ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σχεδίαση και ανάπτυξη διαδικτυακού συστήματος διαχείρησης εξοπλισμού / αποθήκης

ΑΥΤΑΝΤΙΛΙΔΗΣ ΡΟΜΑΝ

Επιβλέπων: Αγγελίδης Παντελής
Αναπληρωτής Καθηγητής

Κοζάνη, Σεπτέμβριος 2014

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σχεδίαση και ανάπτυξη διαδικτυακού συστήματος διαχείρησης εξοπλισμού / αποθήκης

ΑΥΤΑΝΤΙΛΙΔΗΣ ΡΟΜΑΝ

Επιβλέπων: Αγγελίδης Παντελής
Αναπληρωτής Καθηγητής

Εξεταστική επιτροπή 25η Σεπτεμβρίου 2014.

Π. Αγγελίδης	Θ. Ζυγκιρίδης
Αναπληρωτής Καθηγητής	Επίκουρος Καθηγητής

Κοζάνη, Σεπτέμβριος 2014

ΠΕΡΙΛΗΨΗ

Στην παρούσα εργασία παρουσιάζουμε ένα λογισμικό διαχείρισης ιατρο-τεχνολογικού εξοπλισμού το οποίο σχεδιάστηκε στα πλαίσια της παρούσας διπλωματικής εργασίας του Πανεπιστημίου Δυτικής Μακεδονίας του τμήματος Μηχανικών Πληροφορικής και Τηλεπικοινωνιών. Η σχεδίαση αυτού του λογισμικού αποσκοπεί στην μηχανοργάνωση των συγκεκριμένων λειτουργικών διαχείρισης της αποθήκης, την προσθήκη επιπρόσθετων λειτουργιών και την διασύνδεση των τμημάτων που διαχειρίζονται τον εξοπλισμό μέσα από μια ενιαία εφαρμογή. Κύριες λειτουργίες του λογισμικού είναι η δυνατότητα διαχείρισης των οντοτήτων που αλληλεπιδρούν με την αποθήκη, όπως οι πελάτες και οι προμηθευτές. Παράλληλα η μηχανοργάνωση του εξοπλισμού η καταγραφή αλλαγών καθώς και ενημέρωση κατάστασης και θέσης των προϊόντων.

ABSTRACT

In this paper we present a software management medico-technical equipment which is designed in the context of this thesis, University of Western Macedonia in the Department of Informatics and Telecommunications.
H design of this software is aimed at computerizing the specific functional warehouse management, adding additional functions and interconnect segment manage equipment from a single application. Main functions of the software is the ability to manage the entities that interact with the warehouse, such as customers and suppliers. Alongside the computerized equipment, inventory changes and status update and release of the products. The application must be able to offer immediate image real-time status of the warehouse.
Is web-based and designed architecture Model-View-Controller.

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ τον επιβλέποντα καθηγητή μου κ. Παντελή Αγγελίδη, καθηγητή του Πανεπιστημίου Δυτικής Μακεδονίας για την ανάθεση και καθοδήγηση της διπλωματικής μου εργασίας, καθώς επίσης και για την προσωπική του ενασχόληση κατά την πορεία της εξέλιξης της εργασίας μου και την άριστη συνεργασία που είχαμε όλα αυτά τα χρόνια. Τέλος ευχαριστώ τους γονείς μου για την συνεχή τους στήριξη.

1. **Εισαγωγή**

1.1 **Αντικείμενο διπλωματικής εργασίας**

Σκοπός είναι η ανάπτυξη διαδικτυακής εφαρμογής μηχανογράφησης που θα αποτελέσει ένα σύστημα διαχείρισης ιατρο – τεχνολογικού εξοπλισμού, όπου οι εξουσιοδοτημένοι χρήστες θα έχουν άμεση εικόνα και πληροφόριση για το σύνολο του εξοπλισμού, τη χωροταξική του κατανομή, τα χαρακτηριστικά του και την κατάσταση στην οποία βρίσκεται. Το σύστημα αφορά στην καταγραφή, κωδικοιποίηση και αρχειοθέτηση εξοπλισμού, καθώς και στην αλυσίδα παρακολούθησης διεργασιών συντήρησης, επισκευής, πώλησης και ελέγχου ποιότητας. Το σύνολο των λειτουργιών επιτυγχάνουν την διασύνδεση των τμημάτων που διαχειρίζονται τον εξοπλισμό μέσα από μια ενιαία εφαρμογή.

1.2 **Διαδικτυακή εφαρμογή**

Μια διαδικτυακή εφαρμογή είναι ένα οποιοδήποτε πρόγραμμα που εκτελείται στον περιηγητή του υπολογιστή, έχει δημιουργηθεί με γλώσσες προγραμματισμού όπως η javascript, HTML, και CSS.

Βασικός στόχος μιας διαδικτυακής εφαρμογής είναι να προσφέρει στον χρήστη την δυνατότητα να αλληλεπιδράσει με πληροφορίες, να εκτελέσει διεργασίες, να επεξεργαστεί δεδομένα ανεξάρτητα απο την συσκευή και το λειτουργικό σύστημα που διαθέτει.

Οι διαδικτυακές εφαρμογές είναι δημοφιλείς λόγο της διάδοσης των εφαρμογών περιήγησης στον ιστό και της οικιότητας που έχουν πλέον οι χρήστες με αυτές. Βασικός παράγοντας στην εξάπλοση τους είναι η δυνατότητα που προσφέρουν για ανανεώσεις και συντήρηση χωρίς την απαίτηση για διαμοιρασμό και εγκατάσταση λογισμικού προς χιλιάδες χρήστες που μπορεί να βρίσκονται οπουδήποτε στον κόσμο. Διαδικτυακές εφαρμογές γνωστές σε όλους μας είναι οι εφαρμογές ηλεκτρονικού ταχυδρομοίου, αποθήκευσης αρχείων στον ιστό, επεξεργασία αρχείων στον ιστό, κλπ.

Μια διαδικτυακή εφαρμογή μπορεί να είναι προσβάσιμη στους χρήστες μέσω ίντερνετ ή τοπικού δικτύου.

Βέβαια στο αρνητικό σκέλος συγκαταλέγεται το γεγονός ότι χωρίς πρόσβαση στο διαδίκτυο ή στο δίκτυο δεν υπάρχει δυνατότητα χρήσης της εφαρμογής. Υπάρχουν λύσεις που

προσφέρουν κάποιες βασικές εργασίες εκτός δικτύου αλλά αυτό δεν είναι αρκετό.

1.2.1 **Ιστορία του διαδικτύου**

Το Διαδίκτυο ή Ίντερνετ (Internet) είναι ένα επικοινωνιακό δίκτυο ηλεκτρονικών υπολογιστών, που επιτρέπει την ανταλλαγή δεδομένων μεταξύ οποιουδήποτε διασυνδεδεμένου υπολογιστή. Η τεχνολογία του είναι κυρίως βασισμένη στην διασύνδεση επιμέρους δικτύων ανά τον κόσμο και πολυάριθμα τεχνολογικά πρωτόκολλα, με κύριο το TCP/IP. Ο αντίστοιχος αγγλικός όρος internet προκύπτει από τη σύνθεση λέξεων : inter-network.

Στην πιο εξειδικευμένη και περισσότερο χρησιμοποιούμενη μορφή του, με τους όρους Διαδίκτυο, Ιντερνέτ ή Ίντερνετ (με κεφαλαίο το αρχικό γράμμα) περιγράφεται το παγκόσμιο πλέγμα διασυνδεδεμένων υπολογιστών και των υπηρεσιών και πληροφοριών που παρέχει στους χρήστες του. Το Διαδίκτυο χρησιμοποιεί μεταγωγή πακέτων (packet switching) και τη στοίβα πρωτοκόλλων TCP/IP.

Οι πρώτες απόπειρες για την δημιουργία ενός διαδικτύου ξεκίνησαν στις ΗΠΑ κατά την διάρκεια του ψυχρού πολέμου. Η Ρωσία είχε ήδη στείλει στο διάστημα τον δορυφόρο Σπούτνικ 1 κάνοντας τους Αμερικανούς να φοβούνται όλο και περισσότερο για την ασφάλεια της χώρας τους. Θέλοντας λοιπόν να προστατευτούν από μια πιθανή πυρηνική επίθεση των Ρώσων δημιούργησαν την υπηρεσία προηγμένων αμυντικών ερευνών ARPA (Advanced Research Project Agency) γνωστή ως DARPA (Defense Advanced Research Projects Agency) στις μέρες μας.

Αποστολή της συγκεκριμένης υπηρεσίας ήταν να βοηθήσει τις στρατιωτικές δυνάμεις των ΗΠΑ να αναπτυχθούν τεχνολογικά και να δημιουργηθεί ένα δίκτυο επικοινωνίας το οποίο θα μπορούσε να επιβιώσει σε μια ενδεχόμενη πυρηνική επίθεση.

Το αρχικό θεωρητικό υπόβαθρο δόθηκε από τον Τζ. Λικλάιντερ (J.C.R. Licklider) που ανέφερε σε συγγράμματά του το "γαλαξιακό δίκτυο". Η θεωρία αυτή υποστήριζε την ύπαρξη ενός δικτύου υπολογιστών που θα ήταν συνδεδεμένοι μεταξύ τους και θα μπορούσαν να ανταλλάσσουν γρήγορα πληροφορίες και προγράμματα. Το επόμενο θέμα που προέκυπτε ήταν ότι το δίκτυο αυτό θα έπρεπε να ήταν αποκεντρωμένο έτσι ώστε ακόμα κι αν κάποιος κόμβος του δεχόταν επίθεση να υπήρχε δίοδος επικοινωνίας για τους υπόλοιπους υπολογιστές.

Τη λύση σε αυτό έδωσε ο Πολ Μπάραν (Paul Baran) με τον σχεδιασμό ενός κατανεμημένου δικτύου επικοινωνίας που

χρησιμοποιούσε την ψηφιακή τεχνολογία. Πολύ σημαντικό ρόλο έπαιξε και η θεωρία ανταλλαγής πακέτων του Λέοναρντ Κλάινροκ (Leonard Kleinrock), που υποστήριζε ότι πακέτα πληροφοριών που θα περιείχαν την προέλευση και τον προορισμό τους μπορούσαν να σταλούν από έναν υπολογιστή σε έναν άλλο. Στηριζόμενο λοιπόν σε αυτές τις τρεις θεωρίες δημιουργήθηκε το πρώτο είδος διαδικτύου γνωστό ως ARPANET.

Εγκαταστάθηκε και λειτούργησε για πρώτη φορά το 1969 με 4 κόμβους μέσω των οποίων συνδέονται 4 μίνι υπολογιστές (mini computers 12k): του πανεπιστημίου της Καλιφόρνια στην Σάντα Μπάρμπαρα του πανεπιστημίου της Καλιφόρνια στο Λος Άντζελες, το SRI στο Στάνφορντ και το πανεπιστήμιο της Γιούτα. Η ταχύτητα του δικτύου έφθανε τα 50 kbps και έτσι επιτεύχθηκε η πρώτη dial up σύνδεση μέσω γραμμών τηλεφώνου. Μέχρι το 1972 οι συνδεδεμένοι στο ARPANET υπολογιστές έχουν φτάσει τους 23, οπότε και εφαρμόζεται για πρώτη φορά το σύστημα διαχείρισης ηλεκτρονικού ταχυδρομείου

1.2.2 **Ο Παγκόσμιος Ιστός (World Wide Web)**

Το διαδίκτυο (Internet) είναι ένα διεθνές δίκτυο υπολογιστών , που επιτρέπει την επικοινωνία μεταξύ ανθρώπων σε όλο τον κόσμο με τη χρήση του πρωτοκόλλου επικοινωνίας TCP/IP. Το World Wide Web , το οποίο αποκαλείται και WWW είναι αναμφισβήτητα μια από τις πιο δημοφιλείς υπηρεσίες μαζί με το ηλεκτρονικό ταχυδρομείο. Η λειτουργία του βασίζεται σε δύο τεχνολογίες: το HTTP (Hypertext Transfer Protocol)και το HTML(Hypertext Markup Language).

