {"responseType" : "json"}).then(function(response) {
            $scope.drugs = response.data.drugs;
        });
    },
```

Στην κλάση MedicsResource η μέθοδος drugsListMedics() αναλαμβάνει να επιστρέψει ένα JSON Object με τη λίστα των χαρακτηριστικών των φαρμάκων. Στην μέθοδο επίσης διακρίνουμε τη χρήση lamda expression για το iteration της λίστας των Drugs entities, κάτι που αποτελεί στοιχείο της Java 8.

```
@GET
@Path("/drugs/list")
@Produces(MediaType.APPLICATION_JSON+";charset=utf-8")
public Response drugsListMedics(){

    List<Drugs> list = Drugs.getList();

    StringBuilder sb = new StringBuilder("{ \"drugs\" : [");

    list.forEach(drug ->
    {
        sb.append("{");
        sb.append("\"id\" : \""+drug.getId());
        sb.append("\", \"name\" : \""+drug.getName());
        sb.append("\", \"category\" : \""+drug.getCategory());
        sb.append("\", \"drasticSubstance\" : \""+drug.getDrasticSubstance());
        sb.append("\", \"volume\" : \""+drug.getVolume());
        sb.append("\", \"expirationDate\" : \""+drug.getExpirationDate().toString());
        sb.append("\", \"price\" : \""+drug.getPrice());
        sb.append("\", \"stock\" : \""+drug.getStock());
        sb.append("\", \"minimum\" : \""+drug.getMinimum());
        sb.append("\", \"priceCoverage\" : \""+drug.getPriceCoverage());

        sb.append("\"},");
    });

    if(sb.lastIndexOf(",") > 0)
        sb.deleteCharAt(sb.lastIndexOf(","));

    sb.append("] }");


    return Response.ok(sb.toString()).type(MediaType.APPLICATION_JSON).build();

}
```

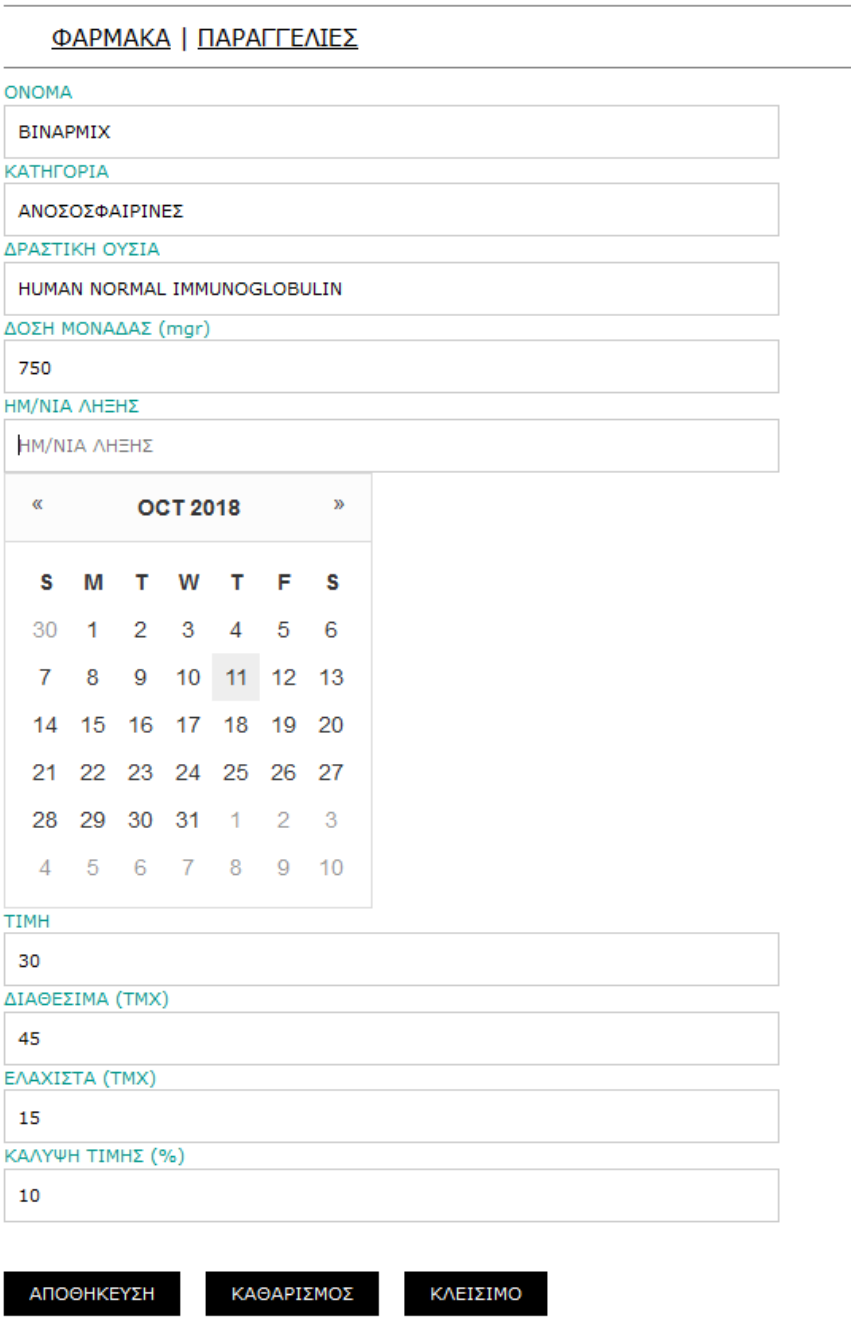
Για την ανάκτηση της λίστας, γίνεται χρήση της μεθόδου getList() του entity Drugs

```
@SuppressWarnings("unchecked")
public static List<Drugs> getList(){

    SessionFactory factory = PersistentUtils.createFactory();
    Session qSession = factory.getCurrentSession();

    qSession.getTransaction().begin();

    Query query =
            qSession.createQuery("from Drugs");
    List<Drugs> drugsList = query.list();

    // Commit transaction
    qSession.getTransaction().commit();