Με τη γλώσσα HTML μπορούμε να δημιουργήσουμε μεμονωμένες ιστοσελίδες ή ακόμα και ολόκληρους δικτυακούς τόπους. Οι ιστοσελίδες δημοσιεύονται στο διαδίκτυο με βάση το πρωτόκολλο HTTP.

Οι υπολογιστές αντίστοιχα χρησιμοποιούν μία εφαρμογή πλοήγησης – web browser (όπως το Netscape Navigator ή το Microsoft Internet Explorer) που λαμβάνει, ερμηνεύει και εμφανίζει τις ιστοσελίδες στην οθόνη .

Παγκόσμιος ιστός και Internet συχνά θεωρούνται το ίδιο πράγμα. Η αντίληψη αυτή είναι λανθασμένη καθώς ο ιστός αποτελεί μία μόνο εφαρμογή του Internet. Για την ακρίβεια, την δημοφιλέστερη. Σε αντίθεση με το Internet, που έχει και υλική υπόσταση, ο ιστός δεν έχει, μιας και αποτελείται από πακέτα πληροφορίας.

Η τεχνολογία του ιστού καθιστά δυνατή την δημιουργία "υπερκειμένων", μία διασύνδεση δηλαδή πάρα πολλών μη ιεραρχημένων στοιχείων που παλαιότερα ήταν απομονωμένα. Τα

στοιχεία αυτά μπορούν να πάρουν και άλλες μορφές πέραν της μορφής του γραπτού κειμένου, όπως εικόνας και ήχου.

Η τεχνολογία του ιστού δημιουργήθηκε το 1989 από τον Βρετανό Τιμ Μπέρνερς Λι, που εκείνη την εποχή εργαζόταν στον Ευρωπαϊκό Οργανισμό Πυρηνικών Ερευνών (CERN) στην Γενεύη της Ελβετίας. Το όνομα που έδωσε στην εφεύρεσή του ο ίδιος ο Lee είναι World Wide Web, όρος γνωστός στους περισσότερους από το "www".

Αυτό που οδήγησε τον Lee στην εφεύρεση του Παγκόσμιου ιστού ήταν το όραμά του για ένα κόσμο όπου ο καθένας θα μπορούσε να ανταλλάσσει πληροφορίες και ιδέες άμεσα προσβάσιμες από τους υπολοίπους. Το σημείο στο οποίο έδωσε ιδιαίτερο βάρος ήταν η μη ιεράρχηση των διασυνδεδεμένων στοιχείων.

Οραματίστηκε κάθε στοιχείο, κάθε κόμβο του ιστού ίσο ως προς την προσβασιμότητα με τα υπόλοιπα. Αν σκεφτεί, όμως, κανείς τον 13 βαθμό ιεράρχησης με τον οποίο λειτουργούν οι μηχανές αναζήτησης του ιστού, όπως για παράδειγμα το google, γίνεται εύκολα κατανοητό ότι στην πράξη κάτι τέτοιο δεν συμβαίνει, τουλάχιστον στον βαθμό που το είχε οραματιστεί ο Lee.

1.2.3 **Το πρωτόκολλο HTTP(Hyper Text Transfer Protocol)**

HTTP: Συντομογραφία της φράσης: «Hyper Text Transfer Protocol». Είναι ένα σύνολο κανόνων, ή αλλιώς πρωτόκολλο, που καθορίζει τον τρόπο ε τον οποίο θα γίνει η μεταφορά του υπερκειμένου (hypertext) μεταξύ δύο ή περισσοτέρων υπολογιστών.

Το πρωτόκολλο HTTP είναι το πιο συνηθισμένο στον ηλεκτρονικό χώρο του World Wide Web. Η ονομασία του προέρχεται από τα αρχικά των αγγλικών λέξεων Hyper Text Transfer Protocol (Πρωτόκολλο Μεταφοράς Υπερκειμένου). Το πρωτόκολλο αυτό χρησιμοποιείται από τη συγκεκριμένη υπηρεσία του δικτύου Internet από το 1990.

Το HTTP αποτελεί ένα πρωτόκολλο του επιπέδου εφαρμογών στα δίκτυα υπολογιστών και χρησιμοποιείται κυρίως σε διανεμημένα πληροφορικά συστήματα υπέρ-μέσων.

Είναι ένα γενικό, αντικειμενοστραφές πρωτόκολλο που μπορεί να χρησιμοποιηθεί σε ένα πλήθος εφαρμογών, για παράδειγμα σε εξυπηρετητές-διανομείς (servers) και διανεμημένα συστήματα διαχείρισης αντικειμένων.

Το βασικότερο και πιο σημαντικό ίσως χαρακτηριστικό του πρωτοκόλλου αυτού είναι ότι επιτρέπει στα διάφορα συστήματα μετάδοσης δεδομένων να υφίστανται ανεξάρτητα από τα δεδομένα που αυτά μεταφέρουν

1.3 **Είδη Ιστοσελίδων**

1.3.1 **Στατικές Ιστοσελίδες**

Οι στατικές ιστοσελίδες χαρακτηρίζονται από την μονιμότητα του περιεχομένου τους και της διάταξής τους (layout), τα οποία μπορούν να αλλάξουν/να τροποποιηθούν μόνο με αίτημα για αναβάθμιση (update) από τον προγραμματιστή/διαχειριστή της σελίδας αυτής. Μια απλή σελίδα html (έγγραφο html) που περιλαμβάνει κείμενο, συνδέσμους και φωτογραφίες για παράδειγμα, είναι ένα απλό παράδειγμα στατικής σελίδας. Τα δεδομένα σε μια στατική ιστοσελίδα δεν αλλάζουν δυναμικά. Έχουν σταθερό, αμετάβλητο περιεχόμενο.

Στα υπέρ της στατικής σελίδας είναι το χαμηλό κόστος κατασκευής και συντήρησης, γιατί μια στατική σελίδα, δεν απαιτεί μεγάλο χώρο σε φιλοξενία από webserver. Μια στατική σελίδα αναπτύσσεται/σχεδιάζεται γρηγορότερα από ότι μια δυναμική. Επίσης οι στατικές σελίδες φορτώνουν πιο γρήγορα από ότι οι δυναμικές.

Μεγάλο πλεονέκτημα των στατικών ιστοσελιδών έναντι των δυναμικών είναι επίσης ότι πιο SEO friendly. Το SEO σημαίνει Search engine optimization και στα ελληνικά "Βελτιστοποίηση Σελίδας για τις μηχανές αναζήτησης". Οι στατικές σελίδες λοιπόν είναι πιο φιλικές προς τις μηχανές αναζήτησης. Αυτό συμβαίνει γιατί στις στατικές ιστοσελίδες υπάρχει δυνατότητα τοποθέτησης στον κώδικα html, των meta tags τα οποία αναγνωρίζουν οι μηχανές αναζήτησης και κατατάσσουν την στατική σελίδα πάνω από μια δυναμική που δεν έχει ακριβώς αυτή την δυνατότητα.

Αυτό είναι πολύ ισχυρό πλεονέκτημα των στατικών ιστοσελιδών, γιατί όπως καταλαβαίνετε έχει άμεση σχέση με την επισκεψιμότητά τους. Και η επισκεψιμότητα σε έναν ιστότοπο μπορεί να μεταφραστεί σε κέρδος. Και όπως είναι γνωστό, οι μηχανές αναζήτησης είναι αυτές που παραπέμπουν τους χρήστες στις ιστοσελίδες. Οπότε όσο καλύτερα καταταγμένη η σελίδα στις μηχανές αναζήτησης σε διάφορες λέξεις-κλειδιά, τόσο μεγαλύτερο πλεονέκτημα έχει έναντι των ανταγωνιστών της.

Στην αγορά γίνεται σκληρή μάχη ανάμεσα στους SEO experts για διάφορες λέξεις-κλειδιά. Είναι αξιοσημείωτο, ότι μόνο στην ελληνική αγορά για λέξεις-κλειδιά όπως (fashion, gadgets, ρούχα), για να επιτευχθεί η πρώτη θέση στις μηχανές αναζήτησης, δαπανούνται μέχρι και 20 χιλιάδες ευρώ. Για αυτές τις φράσεις, στην παγκόσμια αγορά τα ποσά εκτοξεύονται στα ύψη όπως είναι αντιληπτό .

1.3.2 **Δυναμικές Ιστοσελίδες**

Οι δυναμικές ιστοσελίδες προσαρμόζουν το περιεχόμενο τους και την εμφάνισή τους σύμφωνα με την καταχώρηση/αλληλεπίδραση ή τις αλλαγές του τελικού χρήστη στο περιβάλλον προγραμματισμού (χρήστης, ώρα, τροποποιήσεις στη βάση δεδομένων κτλ.).

Το περιεχόμενο μπορεί να αλλάζει στον υπολογιστή του τελικού-χρήστη με τη χρήση των γλωσσών προγραμματισμού που εκτελούνται στον υπολογιστή του χρήστη (JavaScript, VBScript, Actionscript, etc.).

Το περιεχόμενο στις δυναμικές σελίδες συχνά μεταφράζεται στον εξυπηρετητή (server), που εκεί αποστέλλεται μέσω του διακομιστή (Apache), μέσω γλωσσών προγραμματισμού που εκτελούνται στον εξυπηρετητή (Perl, PHP, ASP, JSP, ColdFusion, .NET κτλ).

Πίσω από δυναμικά websites κρύβονται πάντα βάσεις δεδομένων (databases) οπού εκεί αποθηκεύονται δεδομένα και πληροφορίες του εν λόγω website.

Και με εντολή του χρήστη, το website επικοινωνεί με την database, από όπου θα αντλήσει το περιεχόμενο που απαίτησε να δει ο χρήστης. Λόγω των databases καθίσταται εύκολη η προσθαφαίρεση περιεχομένου στις δυναμικές ιστοσελίδες, ακόμα και από τον πιο άσχετο (σε γνώσεις προγραμματισμού) χρήστη-επισκέπτη της σελίδας .

Διότι σε μια στατική σελίδα για να αλλάξει ή να τροποποιήσει κανείς το περιεχόμενο της σελίδας θα πρέπει να επέμβει στον κώδικα. Άρα πρέπει να ξέρει προγραμματισμό. Ενώ σε μια δυναμική σελίδα, απλά πρέπει να ξέρει πως να διαχειρίζεται το περιεχόμενο στη βάση δεδομένων και όλα τα υπόλοιπα γίνονται αυτοματοποιημένα από το πρόγραμμα. Όπως είπαμε οι στατικές σελίδες είναι απλά html έγγραφα.

Οι δυναμικές ιστοσελίδες είναι εφαρμογές-προγράμματα. Για την δημιουργία ενός δυναμικού site χρειάζονται πολλές-πολλές γραμμές κώδικα προγραμματισμού.

Παρόλο όμως που είναι σχετικά δύσκολο να κατασκευαστεί ένα δυναμικό site, είναι εύκολο να διαχειριστεί και να ανανεωθεί.

Εδώ είναι και το μεγάλο πλεονέκτημα, έναντι των στατικών. Για να μπορεί ένας απλός χρήστης- επισκέπτης να διαχειρίζεται (να το αλλάζει, να το τροποποιεί ή να το διαγράφει) το περιεχόμενο μιας

δυναμικής σελίδας εύκολα, χωρίς γνώσεις προγραμματισμού, υπάρχουν τα CMS! CMS (Content Managment System) είναι εύχρηστοι μηχανισμοί διαχείρισης περιεχομένου. Έτσι, στην περιοχή διαχείρισης της σελίδας, μπορούν να διαχειριστούν το περιεχόμενό της, απλά μέλη, συντάκτες, διαχειριστές κ.α. Ο καθένας έχει τον δικό του κωδικό πρόσβασης και το τι προνόμια έχει ο καθένας και κατά πόσο μπορεί να αλλάξει το περιεχόμενο της σελίδας, το ορίζει ο υπέρ- διαχειριστής της σελίδας.

1.4. **Μηχανογράφηση – Οργάνωση**

1.4.1 **Ο όρος μηχανογράφηση**

Η μηχανογράφηση αναφέρεται στη χρήση ηλεκτρονικών υπολογιστών για την επεξεργασία διοικητικών και οικονομικών πληροφοριών στο πλαίσιο μιας επιχείρησης ή ενός οργανισμού.

Ειδικότερα τα πεδία εφαρμογής της μ. είναι η μισθοδοσία, οι αποθήκες, ο έλεγχος αποθεμάτων και παραγωγής, η διαχείριση παραγγελιών, η διαχείριση πελατών και προμηθευτών, η πληροφόρηση της διοίκησης για τη λήψη αποφάσεων, η οικονομική μοντελοποίηση κ.ά. Ένα σύστημα ή μια δραστηριότητα μιας επιχείρησης μπορεί να μηχανογραφηθεί, αν οι διαδικασίες του μπορούν να περιγραφούν με ακρίβεια ως μια σειρά από απλά λογικά βήματα, αν η λειτουργία του χαρακτηρίζεται από επαναληπτικές διαδικασίες και αν απαιτείται επεξεργασία μεγάλου όγκου δεδομένων.

Σε ένα μηχανογραφημένο σύστημα διακρίνονται τα δεδομένα, τα οποία είναι κατάλληλα οργανωμένα σε αρχεία ή στις λεγόμενες βάσεις δεδομένων, και οι επεξεργασίες και οι υπολογισμοί που πρέπει να γίνουν πάνω στα δεδομένα για να εξαχθούν τα ζητούμενα αποτελέσματα. Οι επεξεργασίες αυτές υλοποιούνται με τη βοήθεια των προγραμμάτων των ηλεκτρονικών υπολογιστών.

Για τη μ. ενός συστήματος απαιτούνται οι παρακάτω φάσεις:

α) Ανάλυση συστήματος: είναι η βασικότερη φάση στη διαδικασία μ. ενός συστήματος, κατά την οποία συντάσσεται μια μελέτη, με σκοπό να βρεθούν οι στόχοι που θα έχει το μηχανογραφημένο σύστημα και οι πιο πρόσφορες λύσεις και διαδικασίες για την επίτευξή τους, με τη βοήθεια των κατάλληλων μηχανών και ηλεκτρονικών υπολογιστών·

β) σχεδιασμός: εδώ δίνονται αναλυτικές προδιαγραφές όλων των στοιχείων που απαιτούνται για τη μηχανογραφική λύση (εξοπλισμός, αρχεία, προγράμματα)

γ) υλοποίηση: στη φάση αυτή πραγματοποιείται η προμήθεια του εξοπλισμού και των προγραμμάτων και γράφονται και ελέγχονται όλα τα προγράμματα της εφαρμογής με χρήση δοκιμαστικών αρχείων·

δ) εγκατάσταση: γίνεται εγκατάσταση του εξοπλισμού και των προγραμμάτων, δημιουργούνται τα αρχεία δεδομένων και γίνεται δοκιμαστική λειτουργία όλου του συστήματος·

ε) παράδοση: το σύστημα παραδίδεται στους χρήστες του και ακολουθεί πρόγραμμα εκπαίδευσης των χρηστών του στη λειτουργία και στις νέες διαδικασίες·

στ) λειτουργία: είναι η τελευταία φάση, όπου αρχίζει η κανονική λειτουργία του μηχανογραφικού συστήματος, αλλά για να αποφευχθούν τυχόν προβλήματα γίνεται η λεγόμενη παράλληλη ροή κατά την οποία λειτουργεί παράλληλα και το παλιό χειρογραφικό σύστημα.

Έπειτα από κάποιο χρονικό διάστημα, και αφού αξιολογηθούν τα αποτελέσματα της δοκιμαστικής λειτουργίας, επιφέρονται οι απαραίτητες βελτιώσεις και αρχίζει η κανονική λειτουργία του συστήματος.

Για να αναπτυχθούν οι μηχανογραφικές εφαρμογές δεν αρκούν μόνο οι ηλεκτρονικοί υπολογιστές.

Απαιτείται μια σειρά βοηθητικών μηχανημάτων –ψηφιακά αποθηκευτικά μέσα, εκτυπωτές, σκάνερ, σχεδιογράφοι (πλότερ), μόντεμ κ.ά.– καθώς και προσωπικό διαφόρων ειδικοτήτων οργανωμένο κατάλληλα σε μια διοικητική ενότητα που ονομάζεται μηχανογραφικό κέντρο ή υπηρεσία μηχανογράφησης.

1.4.2 **Εφαρμογή μηχανογράφησης**

Η εφαρμογή που αναπτύσεται στην παρούσα διπλωματική εργασία, πέραν από την γενική κατηγορία της διαδικτυακής εφαρμογής, ανήκει και σε μια πιο ειδική, αυτήν των εφαρμογών μηχανογράφισης.

Ένα πρόγραμμα μηχανογράφισης οργανώνει και αυτοματοποιεί καθημερινές λειτουργίες και πληροφορίες μιας επιχείρησης. Προσφέρει δυνατότητα ευκολότερης διαχείρισης των πληροφοριών που σχετίζονται με τα οικονομικά, τις σχέσεις με πελάτες και προμηθευτές. Δίνει στον χρήστη άμεση εικόνα της γενικότερης κατάστασης σε μια επιχείρηση – οργανισμό όπως

διαθεσιμότητα υλικών, ελλείψεις, στατιστικά, οικονομικά στοιχεία, κλπ.

Μια κλασσική εφαρμογή μηχανογράφησης στην πιο απλή της μορφή, παρέχει δυνατότητα σε χρήστες για αλληλεπίδραση σε πραγματικό χρόνο με πληροφορίες που έχουν καταχωρηθεί στην εφαρμογή, ανάλογα πάντα με τα δικαιώματα που έχει παραχωρήσει ο διαχειριστής σε κάθε χρήστη.

Βασικό κοινό χαρακτηριστικό αντίστοιχων εφαρμογών είναι η κεντρική βάση δεδομένων, μέσω της οποίας επιτυγχάνεται η γρήγορη ενημέρωση πληροφοριών έως και σε πραγματικό χρόνο, η διασύνδεση χρηστών, τμημάτων ή ακόμα και επιχειρησεων.

2. **Ανάλυση**

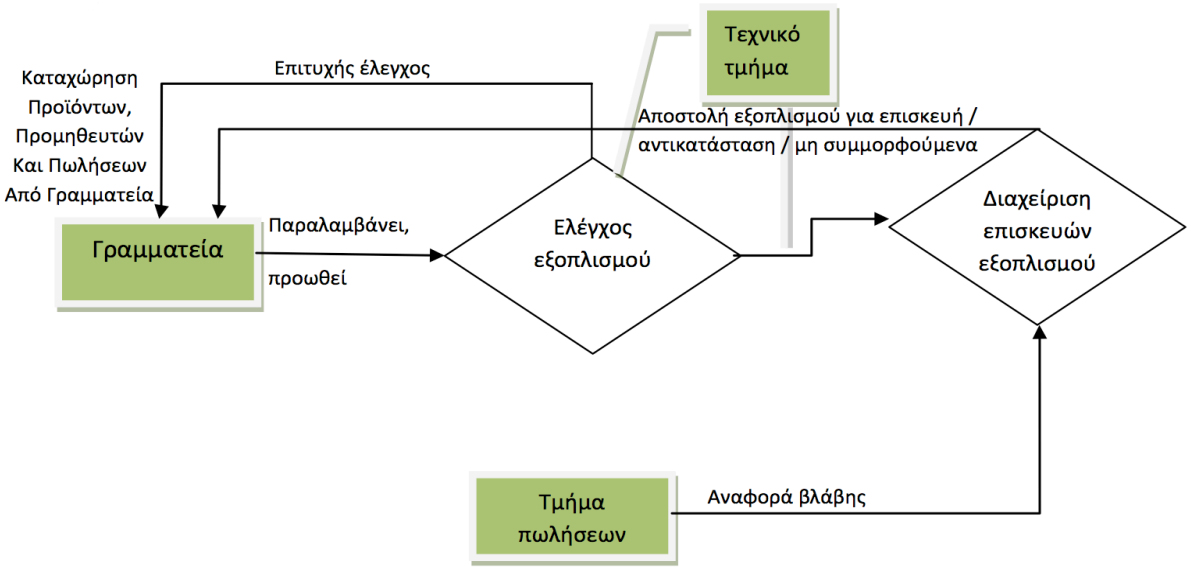
2.1 **Χρήστες εφαρμογής**

Οι χρήστες του συστήματος θα χωρίζονται σε ομάδες και θα είναι οι:

1. Admin: Ο διαχειριστής του συστήματος, μπορεί να καθορίσει αναλυτικά τα δικαιώματα πρόσβασης στο σύστημα για κάθε χρήστη, ενώ το περιβάλλον του χρήστη προσαρμόζεται αυτόματα, με βάση τα δικαιώματα αυτά. Μπορεί να εκτελέσει κάθε δυνατή λειτουργία της εφαρμογής και έχει πρόσβαση σε όλες τις πληροφορίες.
   Γενικά είναι ο άνθρωπος που έχει τα δικαιώματα που απαιτούνται για τη συνεχή ανανέωση της ιστοσελίδας. Διαχειριστής συνήθως σε μια τέτοια ιστοσελίδα μπορεί να
   είναι ένας άνθρωπος χωρίς γνώσεις προγραμματισμού και γι' αυτό όλες οι παραπάνω λειτουργίες εκτελούνται με ειδικές φόρμες, χωρίς την ανάγκη εξειδικευμένων γνώσεων προγραμματισμού. Για να μπορέσει ο διαχειριστής βεβαίως να έχει πρόσβαση σε αυτές
   τις φόρμες, και συνεπώς να ασκήσει τα δικαιώματα του στην ιστοσελίδα, αρκεί να πιστοποιηθεί η ταυτότητα του με τον κατάλληλο ψευδώνυμο διαχειριστή και κωδικό
   πρόσβασης.
2. Manager: Μπορεί να εκτελέσει κάθε δυνατή λειτουργία της εφαρμογής και έχει πρόσβαση σε όλες τις πληροφορίες για όλα τα τμήματα της επιχείρησης -

οργανισμού. Δεν μπορεί να τροποποιήσει δικαιώματα πρόσβασης χρηστών

3. Supervisor: Δυνατότητα εκτέλεσης λειτουργιών και προβολής πληροφοριών σχετικά με το τμήμα που είναι υπεύθυνος.
4. Users: Είναι χρήστες που ανήκουν σε συγκεκριμένα τμήματα, διενεργούν ελέγχους, επισκευές, κλπ.