    return drugsList;
}
```

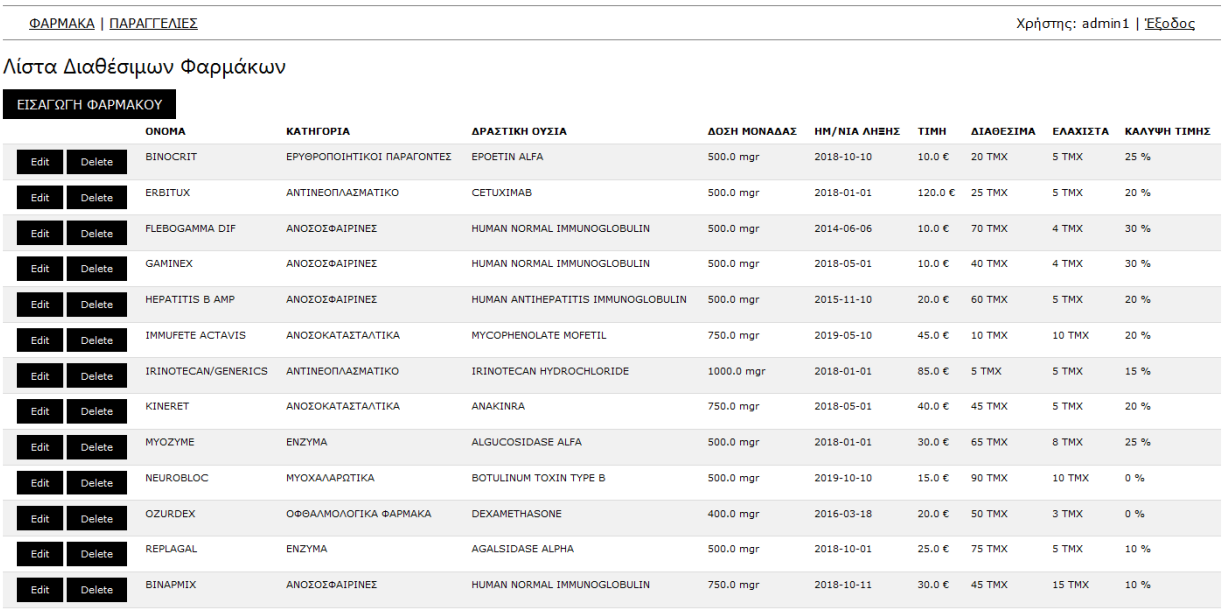
Αφού επιστρέφει το JSON Object, η σχετική σύνταξη του angular μας επιτρέπει να το εμφανίσουμε στον html κώδικα της σελίδας μας όπως φαίνεται παρακάτω:

```
<div class="drugs-list" ng-hide="hidelist">
    <h3>Λίστα Διαθέσιμων Φαρμάκων</h3>
    <button class="w3-btn w3-ripple" ng-click="insertDrug()">ΕΙΣΑΓΩΓΗ ΦΑΡΜΑΚΟΥ</button>
    <table class="w3-table w3-bordered w3-striped">
        <tr>
            <th></th>
            <th>ONOMA</th>
            <th>ΚΑΤΗΓΟΡΙΑ</th>
            <th>ΔΡΑΣΤΙΚΗ ΟΥΣΙΑ</th>
            <th>ΔΟΣΗ ΜΟΝΑΔΑΣ</th>
            <th>ΗΜ/ΝΙΑ ΛΗΞΗΣ</th>
            <th>ΤΙΜΗ</th>
            <th>ΔΙΑΘΕΣΙΜΑ</th>
            <th>ΕΛΑΧΙΣΤΑ</th>
            <th>ΚΑΛΥΨΗ ΤΙΜΗΣ</th>
        </tr>
        <tr ng-repeat="drug in drugs">
           <td>
              <button class="w3-btn w3-ripple" ng-click="editDrug(drug.id)">Edit</button>
              <button class="w3-btn w3-ripple" ng-click="deleteDrug(drug.id)">Delete</button>
           </td>
           <td>{{ drug.name }}</td>
           <td>{{ drug.category }}</td>
           <td>{{ drug.drasticSubstance }}</td>
           <td>{{ drug.volume }} mgr</td>
           <td>{{ drug.expirationDate }}</td>
           <td>{{ drug.price }} &amp;euro;</td>
           <td>{{ drug.stock }} TMX</td>
           <td>{{ drug.minimum }} TMX</td>
           <td>{{ drug.priceCoverage * 100 }} %</td>
         </tr>
    </table>
</div>
```

Επιλέγοντας εισαγωγή, μπορεί να εισάγει ένα νέο φάρμακο, ξεκινώντας από μια κενή φόρμα και εισάγοντας τιμές όπως φαίνεται παρακάτω

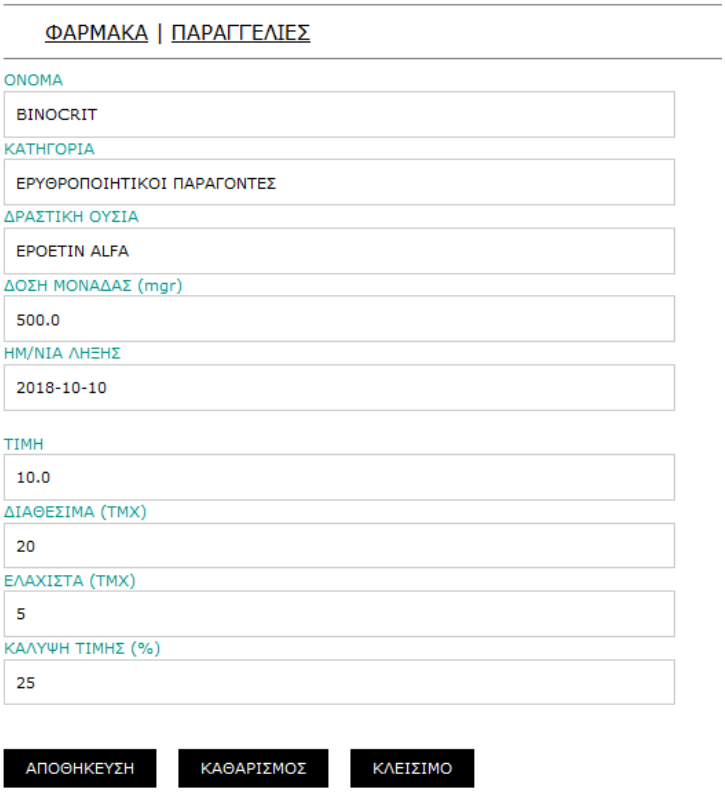


Μετά την εισαγωγή, θα επιστραφεί αυτόματα στη λίστα βλέποντας τα χαρακτηριστικά που εισάγαγε στο τέλος αυτής:



	ΟΝΟΜΑ	ΚΑΤΗΓΟΡΙΑ	ΔΡΑΣΤΙΚΗ ΟΥΣΙΑ	ΔΟΣΗ ΜΟΝΑΔΑΣ	ΗΜ/ΝΙΑ ΛΗΞΗΣ	ΤΙΜΗ	ΔΙΑΘΕΣΙΜΑ	ΕΛΑΧΙΣΤΑ	ΚΑΛΥΨΗ ΤΙΜΗΣ
Edit Delete	BINOCRIT	ΕΡΥΘΡΟΠΟΙΗΤΙΚΟΙ ΠΑΡΑΓΟΝΤΕΣ	EPOETIN ALFA	500.0 mgr	2018-10-10	10.0 €	20 TMX	5 TMX	25 %
Edit Delete	ERBITUX	ΑΝΤΙΝΕΟΠΛΑΣΜΑΤΙΚΟ	CETUXIMAB	500.0 mgr	2018-01-01	120.0 €	25 TMX	5 TMX	20 %
Edit Delete	FLEBOGAMMA DIF	ΑΝΟΣΟΣΦΑΙΡΙΝΕΣ	HUMAN NORMAL IMMUNOGLOBULIN	500.0 mgr	2014-06-06	10.0 €	70 TMX	4 TMX	30 %
Edit Delete	GAMINEX	ΑΝΟΣΟΣΦΑΙΡΙΝΕΣ	HUMAN NORMAL IMMUNOGLOBULIN	500.0 mgr	2018-05-01	10.0 €	40 TMX	4 TMX	30 %
Edit Delete	HEPATITIS B AMP	ΑΝΟΣΟΣΦΑΙΡΙΝΕΣ	HUMAN ANTIHEPATITIS IMMUNOGLOBULIN	500.0 mgr	2015-11-10	20.0 €	60 TMX	5 TMX	20 %
Edit Delete	IMMUFETE ACTAVIS	ΑΝΟΣΟΚΑΤΑΣΤΑΛΤΙΚΑ	MYCOPHENOLATE MOFETIL	750.0 mgr	2019-05-10	45.0 €	10 TMX	10 TMX	20 %
Edit Delete	IRINOTECAN/GENERICS	ΑΝΤΙΝΕΟΠΛΑΣΜΑΤΙΚΟ	IRINOTECAN HYDROCHLORIDE	1000.0 mgr	2018-01-01	85.0 €	5 TMX	5 TMX	15 %
Edit Delete	KINERET	ΑΝΟΣΟΚΑΤΑΣΤΑΛΤΙΚΑ	ANAKINRA	750.0 mgr	2018-05-01	40.0 €	45 TMX	5 TMX	20 %
Edit Delete	MYOZYME	ΕΝΖΥΜΑ	ALGUCOSIDASE ALFA	500.0 mgr	2018-01-01	30.0 €	65 TMX	8 TMX	25 %
Edit Delete	NEUROBLOC	ΜΥΟΧΑΛΑΡΩΤΙΚΑ	BOTULINUM TOXIN TYPE B	500.0 mgr	2019-10-10	15.0 €	90 TMX	10 TMX	0 %
Edit Delete	OZURDEX	ΟΦΘΑΛΜΟΛΟΓΙΚΑ ΦΑΡΜΑΚΑ	DEXAMETHASONE	400.0 mgr	2016-03-18	20.0 €	50 TMX	3 TMX	0 %
Edit Delete	REPLAGAL	ΕΝΖΥΜΑ	AGALSIDASE ALPHA	500.0 mgr	2018-10-01	25.0 €	75 TMX	5 TMX	10 %
Edit Delete	BINAPMIX	ΑΝΟΣΟΣΦΑΙΡΙΝΕΣ	HUMAN NORMAL IMMUNOGLOBULIN	750.0 mgr	2018-10-11	30.0 €	45 TMX	15 TMX	10 %

Αντίστοιχα, όταν επιλέξει να κάνει edit ένα φάρμακο, θα του εμφανιστεί προ συμπληρωμένη η φόρμα ως εξής:

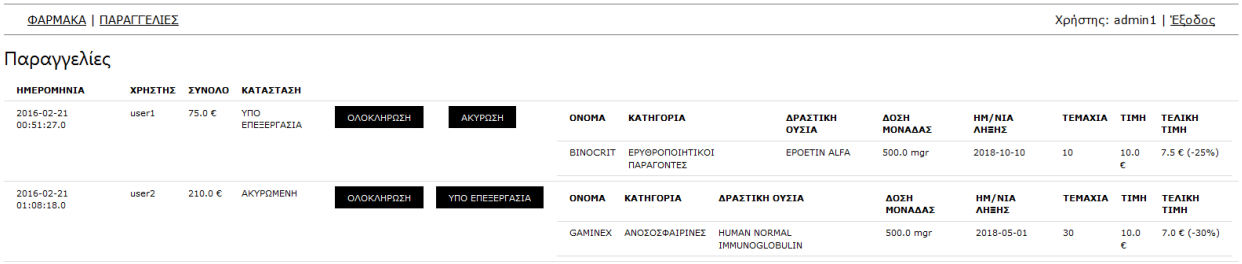


Το format των ημερομηνιών είναι προκαθορισμένο για να μπορεί να μετατραπεί σε TIMESTAMP στη βάση, και ορίζεται από τη συνάρτηση formatDate() του angular controller του application module ‘adminApp’:

```
$scope.formatDate = function (date) {
    function pad(n) {
        return n < 10 ? '0' + n : n;
    }

    return date && date.getFullYear()
        + '-' + pad(date.getMonth() + 1)
        + '-' + pad(date.getDate());
}
```

Επιλέγοντας Διαγραφή, αφού γίνει η αίτηση προς το Rest Endpoint /drugs/delete/{id} θα ανανεωθεί η λίστα χωρίς να περιέχει το φάρμακο που διαγράφηκε.

Σχετικά με τη διαχείριση των παραγγελιών, ο διαχειριστής βλέπει την παρακάτω λίστα

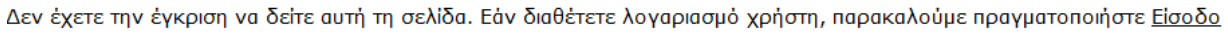


Εκεί, μπορεί να δει όλες τις παραγγελίες των χρηστών μαζί με τις λεπτομέρειες και να προχωρήσει στο χαρακτηρισμό τους ως «ΑΚΥΡΩΜΕΝΗ», «ΥΠΟ ΕΠΕΞΕΡΓΑΣΙΑ» και «ΟΛΟΚΛΗΡΩΜΕΝΗ».

Τέλος, επιλέγοντας έξοδο, εξέρχεται από την περιορισμένης πρόσβασης περιοχή και ανακατευθύνεται στην αρχική σελίδα όπου καλείται να εισάγει τα στοιχεία του. Αυτό επιτυγχάνεται με την μέθοδο logoutUser() του Login Servlet:

```
private void logoutUser(HttpServletRequest request, HttpServletResponse response){
    HttpSession session = request.getSession();
    session.removeAttribute("logged");
    try {
        request.setCharacterEncoding("UTF-8");
        response.sendRedirect(request.getServletContext().getContextPath() + "/index.jsp");

//          request.getRequestDispatcher("/index.jsp").include(request, response);
    } catch (/*ServletException |*/ IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