(εικόνα 2-1, Τρόπος αλληλεπίδρασης χρηστών - τμημάτων)

2.2 **Απαιτήσεις εφαρμογής**

2.2.1 **Απαιτήσεις λειτουργίας**

Υποθέτουμε ότι έχουμε μια διαδικτιακή εφαρμογή η οποία έχει υλοποιηθεί με διάφορα προγραμματιστικά εργαλεία που θα αναλύσουμε παρακάτω.

Για να είναι λειτουργική και προσβάσιμη από χρήστες, πρέπει να φιλοξενείται σε έναν διακομιστή (web server) που διαθέτει σύνδεση στο διαδίκτυο. Απαιτείται ένας server δηλαδή για τα αρχεία της εφαρμογής και ένας για την βάση δεδομένων. Οι δύο αυτοί server μπορούν να βρίσκονται είτε σε διαφορετικά υπολογιστικά συστήματα είτε και στο ίδιο σύστημα.

Από την μεριά του χρήστη απαιτείται ένας υπολογιστής, ο οποίος στις μέρες μας μπορεί να είναι και ένα «έξυπνο» κινητό τηλέφωνο, ένα tablet ή οποιαδήποτε άλλη συσκευή, αρκεί να

διαθέτει ένα περιηγητή ή αλλιώς web browser. Ο web browser είναι ένας client που σε γενικές γραμμές συνδέεται με τον σέρβερ, στην διεύθυνση που βρίσκεται η εφαρμογή και προβάλει κάθε φορά τα αρχεία που στέλνει η εφαρμογή μέσω του server στον χρήστη.

Σε ένα διαδικτυακό περιβάλλον όπως το World Wide Web, αναφερόμαστε στις δύο βασικές πλευρές που παίρνουν μέρος σε οποιαδήποτε αλληλεπίδραση σαν Client και Server. Ο Client αποτελεί τον browser, τον οποίο ένας χρήστης χρησιμοποιεί. Ο Server είναι ένα απομακρυσμένο μηχάνημα στο οποίο υπάρχουν αποθηκεμένα έγγραφα, εικόνες και άλλες διάφορα αρχεία που συνθέτουν μια ιστοσελίδα ή web εφαρμογή. Ο browser του χρήστη στέλνει ένα μήνυμα στο web server ζητώντας ένα document. Ο server λαμβάνει το μήνυμα του browser με το ζητούμενο από τον browser του Client document, και στέλνει πίσω το αρχείο στον browser αφού το διερμηνεύσει.

Πέραν όμως από τα δύο παραπάνω βασικά εργαλεία για την λειτουργία της εφαρμογής μας, απαιτείται ένα όνομα χώρου ή αλλιώς domain name μέσω του οποίου γίνεται η σύνδεση του browser στον server και στην εφαρμογή.

2.2.2 **Απαιτήσεις υλοποίησης**

Για την υλοποίηση διαδικτυακής εφαρμογής, χρειάζονται προγραμματιστικά εργαλεία που θα υλοποιούν το κομμάτι που εκτελείται στον σέρβερ (server-side scripting) και αντίστοιχα τα τμήματα που βρίσκονται στον browser του χρήστη (browser-side scripting).

Σε πρώτη φάση δηλαδή, διαιρούμε την εφαρμογή σε δύο βασικά τμήματα, αυτό που εκτελείται στον σέρβερ και αυτό που εκτελείται στον browser.

2.2.2.1 **Server-side Scripting**

Το server-side scripting αναφέρεται στον προγραμματισμό της συμπεριφοράς του σέρβερ. Κανονικά όταν ένας browser ζητά κάποιο αρχείο από τον σέρβερ, αυτός τον αποστέλει. Όμως αν το αρχείο περιλαμβάνει server-side scripting, πριν ο σέρβερ το αποστείλει στον browser, το script εκτελείται στον σερβερ, παραμετροποιεί το αρχείο βάσει του προγραμματισμού που έχει γίνει και κατόπιν το αποστέλλει.

Η επεξεργασία στην πλευρά του Server(Server-side processing) σημαίνει ότι ο web server, χρησιμοποιώντας το

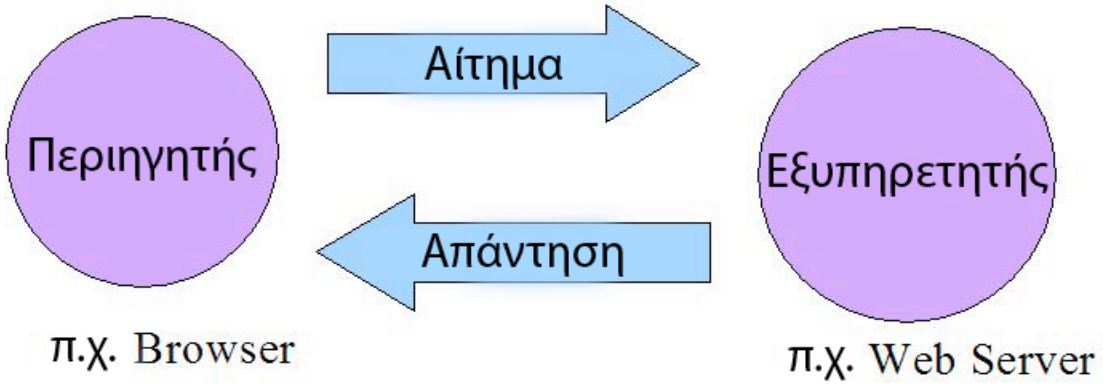
λογισμικό του, παίρνει αποφάσεις βασισμένες σε πληροφορίες που του παρείχε ο Client. Μηνύματα σε μορφή εντολών και δεδομένων αποστέλλονται μέσω Internet ή τοπικού δικτύου από το μηχάνημα του Client. Ο Web Server, εν συνεχεία τρέχει το κατάλληλο λογισμικό.

Τα δεδομένα που προκύπτουν από τη διαδικασία αυτή αποστέλλονται πάλι μέσω Internet ή τοπικού δικτύου πίσω, από τον Server στον Client και παρουσιάζονται από τον Browser.

2.2.2.2 **Browser-side Scripting**

Το browser-side scripting αναφέρεται στον προγραμματισμό της συμπεριφοράς του browser. Σαν επεξεργασία στην πλευρά του Client (Client-side processing) αναφερόμαστε στην επεξεργασία των δεδομένων που γίνεται στον browser του χρήστη. Τα επεξεργασμένα και μη δεδομένα και αρχεία που αποστέλλει ο σέρβερ στον browser μπορούν να υποστούν περαιτέρω επεξεργασία στον browser.

Ο browser αποτελείται από δύο βασικά τμήματα: τη διεπαφή χρήστη (user interface), που είναι το τμήμα που ο χρήστης βλέπει στην οθόνη του υπολογιστή του και το λογισμικό επεξεργασίας που είναι το τμήμα που ο χρήστης δεν βλέπει. Στην περίπτωση αυτή, της επεξεργασίας στην πλευρά του Client, το interface του browser και πάλι πρέπει να στείλει μηνύματα για την επεξεργασία των δεδομένων, αλλά τα στέλνει εσωτερικά σε ένα άλλο τμήμα του λογισμικού του browser, που αναλαμβάνει την επεξεργασία των δεδομένων χωρίς να επικοινωνεί άλλο με τον σερβερ.



(εικόνα 2-2, απλοποιημένη γραφική απεικόνηση της επικοινωνίας μεταξύ server - browser)

2.2.2.3 **Βάση δεδομένων**

Όταν λέμε βάση δεδομένων αναφερόμαστε σε ένα σύνολο απο συστηματικά μορφοποιημένα δεδομένα που σχετίζονται και στα οποία είναι δυνατή η ανάκτηση δεδομένων μέσω αναζήτησης κατ' απαίτηση. Ένας γνωστός σε όλους μας τηλεφωνικός κατάλογος, για παράδειγμα, είμαι μια βάση δεδομένω, καθώς αποθηκεύει και οργανώνει σχετιζόμενα τμήματα πληροφορίας, όπως το όνομα και ο αριθμός τηλεφώνου.

Με τον όρο σχεσιακή βάση δεδομένων, αναφερόμαστε σε ένα σύνολο δεδομένων οργανωμένα σε συσχετιζόμενους πίνακες που παρέχουν ταυτόχρονα μηχανισμούς ανάγνωσης, εγγραφής, τροποποίησης και άλλες διαδικασίες στα δεδομένα. Η Βάση δεδομένων έχει σαν στόχο την οργάνωση της πληροφορίας και την παρωχή μηχανισμών για την εξαγωγή αυτής σε πιο οργανομένη μορφή, όπως για παράδειγμα σε νοητούς πίνακες. Η σχεσιακή βάση δεδομένων επινοήθηκε από τον Έντγκαρ Κοντ το 1970.

Βασικό εργαλείο μέσω του οποίου αλληλεπιδρούμε με την βάση δεδομένων, είναι γλώσσα SQL. Με την SQL ο χρήστης διατυπώνει ερωτήσεις προς τη βάση δεδομένων και ανάλογα με τα δικαιώματα του, μπορεί να δημιουργήσει δεδομένα, να μεταβάλλει και να διαγράψει βάσει και των κριτηρίων αναζήτησης.

Οποιαδήποτε εφαρμογή που θέλει να αποθηκεύει πληροφορία εύκολα προσβάσιμη σε πραγματικό χρόνο από τους χρήστες της, είναι αναγκασμένη να καταφύγει σε μια λύση συστήματος διαχείρισης βάσης δεδομένων(DBMS).

Το DBMS απαιτεί ένα σέρβερ που θα το φιλοξενεί. Μπορεί να βρίσκεται στον ίδιο σέρβερ η σε έναν άλλο από αυτόν της βασικής εφαρμογής.

3. **Σχεδίαση διεπαφών**

3.1 **Βασικά εργαλεία**

Τα εργαλεία ανάπτυξης μιας διαδικτυακής εφαρμογής, χωρίζονται σε δύο γενικές κατηγορίες. Οι κατηγορίες αυτές ονομάζονται Frontend και Backend. Τα δύο αυτά τμήματα της εφαρμογής με χρήση κατάλληλων αρχιτεκτονικών και μεθόδων σχεδιασμού μπορούν να αναπτυχθούν εντελώς ανεξάρτητα και στο τέλος να συνδεθούν και να λειτουργούν ως μια οντότητα. Πολλοί

προγραμματιστές ασχολούνται συνήθως με τμήμα ώστε να προσφέρουν ποιοτικές λύσεις.

3.1.1 **Front End**

Με την έννοια front-end αναφερόμαστε στον τμήμα της εφαρμογής το οποίο βλέπει ο χρήστης και αλληλεπιδρά μαζί της. Το τμήμα αυτό απαιτεί σε πρώτη φάση γραφική σχεδίαση και μετά ανάπτυξη με προγραμματιστικά εργαλεία που θα παρουσιαστούν παρακάτω.

Απαιτούνται εργαλεία για την παρουσίαση της πληροφορίας, την μορφοποίηση της πληροφορίας

Στο τμήμα αυτό ανήκουν οι γραμματοσειρές, τα κουμπιά, οι φόρμες που συμπληρώνουμε, τα διάφορα γραφικά και εφέ κίνησης.

Αυτό που βλέπει ο χρήστης κάθε φορά στο front-end τμήμα της εφαρμογής είναι ένα στιγμιότυπο που έχει δημιουργηθεί από το back-end τμήμα βάσει της εισόδου και των αιτημάτων του χρήστη.

Με λίγα λόγια αποτελεί το περιβάλλον χρήστη. Στην εποχή μας δεν αρκεί μια απλή υλοποίηση του ώστε να καταστεί η εφαρμογή μας λειτουργική. Έχει αποδειχτεί ότι η λανθασμένη προσέγγιση στο κομμάτι αυτό μπορεί να έχει πολύ αρνητικές συνέπειες στην διάδοσή της στους χρήστες αλλά και στις πωλήσεις σε περίπτωση μιας εμπορικής εφαρμογής ανεξάρτητα αν έχει γίνει πολύ καλή υλοποίηση στο back-end κομμάτι. Οι σύγχρονες εταιρίες έχουν ανάγει την διεπαφή χρήστη σε επιστήμη και κάθε μια ακολουθεί την δικιά της φιλοσοφία και χαρακτηριστικα στον σχεδιασμό.

3.1.2 **Back End**

Το δεύτερο αυτό τμήμα της εφαρμογής, αποτελείται από τρία μέρη, τον διακομιστή, την εφαρμογή και την βάση δεδομένων. Τα τρία κομμάτια συνεργάζονται σε κάθε είσοδο του χρήστη και εξάγουν την πληροφορία στην διεπαφή χρήστη.

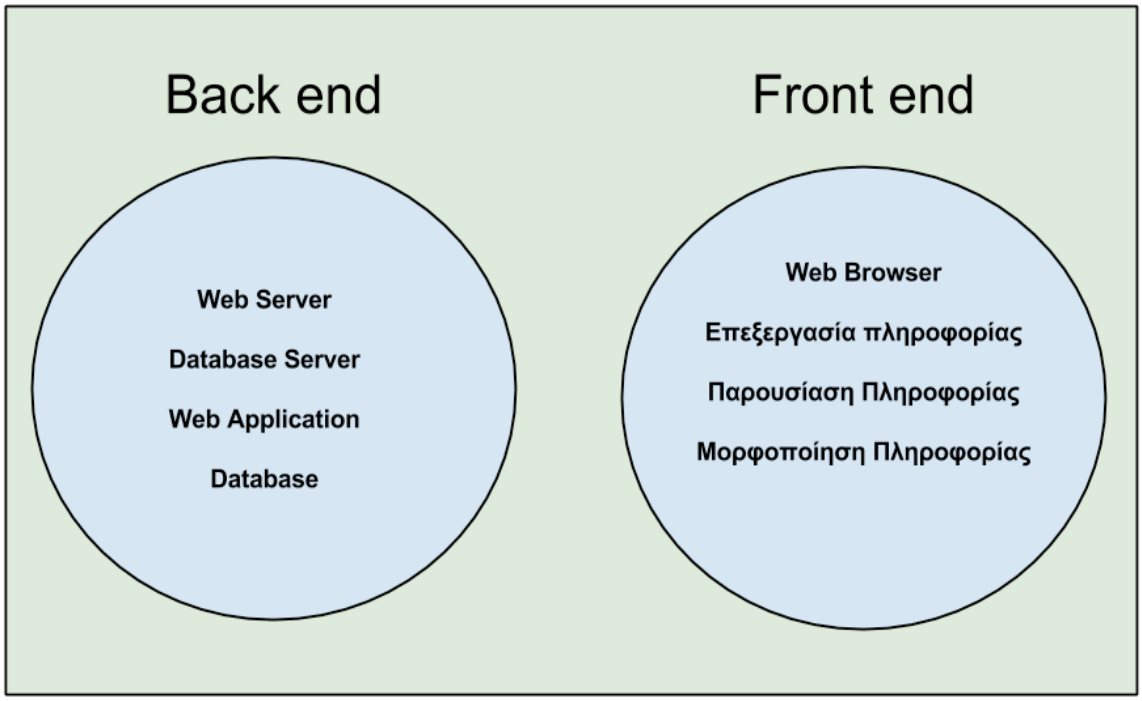
Υπάρχουν πολλά εργαλεία με τα οποία μπορούν να προγραμματιστούν τα παραπάνω τρία κομμάτια. Κάθε ένα εξειδικεύεται ανάλογα με το μέγεθος και την φύση της εφαρμογής.

Ένα παράδειγμα λειτουργίας στο τμήμα αυτό περιλαμβάνει την δρομολόγηση του αιτήματος χρήστη από τον σέρβερ στην κατάλληλη web εφαρμογή. Λήψη του αιτήματος από την εφαρμογή,

επεξεργασία του, αλληλεπίδραση με την βάση δεδομένων και επιστοφή πληροφοριών προς τον χρήστη.

3.2 **Επιλογή εργαλείων**

Κάνοντας την παραπάνω διαφοροποίηση στην εφαρμογή μας έχουμε την παρακάτω γενική εικόνα αυτής και είμαστε σε θέση για την επιλογή των κατάλληλων εργαλείων για υλοποίηση της.



(εικόνα 3-1, γραφική απεικόνηση των χαρακτηριστικών της εφαρμογής που αναφέρθηκαν)

3.2.1 **Διακομιστής εφαρμογής**

Ξεκινώντας τις απαιτήσεις μας από το Back end τμήμα της διαδικτυακής εφαρμογής, πρέπει να επιλέξουμε τον web server που θα την φιλοξενεί.

Ο διακομιστής ή αλλιώς εξυπηρετητής είναι υλικό ή και λογισμικό. Έχει σαν ρόλο την παραχή υπηρεσιών εξυπηρετώντας αιτήσεις άλλων προγραμμάτων συνήθως browser και κατ' επέκταση

χρηστών που βρίσκονται σε σύνδεση με το δίκτυο. Ο σέρβερ λειτουργεί συνεχόμενα 24 ώρες καθώς αυτή είναι και η κύρια λειτουργία του, να εξυπηρετεί κάθε στιγμή. Είναι συνήθως ένας υπολογιστής ή σύνολο υπολογιστών πολύ ηψηλών επιδόσεων και συνδεδεμένο στο δίκτυο ή το διαδίκτυο με πολύ ηψηλές ταχύτητες.

Για την φιλοξενία της εφαρμογής μας επιλέγεται ο Apache 2.0 κυρίως για την σχέση κόστους και αξιοπιστίας που προσφέρει, καθώς και για το γεγονός ότι βρίσκεται σε πλήρη συμβατότητα με τις υπόλοιπες τεχνολογίες που έχουν επιλεχτεί για την ανάπτυξη της διαδικτυακής εφαρμογής.

Ο Apache HTTP ή αλλιώς απλά Apache είναι ο πιο διαδεδομένος web server. Λειτουργεί σε όλες τις πλατφόρμες όπως Windows, Linux, Unix και OS X. Οταν ένας χρήστης επισκέπτεται μια ιστοσελίδα, το πρόγραμμα πλοήγησης επικοινωνεί με τον σέρβερ μέσω του προτοκόλλου HTTP, ο οποίος παράγει την σελίδα βάσει της εισόδου του χρήστη και την αποστέλλει στο browser.

Χρησιμοποιείται και σε τοπικά δίκτυα σαν διακομιστής συνεργαζόμενος με συστήματα διαχείρισης Βάσης Δεδομένων.

Άλλο βασικό πλεονέκτημα του παραπάνω διακομιστή είναι τα γνωστά πλεονεκτήματα που προσφέρονται από την φύση του ανοιχτού κώδικα. Συντηρείται από μια κοινότητα ανοιχτού κώδικα με επιτήρηση από το Ίδρυμα Λογισμικού Apache (Apache Software Foundation)

Είναι ευρέως διαδεδομένος και υπάρχει τεράστια κοινότητα που τον υποστηρίζει. Καλύπτεται από πλήρες documentation στο οποίο αναλύονται όλες οι λειτουργίες και παρουσιάζονται αναλυτικά τρόποι αντιμετώπισης προβλημάτων.

Κυκλοφόρησε για πρώτη φορά το 1993 από τον Robert McCool. Ο ρόλος του ήταν καθοριστικός στην αρχική επέκταση του παγκόσμιου ιστού. Το μεγαλύτερο ποσοστό χρήσης του παρατηρήθηκε το 2006.

3.2.2 **Διακομιστής βάσης δεδομένων**

Διακομιστής βάσης δεδομένων είναι ο σέρβερ που φιλοξενεί μία ή περισσότερες βάσεις δεδομένων. Περιλαμβάνει λογισμικό που καταχωρεί βάσεις δεδομένων στις οποίες μπορούν να έχουν πρόσβαση άλλες εφαρμογές.

Για παράδειγμα μια εφαρμογή ηλεκτρονικού καταστήματος που έχει προϊόντα καταχωρεί όλα τα χαρακτηριστικά αυτών σε μία βάση δεδομένων, έτσι ώστε χρήστες και επισκέπτες της εφαρμογής του να μπορούν να ανακτούν πληροφορίες σχετικά με αυτά. Με λίγα λόγια ο database server προσφέρει τις υπηρεσίες του σε χρήστες ή και άλλους υπολογιστές, στο τοπικό δίκτυο ή στο διαδίκτυο.

Η λειτουργικότητά του προέρχεται από το σύστημα διαχείρισης βάσης δεδομένων που είναι εγκατεστημένο σε αυτό. Το σύστημα αυτό είναι προσβάσιμο και από το front end μιας εφαρμογής καθώς και από το back end που αναλάμβάνει την ανάλυση και την αποθήκευση των δεδομένων.

Τα περισσότερα συστήματα διαχείρισης βάσεων δεδομένων λειτουργούν με μια γλώσσα ερωτημάτων ή αλλιώς Query Language. Το σύστημα που λειτουργεί στην δικιά μας εφαρμογή είναι η MySQL.

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων διαδεδομένο σε όλο τον κόσμο. Βάσει πρόσφατων αναφορών κατατάσσεται ως το καρυφαίο σύστημα διαχείρισης βάσεων δεδομένων. Προσφέρει δυνατότηταν για ανάκτηση εγγραφών, εισαγωγή νέων, διαγραφή, ενημέρωση κλπ. Δεν χαρακτηρίζεται ως μια πλήρης γλώσσα προγραμματισμού αλλά προσφέρει μέσω των λειτουργιών της μια ολοκληρωμένη διαχείριση των σχεσιακών βάσεων δεδομένων. Πρωτοεμφανίζεται το 1974 και από τότε ακολουθεί ανοδική πορεία.