}
```

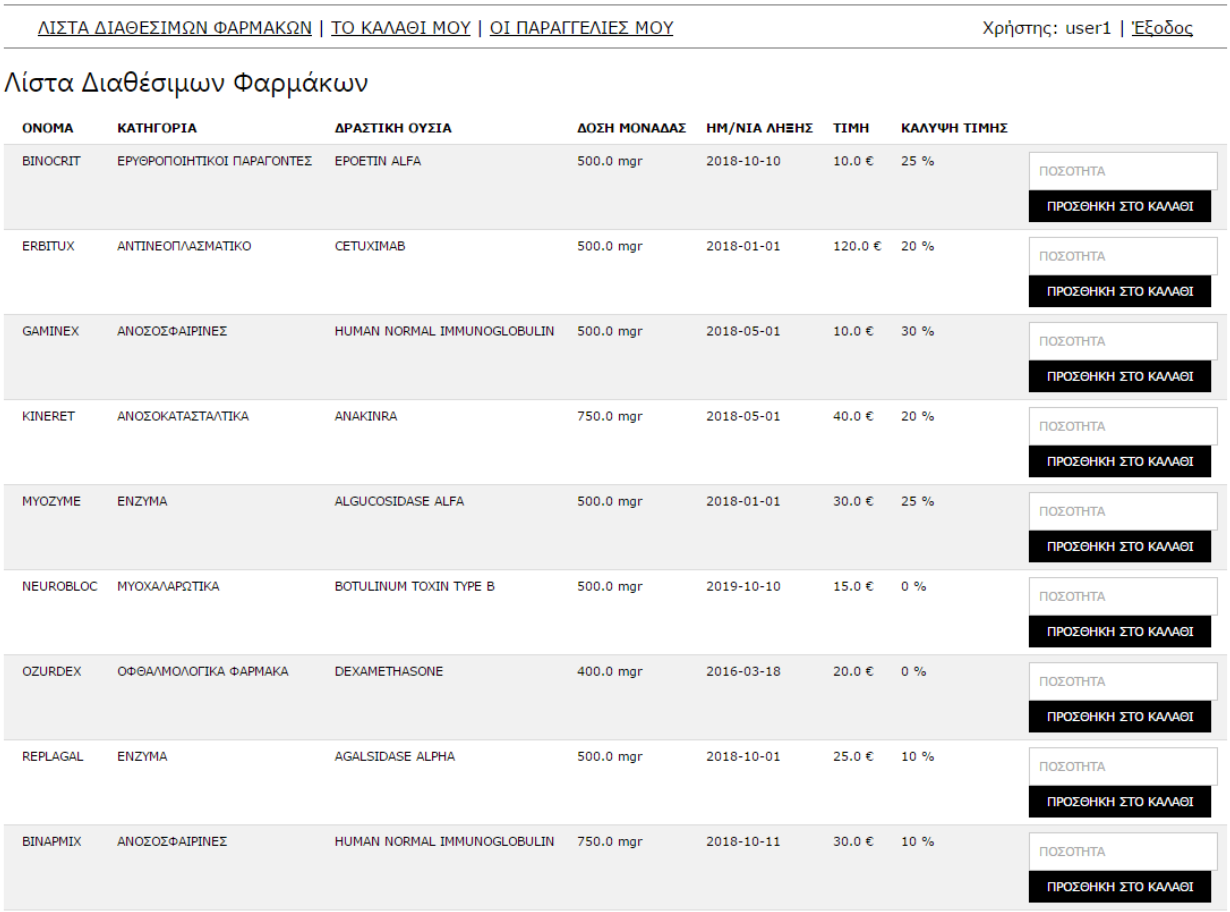
Σε περίπτωση που είτε λήξει η διάρκεια της συνεδρίας του, είτε έχει οδηγηθεί εσφαλμένα στο url της περιοχής περιορισμένης πρόσβασης και δεν έχει ταυτοποίησει τα στοιχεία του, θα του εμφανιστεί η παρακάτω σελίδα:



Τόσο η λειτουργεία της εξόδου, όσο και η εσφαλμένη πλοήγηση, είναι επίσης λειτουργίες που συναντώνται και στον λογαριασμό του χρήστη.

5.3 Λειτουργείες Χρήστη

Κατά αντιστοιχία με τον διαχειριστή, ο χρήστης αφού κάνει login βλέπει την παρακάτω σελίδα που περιέχει τη λίστα των φαρμάκων, με ορισμένους περιορισμούς:



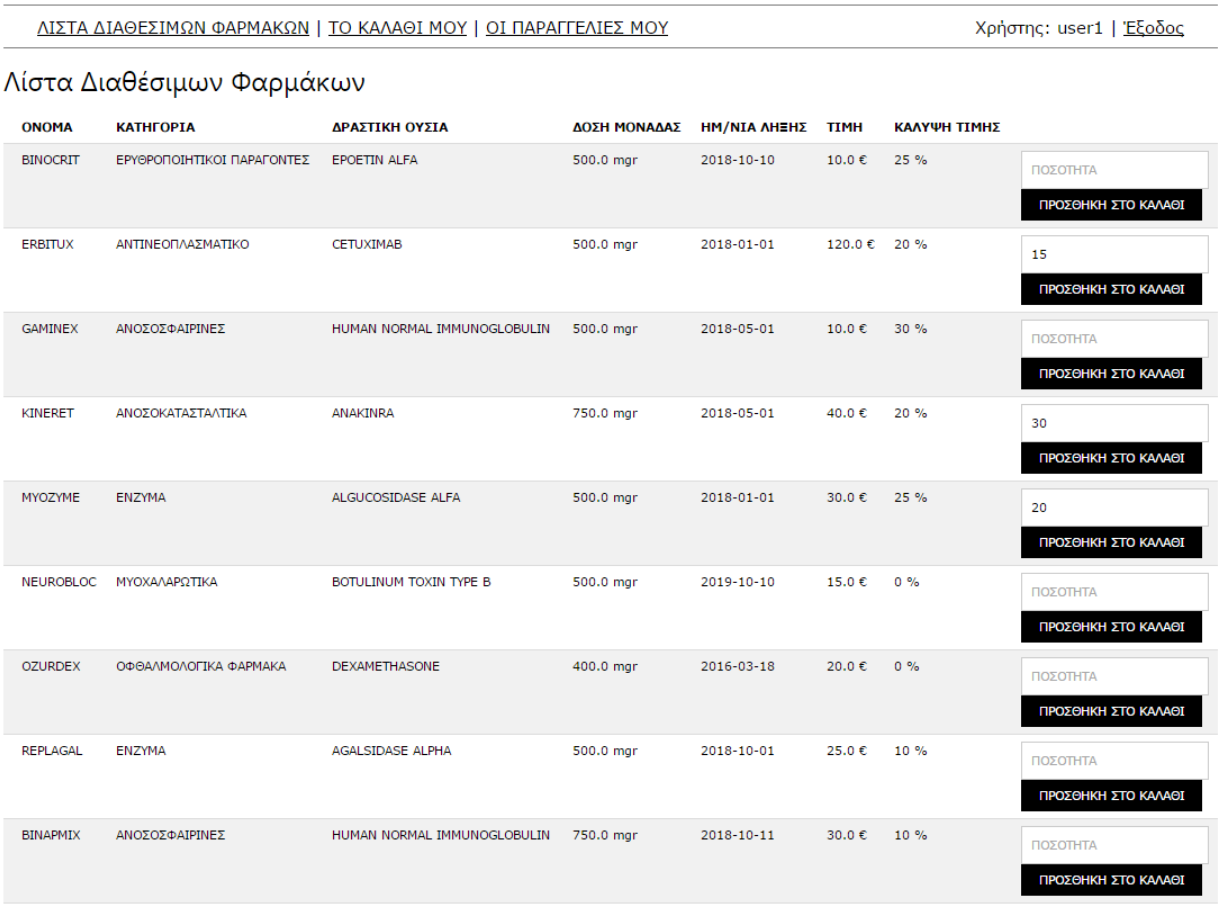
Για την ανάκτηση αυτής της λίστας γίνεται η κλήση στο rest path /drugs/list μέσω της συνάρτησης του angular controller του application module ‘userApp’ που περιγράφεται στο αρχείο user-angular-app.js, η οποία όμως επιστέφει μια λίστα η οποία δεν περιέχει τα φάρμακα για τα οποία έχει παρέλθει η ημερομηνία λήξης, καθώς και αυτά που το ελάχιστο επιτρεπτό όριο αποθέματος ισούται με το απόθεμα, καθώς ο χρήστης δεν θα είχε τη δυνατότητα να τα παραγγείλει:

```
var app = angular.module('userApp', [ 'mp.datePicker' ]);

app.controller('drugsCtrl', function($scope, $http) {

    $scope.showDrugsList = function(){
        $scope.hidelist = false;
        $scope.hideBasket = true;
        $scope.hideOrders = true;
    },

    $scope.getDrugs = function() {
        $scope.showDrugsList();
        $http.get(baseUrl+"/farmakeio/rest/medics/drugs/list", {"responseType" : "json"}).then(function(response) {

            $scope.drugs = new Array();

            response.data.drugs.forEach(function(item){

                var includeItem = true;
                // check expiration Date and stock against minimum
                if(Date.now() < Date.parse(item.expirationDate) && parseInt(item.stock) > parseInt(item.minimum))
                    $scope.drugs.push(item);
            });
        });
    },
```

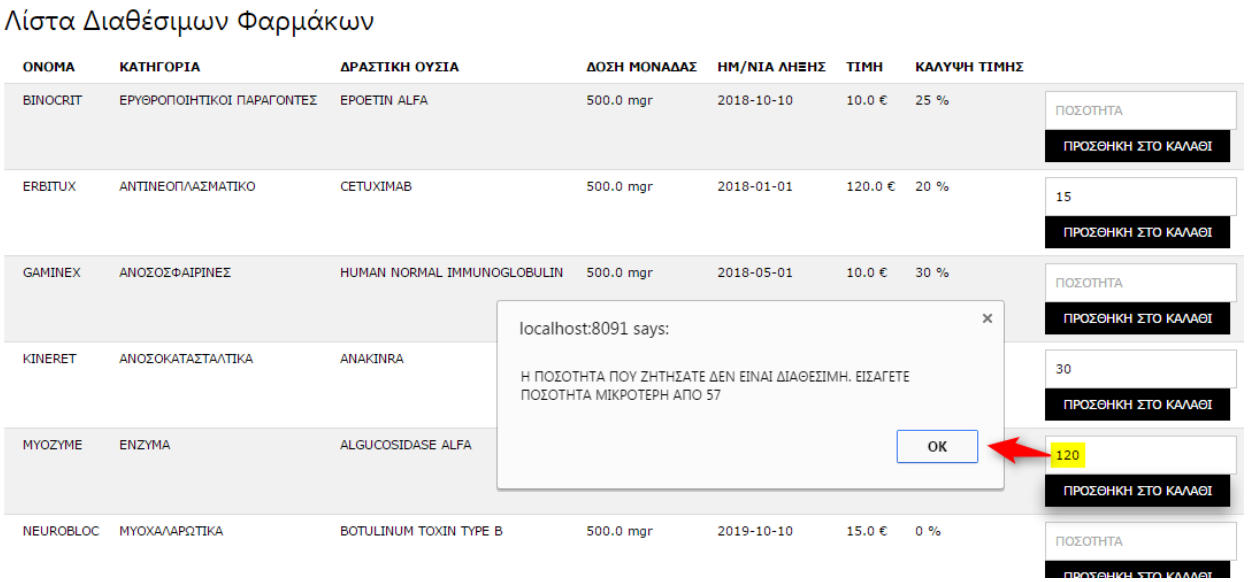
Αφού επιστραφεί το JSON Object, η σχετική σύνταξη του angular μας επιτρέπει να το εμφανίσουμε στον html κώδικα της σελίδας μας όπως φαίνεται παρακάτω:

```
<div class="drugs-list" ng-hide="hidelist">
    <h3>Λίστα Διαθέσιμων Φαρμάκων</h3>
    <table class="w3-table w3-bordered w3-striped">
        <tr>
            <th>ONOMA</th>
            <th>ΚΑΤΗΓΟΡΙΑ</th>
            <th>ΔΡΑΣΤΙΚΗ ΟΥΣΙΑ</th>
            <th>ΔΟΣΗ ΜΟΝΑΔΑΣ</th>
            <th>ΗΜ/ΝΙΑ ΛΗΞΗΣ</th>
            <th>ΤΙΜΗ</th>
            <th>ΚΑΛΥΨΗ ΤΙΜΗΣ</th>
            <th></th>
        </tr>
        <tr ng-repeat="drug in drugs">
            <td>{{ drug.name }}</td>
            <td>{{ drug.category }}</td>
            <td>{{ drug.drasticSubstance }}</td>
            <td>{{ drug.volume }} mgr</td>
            <td>{{ drug.expirationDate }}</td>
            <td>{{ drug.price }} &amp;euro;</td>
            <td>{{ drug.priceCoverage * 100 }} %</td>
            <td>
                <form>
                        <input type="hidden" ng-model="drugId" ng-value="{{drug.id}}">
                    <input type="hidden" ng-model="drugStock" ng-value="{{drug.stock}}">
                    <input type="hidden" ng-model="drugMinimum" ng-value="{{drug.minimum}}"> -->
                    <input class="w3-input w3-border quantity-{{drug.id}}" type="text" placeholder="ΠΟΣΟΤΗΤΑ" value="">
                    <button class="w3-btn w3-ripple" ng-click="validateAndAdd(drug,drug.stock - drug.minimum,'')">ΠΡΟΣΘΗΚΗ ΣΤΟ ΚΑΛΑΘΙ</button>
                </form>
            </td>
         </tr>
    </table>
</div>
```

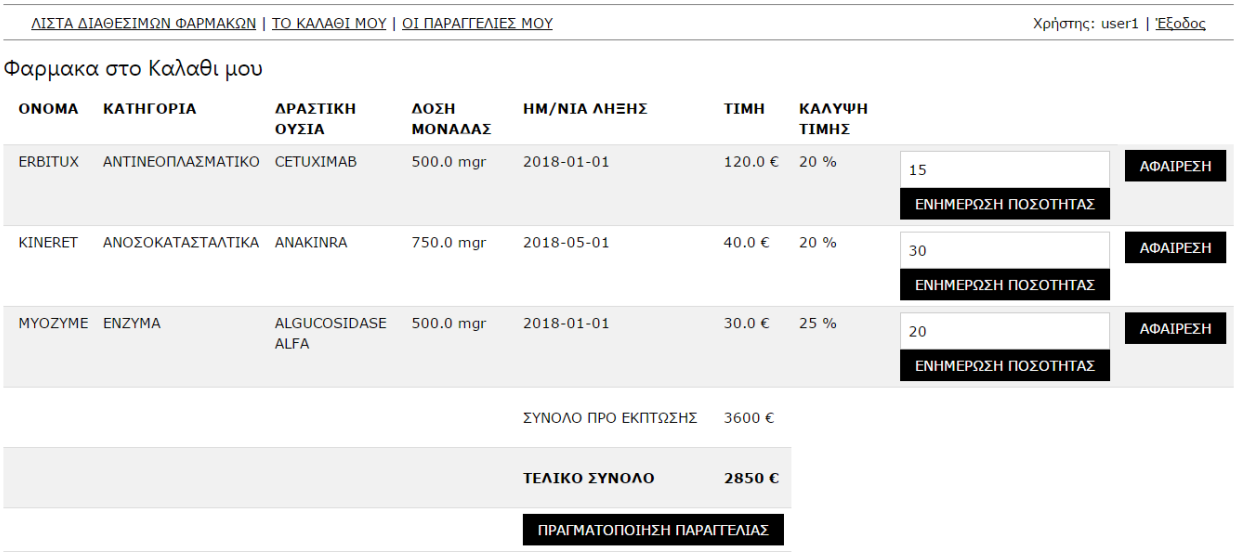
Ο χρήστης, για να πραγματοποιήσει παραγγελία, πρέπει να συμπληρώσει τον αριθμό των τεμαχίων στο κάθε φάρμακο που τον ενδιαφέρει και να πατήσει «ΠΡΟΣΘΗΚΗ ΣΤΟ ΚΑΛΑΘΙ» όπως φαίνεται στην παρακάτω εικόνα:



Πριν γίνει η προθήκη στο καλάθι, τρέχει η συνάρτηση validateAndAdd() του angular controller του user που ελέγχει ότι η επιλογή του χρήστη δεν υπερβαίνει σε ποσότητα τη διαφορά stock - ελάχιστο αριθμό stock. Σε αντίθετη περίπτωση του εμφανίζεται ένα μήνυμα όπως το παρακάτω:



Εφόσον ο χρήστης τελειώσει, μπορεί να ολοκληρώσει την παραγγελία του επισκέπτοντας «ΤΟ ΚΑΛΑΘΙ ΜΟΥ» όπου θα δει την παρακάτω εικόνα:



Εκεί βλέπει λεπτομερώς τα στοιχεία του κάθε φαρμάκου που επέλεξε, καθώς και τα σύνολα τιμών. Επίσης, έχει τη δυνατότητα αφαίρεσης τεμαχίων και τροποποίησης ποσοτήτων. Η τελική τιμή της παραγγελίας υπολογίζεται συνολικά λαμβάνοντας υπόψιν τις ποσοστιαίες καλύψεις τιμών. Ο υπολογισμός αυτός καθώς και η γενικότερη διαχείριση του καλαθιού γίνεται αποκλειστικά στο frontend με τη βοήθεια του angular και ο αντίστοιχος κώδικας εμφανίζεται παρακάτω.