Σημαντικό της χαρακτηριστικό είναι πως αποτελεί εφαρμογή ανοιχτού κώδικα. Το σύστημα διαχείρισης MySQL λοιπόν δίνει τη δυνατότητα αποθήκευσης, αναζήτησης, ταξινόμησης, ομαδοποίησης, ανάκλησης δεδομένων με βάση τη γλώσσα ερωτημάτων SQL. Το γεγονός ότι η MySQL είναι σχεσιακή συνεπάγεται ότι η οργάνωση των δεδομένων γίνεται σε διαφορετικούς πίνακες οι οποίοι σχετίζονται μεταξύ τους με κάποιο σαφώς ορισμένο τρόπο. Η MySQL επιπλέον μπορεί να ελέγχει την πρόσβαση στα δεδομένα, εξασφαλίζοντας έτσι τη δυνατότητα αυτή να γίνεται από διαφορετικούς χρήστες. Κάθε χρήστης έχει συγκεκριμένα δικαιώματα πάνω στις βάσεις δεδομένων που του τα δίνει η MySQL.

Η MySQL χαρακτηρίζεται για την ταχύτητά της. Βάσει πολλών δοκιμών φαίνεται να υπερέχει όλων των αντίστοιχων εφαρμογών στην απόδηση. Ο ανοιχτός κώδικας είναι προσβάσιμος σε όλους για προσωπική χρήση, αλλά και για εμπορικούς σκοπούς το κόστος κινείται σε χαμηλά επίπεδα. Η εκμάθηση της είναι πολύ ευκόλη καθώς είναι μια περιγραφική γλώσσα κοντά στον τρόπο επικοινωνίας του ανθρώπου σε σχέση με άλλες γλώσσες.

Χρησιμοποιείται σε όλα τα μεγάλα λειτουργικά συστήματα όπως windows, linux, unix, osx. Και προσφέρεται η δυνατότητα για εξαγωγή όλων των δεδομένων και μεταφορά τους σε άλλο σύστημα.

3.2.3 **Γλώσσα προγραμματισμού εφαρμογής**

Ως γλώσσα προγραμματισμού της διαδικτυακής εφαρμογής επιλέχτηκε η php. Έχει άριστη συνεργασία με τις υπόλοιπες τεχνολογίες που απαρτίζουν την εφαρμογή και ειναι ανοιχτού κώδικα.

Η PHP, όπου τα αρχικά σημαίνουν Hypertext PreProcessor, είναι μια γλώσσα συγγραφής σεναρίων (scripting language) που ενσωματώνεται μέσα στον κώδικα της HTML και εκτελείται στην πλευρά του server (server-side scripting).

Ανταγωνιστικές της τεχνολογίας PHP είναι οι εξής γλώσσες προγραμματισμού : ASP (Active Server Pages) της εταιρείας Microsoft, CFML (ColdFusion Markup Language) της εταιρείας Allaire και JSP (JavaServer Pages) της εταιρείαςSun.

Το μεγαλύτερο μέρος της σύνταξής της, η PHP το έχει δανειστεί από την C, την Java και την Perl και διαθέτει και μερικά δικά της μοναδικά χαρακτηριστικά. Ο σκοπός της γλώσσας είναι να δώσει τη δυνατότητα στους web developers να δημιουργούν δυναμικά παραγόμενες ιστοσελίδες.

Ακολουθεί ένα εισαγωγικό παράδειγμα :

```
<html>
    <head>
        <title> Παράδειγμα </title>
    </head>
    <body>
        <?php echo "Γειά σου κόσμε!"; ?>
    </body>
</html>
```

Προσέξτε πόσο διαφέρει από ένα CGI script που γράφεται σ' άλλες γλώσσες, όπως η Perl ή η C, όπου αντί να γράψουμε ένα πρόγραμμα με πολλές εντολές για να δημιουργήσουμε κώδικα HTML, γράφουμε ένα HTML script με κάποιον ενσωματωμένο κώδικα για να κάνει κάτι, όπως στην συγκεκριμένη περίπτωση να εμφανίσει κάποιο κείμενο (μήνυμα). Ο κώδικας της PHP περικλείεται με ειδικά tags αρχής και τέλους για να μπορούμε να εισερχόμαστε και να εξερχόμαστε από το PHP mode.

Αυτό που ξεχωρίζει την PHP από μια γλώσσα όπως η JavaScript, η οποία εκτελείται στην πλευρά του χρήστη (client-side), είναι ότι ο κώδικάς της εκτελείται στον server. Αν είχαμε σ' έναν server ένα script παρόμοιο με το παραπάνω, ο χρήστης (client) θα λάμβανε το αποτέλεσμα της εκτέλεσης αυτού του script, χωρίς να είναι σε θέση να γνωρίζει ποιος μπορεί να είναι ο αρχικός κώδικας.

Μπορούμε ακόμη να ρυθμίσουμε (configure) τον web server ώστε να επεξεργάζεται όλα τα HTML αρχεία με τηνPHP και τότε δεν θα υπάρχει πράγματι κανένας τρόπος να μάθουν οι χρήστες τον κώδικά μας.

Στο πιο βασικό επίπεδο, η PHP μπορεί να κάνει ό,τι και τα άλλα προγράμματα της τεχνολογίας CGI, όπως επεξεργασία των δεδομένων μιας φόρμας, δημιουργία δυναμικού περιεχομένου ιστοσελίδων ή αποστολή και λήψη cookies.

Ίσως το δυνατότερο και πιο σημαντικό χαρακτηριστικό της PHP είναι η υποστήριξη που παρέχει σε μια ευρεία γκάμα από βάσεις δεδομένων. Έτσι, το να δημιουργήσουμε μια ιστοσελίδα που να παρέχει υποστήριξη σε βάσεις δεδομένων είναι απίστευτα απλό.

Η ιδέα για την δημιουργία της PHP ελήφθη το φθινόπωρο του 1994 από τον Rasmus Lerdorf. Οι πρώτες ανεπίσημες εκδόσεις (versions) της PHP χρησιμοποιήθηκαν στην αρχική του σελίδα (home page) για να μπορεί να παρακολουθεί αυτούς που έμπαιναν στην σελίδα. Η πρώτη έκδοση που δόθηκε για χρήση στο κοινό ήταν διαθέσιμη στις αρχές του 1995 με το όνομα Personal Home Page Tools.

Αποτελείτο από μια πολύ απλοϊκή μηχανή ανάλυσης (parser engine) η οποία καταλάβαινε λίγες μόνο ειδικές μακροεντολές (macros) και έναν αριθμό από utilities που βρίσκονταν σε κοινή χρήση στις home pages εκείνη την εποχή. Ένα guestbook, ένας μετρητής (counter) και κάποιο άλλο υλικό. Ο αναλυτής (parser) ξαναγράφηκε στα μέσα του 1995 και ονομάστηκε PHP/FI Version 2.

Το όνομα FI προέρχεται από ένα άλλο πακέτο που είχε γράψει ο Rasmus και το οποίο διερμήνευε (interpreted) τα δεδομένα από φόρμες της HTML. Συνδύασε τα εργαλεία scripts της Personal Home Page με τον Form Interpreter και πρόσθεσε υποστήριξη για mSQL. Έτσι γεννήθηκε η PHP/FI, η οποία αναπτύχθηκε αλματωδώς και διάφοροι χρήστες άρχισαν να συνεισφέρουν κώδικα σ' αυτήν.

Ο τύπος δεδομένων μιας μεταβλητής δεν ορίζεται συνήθως από τον προγραμματιστή αλλά αποφασίζεται την ώρα εκτέλεσης (runtime) από την PHP ανάλογα με το περιβάλλον (context) στο οποίο χρησιμοποιείται η μεταβλητή.

Οι εντολές στην PHP τερματίζονται με τον ίδιο τρόπο όπως στην C και την Perl, δηλ. μ' έναν χαρακτήρα ; (semicolon). Μπορούμε, όμως, να δηλώσουμε το τέλος μιας εντολής και με το tag κλεισίματος (closing tag) ?>.

3.2.3.1 **Βασικά στοιχεία της PHP**

Η Εντολή "echo"

Η εντολή echo χρησιμοποιείται για να στείλουμε ένα κείμενο (string) στον φυλλομετρητή (browser). Όλες οι εντολές της Php πρέπει να τελειώνουν με τον χαρακτήρα ; και μια εντολή μπορεί να επεκταθεί και σε περισσότερες από μία γραμμές.

Όταν ο φυλλομετρητής ενός χρήστη ζητήσει μια σελίδα Php σαν την παραπάνω, ο server θα την επεξεργαστεί, θα μετατρέψει τον κώδικα Php σε καθαρή HTML μορφή και έτσι ο χρήστης δεν θα μπορέσει να δει τον αρχικό κώδικα Php.

Μεταβλητές

Στην Php μπορούμε να χρησιμοποιήσουμε και μεταβλητές (variables) για να αποθηκεύουμε και να ανακτούμε δεδομένα που χρησιμοποιούμε συχνά.

Όλες οι μεταβλητές πρέπει να αρχίζουν με τον χαρακτήρα $ και ένα απλό παράδειγμα κώδικα Php που χρησιμοποιεί μεταβλητές και εμφανίζει το ίδιο αποτέλεσμα με το πρώτο παράδειγμα είναι το εξής :

```
< ?
        $string="Hello";
         echo “”;
        echo "$string";
        echo "";38
?>
```

Η Php και οι Φόρμες (Forms)

Ένα από τα ισχυρότερα χαρακτηριστικά της Php είναι ο τρόπος που χειρίζεται τις φόρμες της HTML. Όλα τα στοιχεία μιας φόρμας δημιουργούν μια μεταβλητή με το ίδιο όνομα.

Θα δούμε ένα απλό παράδειγμα που περιέχει μια φόρμα σαν την ακόλουθη :

```
<form action="action.php" method="post">

 Όνομα : <input name="name" type="text">
 Ηλικία : <input name="age" type="text"> <input type="submit">

</form>
```

Όταν ο χρήστης καταχωρήσει κάποια στοιχεία σ’ αυτή τη φόρμα και κάνει κλικ στο πλήκτρο Submit για να την υποβάλλει, θα κληθεί η σελίδα action.php, μέσα στην οποία θα μπορούμε να επεξεργαστούμε τις μεταβλητές για το όνομα και την ηλικία, ως εξής :

```
Γεια σου < ?php echo $name; ?>
```

Είσαι < ?php echo $age; ?> ετών.

Οι Δομές Ελέγχου (Control Structures)

Η PHP, όπως όλες οι γλώσσες προγραμματισμού, παρέχει δυνατότητες για να επηρεάσουμε τη ροή ελέγχου (flow of control) σ' ένα script, δηλ. περιέχει ειδικές 39 εντολές που μας επιτρέπουν να παρεκκλίνουμε από τη σειριακή σειρά εκτέλεσης των εντολών που έχουμε δει μέχρι τώρα. Αυτές οι εντολές αποκαλούνται δομές ελέγχου (control structures).

Η βασικότερη και πιο συχνά χρησιμοποιούμενη δομή ελέγχου είναι η εντολή if-else, η σύνταξη της οποίας είναι η εξής :

```
if ( <συνθήκη> ) {

// Εντολή(ές) που θα εκτελεσθούν αν η <συνθήκη>
// είναι αληθής (true) } else {
// (Προαιρετικές) Εντολές που θα εκτελεσθούν
// αν η <συνθήκη> είναι ψευδής (false)

}
```

Αυτή η δομή ελέγχου μάς δίνει τη δυνατότητα να πούμε στην PHP να εκτελέσει ένα σύνολο εντολών ή κάποιο άλλο ανάλογα με το αν κάποια συνθήκη είναι true ή false.

Ο Βρόχος While

Μια άλλη χρήσιμη δομή ελέγχου (control structure) της PHP είναι ο βρόχος while. Ενώ η εντολή if-else μάς δίνει τη δυνατότητα να επιλέξουμε αν θα εκτελέσουμε ή όχι ένα σύνολο εντολών ανάλογα με την τιμή επιστροφής μιας συνθήκης, ο βρόχος while μάς δίνει τη δυνατότητα να χρησιμοποιήσουμε μια συνθήκη για να καθορίσουμε πόσες φορές θα εκτελεσθεί επανειλημμένα ένα σύνολο εντολών. Η σύνταξη του βρόχου while είναι η εξής :

```
while ( <συνθήκη> ) {

// εντολές που θα εκτελούνται συνέχεια
// για όσο διάστημα η <συνθήκη> παραμένει αληθής (true)

 }
```

3.2.3.2 **Έλεγχος Συνόδων στην PHP**

Το HTTP είναι ένα πρωτόκολλο χωρίς κατάσταση. Αυτό σημαίνει ότι το πρωτόκολλο δεν έχει ενσωματωμένο τρόπο να διατηρεί την κατάσταση μεταξύ δύο συναλλαγών.

Όταν ένας χρήστης ζητά μια σελίδα, ακολουθούμενη από μια άλλη , το HTTP δεν παρέχει ένα τρόπο να μας πει ότι οι δυο αιτήσεις ήρθαν από τον ίδιο χρήστη.

Η ιδέα του ελέγχου συνόδων λειτουργίας είναι για να μπορούμε να παρακολουθούμε ένα χρήστη στη διάρκεια μιας συνόδου λειτουργίας του , σε μια WEB τοποθεσία.

Αν μπορούμε να το κάνουμε αυτό , μπορούμε εύκολα να υποστηρίξουμε σύνδεση ενός χρήστη και εμφάνιση περιεχομένων σύμφωνα με το επίπεδο πιστοποίησης ή των προσωπικών προτιμήσεων του. Μπορούμε να παρακολουθήσουμε τη συμπεριφορά του χρήστη. Επίσης, μπορούμε να χειριστούμε καλάθια αγορών.

Οι σύνοδοι λειτουργίας στην PHP καθοδηγούνται από ένα μοναδικό κωδικό συνόδου, ένα κρυπτογραφικά τυχαίο αριθμό. Ο κωδικός της συνόδου δημιουργείται από την PHP και αποθηκεύεται στην πλευρά του πελάτη κατά τη διάρκεια της συνόδου.

Μπορεί να αποθηκευθεί είτε στον υπολογιστή ενός χρήστη σε ένα cookie ή να περάσει μέσω των URL. Ο κωδικός συνόδου ενεργεί ως ένα κλειδί που μας επιτρέπει να εγγράφουμε συγκεκριμένες μεταβλητές συνόδων λειτουργίας.

Τα περιεχόμενα αυτών των μεταβλητών αποθηκεύονται στον διακομιστή. Ο κωδικός συνόδου είναι η μόνη ορατή πληροφορία στην πλευρά του πελάτη. Αν, στη διάρκεια μιας συγκεκριμένης σύνδεσης στην τοποθεσία μας , ο κωδικός συνόδου είναι ορατός είτε μέσω ενός cookie είτε μέσω του URL, μπορούμε να έχουμε πρόσβαση στις μεταβλητές συνόδου που είναι αποθηκευμένες στον 41 διακομιστή για αυτήν τη σύνοδο. Εξ ορισμού , οι μεταβλητές συνόδου αποθηκεύονται σε επίπεδα αρχεία στον διακομιστή. Πριν χρησιμοποιήσουμε μια σύνοδο , θα πρέπει να την ξεκινήσουμε. Ο απλούστερος τρόπος , είναι να ξεκινήσουμε ένα script με μια κλήση στη συνάρτηση : session_start();

Αυτή η συνάρτηση ελέγχει αν υπάρχει ήδη ένα τρέχον ID συνόδου. Αν όχι θα δημιουργήσει ένα. Αν υπάρχει ένα , θα φορτώσει τις εγγεγραμμένες μεταβλητές συνόδου , ώστε να μπορούμε αργότερα να τις χρησιμοποιήσουμε.

Για να μπορεί μια μεταβλητή να παρακολουθείτε από διάφορα script, ια πρέπει να την εγγράψουμε με μια κλήση στην session_register() . Για παράδειγμα , για να εγγράψουμε μια μεταβλητή $myvar , θα μπορούσαμε να γράψουμε τον παρακάτω κώδικα :

```
$myvar = 5 ;
```

session_register(“myvar”);

Αυτό θα εγγράψει το όνομα της μεταβλητής και θα παρακολουθεί την τιμή της. Η μεταβλητή θα παρακολουθείται μέχρι να τερματιστεί η σύνοδος ή μέχρι να καταργήσουμε την εγγραφή της. Για να φέρουμε μια μεταβλητή συνόδου στο πεδίο δράσης ώστε να μπορούμε να την χρησιμοποιήσουμε θα πρέπει πρώτα να έχουμε ξεκινήσει μια σύνοδο.

Μπορούμε να ελέγξουμε αν η μεταβλητής μας είναι εγγεγραμμένη μεταβλητή συνόδου καλώντας την συνάρτηση session_is_registered() .

$result = session_is_registered($myvar)

Αυτή θα επιστρέψει true ή false ανάλογα. Όταν τελειώσουμε με μια μεταβλητή συνόδου, μπορούμε να την ακυρώσουμε γράφοντας :

session_unregister($myvar);

Όταν τελειώσουμε με μια σύνοδο , θα πρέπει να πρώτα ακυρώσουμε όλες τις εγγεγραμμένες μεταβλητές και μετά να καλέσουμε την συνάρτηση :

session_destroy();

3.2.4 **Παρουσίαση πληροφορίας (Mark Up Language)**

Φυσικά ως γλώσσα παρουσίασης της πληροφορίας χρησιμοποιείται η html (**H**yper**T**ext **M**arkup **L**anguage). Ο φυλλομετρητής (Web browser) παίρνει τις πληροφορίες από τον Web server, τις μορφοποιεί και τις εμφανίζει κατάλληλα για το σύστημά μας. Διαφορετικά προγράμματα φυλλομετρητή μπορεί να μορφοποιούν και να εμφανίζουν το ίδιο αρχείο με διαφορετικό τρόπο, ανάλογα με τις δυνατότητες του συστήματος στο οποίο τρέχουν.
και τις επιλογές διαμόρφωσης του προγράμματος του φυλλομετρητή.

Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες (tags), οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας.

Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα <h1> και </h1>), με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης (ή σε άλλες περιπτώσεις

ετικέτα ανοίγματος και ετικέτα κλεισίματος αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.

Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και τα συνθέτει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να ερμηνεύσει το περιεχόμενο της σελίδας.

Τα στοιχεία της HTML χρησιμοποιούνται για να κτίσουν όλους του ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα, και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες.

Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML.

Οι Web browsers μπορούν επίσης να αναφέρονται σε στυλ μορφοποίησης CSS για να ορίζουν την εμφάνιση και τη διάταξη του κειμένου και του υπόλοιπου υλικού. Ο οργανισμός W3C, ο οποίος δημιουργεί και συντηρεί τα πρότυπα για την HTML και τα CSS, ενθαρρύνει τη χρήση των CSS αντί διαφόρων στοιχείων της HTML για σκοπούς παρουσίασης του περιεχομένου.

Προέλευση

Το 1980, ο φυσικός Τιμ Μπέρνερς Λι, ο οποίος εργαζόταν στο CERN, επινόησε το ENQUIRE, ένα σύστημα χρήσης και διαμοιρασμού εγγράφων για τους ερευνητές του CERN, και κατασκεύασε ένα πρωτότυπό του. Αργότερα, το 1989, πρότεινε ένα σύστημα βασισμένο στο διαδίκτυο, το οποίο θα χρησιμοποιούσε υπερκείμενο.

Έτσι, έφτιαξε την προδιαγραφή της HTML και έγραψε τον browser και το λογισμικό εξυπηρετητή στα τέλη του 1990. Τον ίδιο χρόνο, ο Μπέρνερς Λι και ο μηχανικός συστημάτων πληροφορικής του CERN Robert Cailliau συνεργάστηκαν σε μια κοινή προσπάθεια εύρεσης χρηματοδότησης, αλλά το έργο δεν υιοθετήθηκε ποτέ επίσημα από το CERN. Στις προσωπικές του σημειώσεις από το 1990.

Εξέλιξη

Η HTML 4.0 ήταν μια μεγάλη εξέλιξη των προτύπων της HTML και δίνει ιδιαίτερη έμφαση στη διεθνοποίηση και την υποστήριξη της

HTML για την παρουσίαση της νέας γλώσσας, των πολλαπλών φύλλων στυλ (CSS). . Η HTML 4.0 είχε προταθεί από το W3C το Δεκέμβριο του'97 και έγινε το επίσημο πρότυπο τον Απρίλιο του 1998. Το νέο πρότυπο, τι νέες ετικέτες και τα χαρακτηριστικά υποστήριξε άψογα ο Microsoft Internet Explorer browser, εν αντιθέσει με τον Netscape Navigator 4.7 που δεν ήταν έτοιμος να δεχθεί το νέο πρότυπο της HTML.

Η έκδοση αυτή της HTML, εκτός από το κείμενο, πολυμέσων, και υπερ-σύνδεση χαρακτηριστικά των προηγούμενων εκδόσεων της HTML, υποστηρίζει περισσότερες δυνατότητες πολυμέσων, γλώσσες προγραμματισμού, δελτία τύπου, καλύτερες εκτυπώσεις, καθώς και τα έγγραφα που έχουν μεγαλύτερη πρόσβαση οι χρήστες με ειδικές ανάγκες, ήταν ένα μεγάλο βήμα προς την διεθνοποίηση των εγγράφων.

3.2.5 **Μορφοποίηση πληροφορίας**

CSS (Cascading Style Sheets)

Πρωτοεμφανίστηκε το 1996, αλλά η ολοκληρωμένη υποστήριξη Από τους πιο δημοφιλείς browsers έγινε μέχρι το 2000.
Είναι μία γλώσσα που
χρησιμοποιείται από την HTML και την XHTML για να
ορίσει την
εμφάνιση των
ιστοσελίδων του διαδικτύου. Εφαρμόζεται σε κάθε στοιχείο
της σελίδας
ξεχωριστά, με τα στυλ ορίζουμε το χρώμα, το μέγεθος της
γραμματοσειράς,
την γραφή (bold,underline, κτλ.), το χρώμα του φόντου,
τις
διαστάσεις, την τιμή και μια σειρά από άλλες ιδιότητες των στοιχείω
ν
μιας ιστοσελίδας. Η χρήση CSS κάνει πολύ εύκολη την διαχείριση της

εμφάνισης των σελίδων.