```
$scope.validateAndAdd = function(drug,available,classNamePrefix){

    var quantinty = document.getElementsByClassName(classNamePrefix+"quantity-"+drug.id)[0].value;
    if(quantinty === undefined || quantinty === ""){
        alert("ΠΑΡΑΚΑΛΩ ΕΙΣΑΓΕΤΕ ΠΟΣΟΤΗΤΑ");
    }else if(quantinty > available){
        alert("Η ΠΟΣΟΤΗΤΑ ΠΟΥ ΖΗΤΗΣΑΤΕ ΔΕΝ ΕΙΝΑΙ ΔΙΑΘΕΣΙΜΗ. ΕΙΣΑΓΕΤΕ ΠΟΣΟΤΗΤΑ ΜΙΚΡΟΤΕΡΗ ΑΠΟ "+available);
    }else{

        var found = $scope.searchBasket(drug);

        if(found === undefined)
            $scope.myBasket.push({"drug" : drug, "quantity" : quantinty});
        else{
            var idx = $scope.myBasket.indexOf(found);
            $scope.myBasket[idx].quantity = quantinty;
        }

        $scope.calculateBasketCosts();
    }
},

$scope.removeFromBasket = function(drug){

    var found = $scope.searchBasket(drug);

    if(found !== undefined){
        var idx = $scope.myBasket.indexOf(found);
        $scope.myBasket.splice(idx,1);

        $scope.calculateBasketCosts();
    }
},

$scope.searchBasket = function(drug){

    var found = $scope.myBasket.find(function(item){
        if(item.drug.id == drug.id)
            return true;
        return false;
    });

    return found;
},

$scope.showBasket = function(){

    $scope.hidelist = true;
    $scope.hideBasket = false;
    $scope.hideOrders = true;

    $scope.calculateBasketCosts();
},


  $scope.calculateBasketCosts = function(){

      $scope.basketGross = 0;
      $scope.basketTotal = 0;

      // calculate gross and total cost
      $scope.myBasket.forEach(function(item){

          var currentGross = item.drug.price * item.quantity;
          $scope.basketGross += currentGross;

          var currentTotal = (item.drug.price - (item.drug.price * item.drug.priceCoverage)) * item.quantity;
          $scope.basketTotal += currentTotal;
      });
  },
```

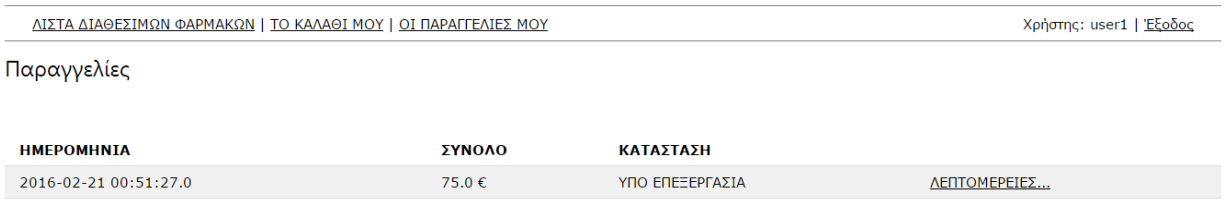
Αντιστοίχως, ο κώδικας html που αναλαμβάνει την εμφάνιση είναι ο ακόλουθος:

```
<div class="user-basket" ng-hide="hideBasket">
    <h3>Φαρμακα στο Καλαθι μου</h3>
    <p ng-hide="myBasket.length">ΔΕΝ ΥΠΑΡΧΟΥΝΕ ΦΑΡΜΑΚΑ ΣΤΟ ΚΑΛΑΘΙ ΣΑΣ</p>

    <table class="w3-table w3-bordered w3-striped" ng-show="myBasket.length">
        <tr>
            <th>ΟΝΟΜΑ</th>
            <th>ΚΑΤΗΓΟΡΙΑ</th>
            <th>ΔΡΑΣΤΙΚΗ ΟΥΣΙΑ</th>
            <th>ΔΟΣΗ ΜΟΝΑΔΑΣ</th>
            <th>ΗΜ/ΝΙΑ ΛΗΞΗΣ</th>
            <th>ΤΙΜΗ</th>
            <th>ΚΑΛΥΨΗ ΤΙΜΗΣ</th>
            <th></th>
        </tr>
        <tr ng-repeat="item in myBasket">
            <td>{{ item.drug.name }}</td>
            <td>{{ item.drug.category }}</td>
            <td>{{ item.drug.drasticSubstance }}</td>
            <td>{{ item.drug.volume }} mgr</td>
            <td>{{ item.drug.expirationDate }}</td>
            <td>{{ item.drug.price }} &euro;</td>
            <td>{{ item.drug.priceCoverage * 100 }} %</td>
            <td>
                <form>
                    <input class="w3-input w3-border basket-quantity-{{item.drug.id}}" type="text" placeholder="ΠΟΣΟΤΗΤΑ" value={{item.quantity}}>
                    <button class="w3-btn w3-ripple" ng-click="validateAndAdd(item.drug,item.drug.stock - item.drug.minimum,'basket-')">ΕΝΗΜΕΡΩΣΗ ΠΟΣΟΤΗΤΑΣ</button>
                </form>
            </td>
            <td>
                <form>
                    <button class="w3-btn w3-ripple" ng-click="removeFromBasket(item.drug)">ΑΦΑΙΡΕΣΗ</button>
                </form>
            </td>
        </tr>
        <tr>
            <td colspan="4"></td>
            <td><p>ΣΥΝΟΛΟ ΠΡΟ ΕΚΠΤΩΣΗΣ</p></td>
            <td><p>{{ basketGross }} &euro;</p></td>
        </tr>
        <tr>
            <td colspan="4"></td>
            <td><p><b>ΤΕΛΙΚΟ ΣΥΝΟΛΟ</b></p></td>
            <td><p><b>{{ basketTotal }} &euro;</b></p></td>
        </tr>
        <tr>
            <td colspan="4"></td>
            <td colspan="2">
                <button class="w3-btn w3-ripple" ng-click="placeOrder()">ΠΡΑΓΜΑΤΟΠΟΙΗΣΗ ΠΑΡΑΓΓΕΛΙΑΣ</button>
            </td>
        </tr>
    </table>
```

Ο χρήστης επιλέγοντας πραγματοποίηση παραγγελίας θα ολοκληρώσει την παραγγελία και θα ανακατευθυνθεί στη σελίδα παραγγελιών. Η πραγματοποίηση της παραγγελίας γίνεται με τη χρήση της συνάρτησης placeOrder() του angular controller του user:

```
$scope.placeOrder = function(){

    var data = {
            datePlaced: Date.now(),
            total: $scope.basketTotal,
            items: $scope.myBasket
    };

    var reqData = angular.toJson(data);
    console.log(reqData);

    $http.post(baseUrl+'/farmakeio/rest/medics/orders/place', reqData).then(function(data) {

        $scope.getDrugs();
        $scope.myBasket = new Array();
        $scope.basketGross = 0;
        $scope.basketTotal = 0;
    });

    $scope.showOrders();
```

Στη λίστα παραγγελιών θα δει κάτι αντίστοιχο με αυτή τη σελίδα:



Η λίστα των παραγγελιών ανακτάται με τη χρήση της συνάρτησης showOrder() του angular controller του user:

```
$scope.showOrders = function(){

    $scope.hidelist = true;
    $scope.hideBasket = true;
    $scope.hideOrders = false;

    $scope.detailActive = 0;

    $http.get(baseUrl+"/farmakeio/rest/medics/orders/userlist", {"responseType" : "json"}).then(function(response) {

        $scope.orders = new Array();

        response.data.orders.forEach(function(item){
            $scope.orders.push(item);
        });
    });
},
```

Επιλέγοντας λεπτομέρειες, ο χρήστης μπορεί να δει λεπτομέρειες για τα τεμάχια της παραγγελίας:

Αυτό επιτυγχανεται με τη χρήση της συνάρτησης showDetails() του angular controller του user:

```
$scope.showOrderDetails = function(id){

    $scope.hidelist = true;
    $scope.hideBasket = true;
    $scope.hideOrders = false;

    $http.get(baseUrl+"/farmakeio/rest/medics/orders/"+id+"/items", {"responseType" : "json"}).then(function(response) {

        $scope.orderItems = new Array();

        response.data.items.forEach(function(item){
            $scope.orderItems.push(item);
        });

        $scope.detailActive = id;
    });
},
```

Τέλος, Υπεύθυνος κώδικας HTML για την εμφάνιση των παραγγελιών και των λεπτομερειών του χρήστη είναι ο παρακάτω:

```
<div class="orders-list" ng-hide="hideOrders">
    <h3>Παραγγελίες</h3>
    <p ng-hide="orders.length">ΔΕΝ ΕΧΕΤΕ ΠΡΑΓΜΑΤΟΠΟΙΗΣΕΙ ΚΑΠΟΙΑ ΠΑΡΑΓΓΕΛΙΑ</p>
    <table class="w3-table w3-bordered w3-striped" ng-show="orders.length">
        <tr>
            <th>ΗΜΕΡΟΜΗΝΙΑ</th>
            <th>ΣΥΝΟΛΟ</th>
            <th>ΚΑΤΑΣΤΑΣΗ</th>
            <th></th>
        </tr>
        <tr ng-repeat="order in orders">
            <td>{{ order.datePlaced }}</td>
            <td>{{ order.total }} &euro;</td>
            <td>{{ orderStatus(order.completed) }}</td>
            <td><a href="" ng-click="showOrderDetails(order.id)">ΛΕΠΤΟΜΕΡΕΙΕΣ...</a></td>
            <br/><br/>
            <table class="w3-table" ng-show="orderItems.length">
                <tr>
                    <th>ΟΝΟΜΑ</th>
                    <th>ΚΑΤΗΓΟΡΙΑ</th>
                    <th>ΔΡΑΣΤΙΚΗ ΟΥΣΙΑ</th>
                    <th>ΔΟΣΗ ΜΟΝΑΔΑΣ</th>
                    <th>ΗΜ/ΝΙΑ ΛΗΞΗΣ</th>
                    <th>ΤΕΜΑΧΙΑ</th>
                    <th>ΤΙΜΗ</th>
                    <th>ΤΕΛΙΚΗ ΤΙΜΗ</th>
                </tr>
                <tr ng-repeat="item in orderItems">
                    <td>{{ item.name }}</td>
                    <td>{{ item.category }}</td>
                    <td>{{ item.drasticSubstance }}</td>
                    <td>{{ item.volume }} mgr</td>
                    <td>{{ item.expirationDate }}</td>
                    <td>{{ item.quantity }}</td>
                    <td>{{ item.price }} &euro;</td>
                    <td>{{ item.value }} &euro; (-{{ item.priceCoverage * 100 }}%)</td>
                </tr>
                <tr><td><a href="" ng-click="closeOrderDetails()">ΚΛΕΙΣΙΜΟ</a></td><td colspan="7"></td></tr>
            </table>
        </tr>
    </table>
</div>
```

6 Συμπεράσματα

6.1 Γενικά

Η ενασχόλησή μας με τις τεχνολογίες που χρησιμοποιούνται για την δημιουργία μίας σύγχρονης σελίδας μας άφησε εντυπωσιασμένους με τις δυνατότητες που προσφέρονται στους σχεδιαστές και προγραμματιστές για το διαδίκτυο.

Αυτές καλύπτουν ένα μεγάλο εύρος εικαστικών απαιτήσεων, δίνοντας πολύ μεγάλη ελευθερία ώστε το αποτέλεσμα να ανταποκρίνεται όσο το δυνατόν περισσότερο στην αρχική σχεδιαστική ιδέα. Επιπλέον από άποψη λειτουργικότητας υπάρχουν εξίσου σημαντικές ευκαιρίες, χωρίς να φαίνονται σχεδόν πουθενά οι περιορισμοί των τεχνολογιών.

Βέβαια όλα αυτά απαιτούν μία εκ βάθους γνώση των τεχνολογιών που χρησιμοποιούνται. Για τη δημιουργίας της σελίδας χρησιμοποιήσαμε πέντε διαφορετικές γλώσσες προγραμματισμού ή περιγραφής, κάθε μία με εντελώς διαφορετικό συντακτικό. Κυριότερα κάθε μία από αυτές έχει τελείως διαφορετική φιλοσοφία και ιδιαιτερότητες, καθιστώντας τον συνδυασμό τους πολύ δύσκολο.

6.2 Μελλοντικοί στόχοι

Αν και γενικά πετύχαμε τον στόχο μας στη δημιουργία της εφαρμογής υπάρχουν διάφορές επιπλέον δυνατότητες που θα μπορούσαν να προστεθούν, με τις οποίες δεν ασχοληθήκαμε γιατί ξέφευγαν από το πλαίσιο και τους στόχους του θέματος.

Το κυριότερο είναι η επέκταση της εφαρμογής ώστε να γίνει ένα πλήρες ηλεκτρονικό φαρμακείο και σε συνδυασμό με περισσότερες δυνατότητες αυτόματης αλληλεπίδρασης με hardware υλικό και προσαρμογή. Κάτι τέτοιο χρειάζεται έρευνα σχετικά με τις κατάλληλες πρακτικές για την ασφάλεια του συστήματος και τον χρηστών / πελατών την διασύνδεση με κάποιο σύστημα ηλεκτρονικών πληρωμών καθώς και την συσχέτιση software και hardware και προγραμματισμό σε χαμηλό επίπεδο γλωσσών.

7 Βιβλιογραφία

Castro, E., & Hyslop, B. (2011). *HTML 5 και CSS 3 με εικόνες.* Κλειδάριθμος.

Spurlock, J. (2013). *Bootstrap-Responsive Web Development.* O'Reilly Media.

Meloni, J. C. (2013). *Μάθετε HTML 5, CSS και JavaSript.* Γκιούρδας Μ.

Overson, J., & Strimpel, J. (2013). *Developing Web Components.* O'Reilly Media, Inc.

Aravind, S., & Ulrich, S. (2014). *Learning Bootstrap.* PACKT.

Pollock, J. (2013). *Οδηγός της JavaScript.* Ιανός.

Στασινόπουλος, Γ. (2007). *Διαδίκτυο και εφαρμογές.* Εθνικό Μετσόβιο Πολυτεχνείο.

Καζολέα, Ι. (2013). *Προγραμματισμός Διεπαφών Φορητών Συσκευών.* Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης.

Database. *Wikipedia.* [Ηλεκτρονικό] http://en.wikipedia.org/wiki/Database.

Dynamic Website. *Marvel Infomatics.* [Ηλεκτρονικό] http://www.marvelinfomatics.com/web-development/dynamic_website.html.

Dynamic Websites . *smooth-step.com.* [Ηλεκτρονικό] http://www.smooth-step.com/web-design/dynamic-websites.

HTML. *Βικιπαίδεια.* [Ηλεκτρονικό] http://el.wikipedia.org/wiki/HTML.

HTML 4.0 Specification. *World Wide Web Consortium.* [Ηλεκτρονικό] http://www.w3.org/TR/1998/REC-html40-19980424/.

HTML5. *Βικιπαίδεια.* [Ηλεκτρονικό] http://el.wikipedia.org/wiki/HTML5.

HTML5. *Wikipedia.* [Ηλεκτρονικό] http://en.wikipedia.org/wiki/HTML5.

HTML5 Differences from HTML4. *World Wide Web Consortium.* [Ηλεκτρονικό] http://www.w3.org/TR/2011/WD-html5-diff-20110405/.

MySQL. *Βικιπαίδεια.* [Ηλεκτρονικό] http://el.wikipedia.org/wiki/MySQL.

onload Event. *w3schools.* [Ηλεκτρονικό] http://www.w3schools.com/jsref/event_onload.asp.

PHP: Hypertext Preprocessor. *PHP.net.* [Ηλεκτρονικό] http://php.net/.

Pilgrim, Mark. Dive Into HTML5. *Dive Into HTML5.* [Ηλεκτρονικό] http://diveintohtml5.info/.

Processing static web pages. *etutorials.org.* [Ηλεκτρονικό] http://etutorials.org/Macromedia/Dream+Weaver+Online+Help/Getting+Started+with+Dreamweaver/Understanding+Web+Applications/How+a+web+application+works/Processing+static+web+pages/.

Usage share of web browsers. *Wikipedia.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/Usage_share_of_web_browsers.

Γλώσσες Προγραμματισμού. [Ηλεκτρονικό] https://foss.ntua.gr/wiki/index.php/%CE%93%CE%BB%CF%8E%CF%83%CF%83%CE%B5%CF%82_%CE%A0%CF%81%CE%BF%CE%B3%CF%81%CE%B1%CE%BC%CE%BC%CE%B1%CF%84%CE%B9%CF%83%CE%BC%CE%BF%CF%8D#PHP.

Εισαγωγή στις Βάσεις Δεδομένων (DataBases). *ΚΕΝΤΡΟ ΠΛΗ.ΝΕ.Τ. Ν. ΦΛΩΡΙΝΑΣ.* [Ηλεκτρονικό] http://dide.flo.sch.gr/Plinet/Tutorials/Tutorials-DataBases.html.

Η Γλώσσα Προγραμματισμού PHP . *ΚΕΝΤΡΟ ΠΛΗ.ΝΕ.Τ. Ν. ΦΛΩΡΙΝΑΣ.* [Ηλεκτρονικό] http://dide.flo.sch.gr/Plinet/Tutorials/Tutorials-Php-Analytical.html.

Τι είναι html? [Ηλεκτρονικό] http://pacific.jour.auth.gr/html/.

Σουίτα ασφαλείας και anti-virus Sophos Mobile Security 3.0 για Android συσκευές . *nss.gr.* [Ηλεκτρονικό] https://www.nss.gr/news/445-latest-free-version-of-sophos-mobile-security-delivers-application-protection-faster-scanning-and-web-protection.html?format=html&lang=el.