Για παράδειγμα, αν σε μια ιστοσελίδα, χωρίς χρήση CSS, θέλου με να αλλάξουμετο χρώμα του φόντου σε όλες τις επικεφαλίδες ό λων των
πινάκων θαπρέπει να πηγαίνουμε σε κάθε μια επικεφαλίδα και να ορίζουμε το χρώμα που επιθυμούμε αλλάζοντας κάθε φορά την τιμή της ιδιότητας bgcolor των ετικετών <th>. Ενώ σε μια σελίδα με χρήση CSS θα χρειαστεί να αλλάξουμε το χρώμα αυτό μια μόνο φορά και

αυτό θα εφαρμοστεί για όλες τις επικεφαλίδες των πινάκων της σελί δας.
Παρόμοια μπορούμε να ορίσουμε
στυλ, όχι μόνο για στοιχεία μιας σελίδας,
αλλά για στοιχεία όλων των σελίδων του Site μας.

Σήμερα υπάρχουν πολύ λίγες ιστοσελίδες που δεν χρησιμοποιούν CSS. Η χρήση των στυλ κάνει την ζωή των Web designers πολύ πιο εύκολη δημιουργώντας έτσι σελίδες – εφαρμογές που μπορούν να διαχειρίζονται εύκολα και γρήγορα.

ΣΥΝΤΑΞΗ ΤΗΣ CSS

Η σύνταξη των CSS αποτελείται από τρία μέρη : έναν επιλογέα (selec tor),
μια ιδιότητα (property) και μια τιμή (value) :

επιλογέας {ιδιότητα: τιμή}
selector {property: value}
Ο επιλογέας είναι συνήθως το στοιχείο/tag που θέλουμε να ορίσο υμε,
η ιδιότητα είναι το χαρακτηριστικό που θέλουμε να αλλάξουμε και η κάθε
ιδιότητα μπορεί να πάρει μια τιμή. Η ιδιότητα και η τιμή ξεχωρίζουν από
τον χαρακτήρα : και περικλείονται από τους χαρακτήρες {
}, ως εξής :

body {color: black}

Αν η τιμή αποτελείται από πολλές λέξεις, πρέπει να τοποθετήσο υμε
εισαγωγικά :

p {font-family: "sans serif"}

Αν θέλουμε να ορίσουμε περισσότερες από μία ιδιότητες, πρέπει να
ξεχωρίσουμε την κάθε ιδιότητα με τον χαρακτήρα ;. Το παρακάτ ω
παράδειγμα δείχνει πώς μπορούμε να ορίσουμε μια κεντραρισμέν η
παράγραφο με χρώμα κειμένου κόκκινο :

p {text-align: center; color: red}

Για να κάνουμε τους ορισμούς των στυλ πιο ευανάγνωστους, μπορο
ύμε
να γράψουμε από μία ιδιότητα σε κάθε γραμμή, ως εξής :

```
p
{
text-align: center;
color: black;
font-family: arial
}
```

ΒΑΣΙΚΑ ΤΗΣ CSS

Ομαδοποίηση (Grouping)
Μπορούμε να ομαδοποιήσουμε τους επιλογείς. Ξεχωρίζουμε τον κάθε
επιλογέα με κόμμα. Στο παρακάτω παράδειγμα έχουμε ομαδοποι
ήσει
όλα ταστοιχεία επικεφαλίδας (headerelements).
Το κάθε στοιχείο επικεφαλίδας θα είναι πράσινο :

```
h1, h2, h3, h4, h5, h6
{
   color: green
}
```

Το Χαρακτηριστικό (Attribute) Class

Με το χαρακτηριστικό class μπορούμε να ορίσουμε διαφορετικά στυ
λ
για το ίδιο στοιχείο (element). Ας υποθέσουμε ότι θέλουμε να έχ
ουμε δύο είδη παραγράφων στο έγγραφό μας :
μια δεξιά στοιχισμένη παράγραφο και μια κεντραρισμένη παράγρ
αφο.
Να πώς μπορούμε να το κάνουμε αυτό με τα στυλ :

```
p.right {text-align: right}
p.center {text-align: center}
```

Πρέπει να χρησιμοποιήσουμε το χαρακτηριστικό class στο HTML
έγγραφο, ως εξής :

```
<p class="right"> Αυτή είναι μια παράγραφος.
Το κείμενο αυτής της παραγράφου θα είναι δεξιά στοιχισμένο. </p>
<p class="center"> Αυτή είναι μια άλλη παράγραφος.
Το κείμενο αυτής της παραγράφου θα είναι κεντραρισμένο. </p>
```

Το Χαρακτηριστικό (Attribute) Id

Με το χαρακτηριστικό id μπορούμε να ορίσουμε ένα μοναδικό στ υλ
που μπορούμε να χρησιμοποιήσουμε σε πολλά στοιχεία.
Να πώς μπορούμε να το κάνουμε αυτό με τα στυλ :

#right {text-align: right}
Στο HTML έγγραφο πρέπει να γράψουμε τα εξής :

 Αυτή είναι μια παράγραφος.
Το κείμενο αυτής της παραγράφου θα είναι δεξιά στοιχισμένο.
<h3 id="right"> Αυτή είναι μια επικεφαλίδα.
Αυτή η επικεφαλίδα θα είναι επίσης δεξιά στοιχισμένη. </h3>
Το χαρακτηριστικό id πρέπει να έχει μια μοναδική τιμή στο έγγραφο.

3.2.6 **Επεξεργασία πληροφοριών στον περιηγητή (browser-side scripting)**

Το εργαλείο με το οποίο θα γίνεται η επιπλέον επεξεργασία των δεδομένων στον περιηγητή είναι η javascript.

Η JavaScript (JS) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές.

Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.

Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototype-based), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης.

Η σύνταξή της είναι επηρεασμένη από τη C. Η JavaScript αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη Java, αλλά γενικά οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία.

Οι βασικές αρχές σχεδιασμού της JavaScript προέρχονται από τις γλώσσες προγραμματισμού Self και Scheme.Είναι γλώσσα βασισμένη σε διαφορετικά προγραμματιστικά παραδείγματα (multi-paradigm), υποστηρίζοντας αντικειμενοστρεφές,προστακτικό και συναρτησιακό στυλ προγραμματισμού.

Η JavaScript χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων. Τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers) και οι μικρές

εφαρμογές της επιφάνειας εργασίας (desktop widgets). Οι νεότερες εικονικές μηχανές και πλαίσια ανάπτυξης για JavaScript (όπως το Node.js) έχουν επίσης κάνει τη JavaScript πιο δημοφιλή για την ανάπτυξη εφαρμογών Ιστού στην πλευρά του διακομιστή (server-side).

Το πρότυπο της γλώσσας κατά τον οργανισμό τυποποίησης ECMA ονομάζεται ECMAscript.

Ιστορία

Η γλώσσα προγραμματισμού JavaScript δημιουργήθηκε αρχικά από τον Brendan Eich της εταιρείας Netscape με την επωνυμία Mocha. Αργότερα, Mocha μετονομάστηκε σε LiveScript, και τελικά σε JavaScript, κυρίως επειδή η ανάπτυξή της επηρεάστηκε περισσότερο από τη γλώσσα προγραμματισμού Java.

LiveScript ήταν το επίσημο όνομα της γλώσσας όταν για πρώτη φορά κυκλοφόρησε στην αγορά σε βήτα (beta) εκδόσεις με το πρόγραμμα περιήγησης στο Web, Netscape Navigator εκδοχή 2.0 τον Σεπτέμβριο του 1995. LiveScript μετονομάστηκε σε JavaScript σε μια κοινή ανακοίνωση με την εταιρεία Sun Microsystems στις 4 Δεκεμβρίου, 1995, όταν επεκτάθηκε στην έκδοση του προγράμματος περιήγησης στο Web, Netscape.

Η JavaScript απέκτησε μεγάλη επιτυχία ως γλώσσα στην πλευρά του πελάτη (client-side) για εκτέλεση κώδικα σε ιστοσελίδες, και περιλήφθηκε σε διάφορα προγράμματα περιήγησης στο Web. Κατά συνέπεια, η εταιρεία Microsoft ονόμασε την εφάρμογή της σε JScript για να αποφύγει δύσκολα θέματα εμπορικών σημάτων. JScript πρόσθεσε νέους μεθόδους για να διορθώσει τα Y2K-προβλήματα στην JavaScript, οι οποίοι βασίστηκαν στην τάξη της JAVA.

Η JavaScript έχει γίνει μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού ηλεκτρονικών υπολογιστών στον Παγκόσμιο Ιστό (Web). Αρχικά, όμως, πολλοί επαγγελματίες προγραμματιστές υποτίμησαν τη γλώσσα διότι το κοινό της ήταν ερασιτέχνες συγγραφείς ιστοσελίδων και όχι επαγγελματίες προγραμματιστές (και μεταξύ άλλων λόγων). Με με τη χρήση της τεχνολογίας Ajax, η JavaScript γλώσσα επέστρεψε στο προσκήνιο και έφερε πιο επαγγελματική προσοχή προγραμματισμού. Το αποτέλεσμα ήταν ένα καινοτόμο αντίκτυπο στην εξάπλωση των πλαισίων και των βιβλιοθηκών, τη βελτίωση προγραμματισμού με JavaScript, καθώς και αυξημένη χρήση της JavaScript έξω από τα προγράμματα περιήγησης στο Web.

Τον Ιανουάριο του 2009, το έργο CommonJS ιδρύθηκε με στόχο τον καθορισμό ενός κοινού προτύπου βιβλιοθήκης κυρίως για την ανάπτυξη της JavaScript έξω από το πρόγραμμα περιήγησης και μέσα σε άλλες τεχνολογίες (π.χ. server-side).

Μοντέλο εκτέλεσης

Η αρχική έκδοση της Javascript βασίστηκε στη σύνταξη στη γλώσσα προγραμματισμού C, αν και έχει εξελιχθεί, ενσωματώνοντας πια χαρακτηριστικά από νεότερες γλώσσες.

Αρχικά χρησιμοποιήθηκε για προγραμματισμό από την πλευρά του πελάτη (client), που ήταν ο φυλλομετρητής (browser) του χρήστη, και χαρακτηρίστηκε σαν client-side γλώσσα προγραμματισμού. Αυτό σημαίνει ότι η επεξεργασία του κώδικα Javascript και η παραγωγή του τελικού περιεχομένου HTML δεν πραγματοποιείται στο διακομιστή, αλλά στο πρόγραμμα περιήγησης των επισκεπτών, ενώ μπορεί να ενσωματωθεί σε στατικές σελίδες HTML. Αντίθετα, άλλες γλώσσες όπως η PHP εκτελούνται στο διακομιστή (server-side γλώσσες προγραμματισμού).

Παρά την ευρεία χρήση της Javascript για συγγραφή προγραμμάτων σε περιβάλλον φυλλομετρητή, από την αρχή χρησιμοποιήθηκε και για τη συγγραφή κώδικα από την πλευρά του διακομιστή, από την ίδια τη Netscape στο προϊόν LiveWire, με μικρή επιτυχία. Η χρήση της Javascript στο διακομιστή εμφανίζεται πάλι σήμερα, με τη διάδοση του Node.js, ενός μοντέλου προγραμματισμού βασισμένο στα γεγονότα (events).

Δείγμα κώδικα Javascript

Ο κώδικας Javascript μιας σελίδας περικλείεται από τις ετικέτες της HTML <script type="text/javascript"> και </script>.

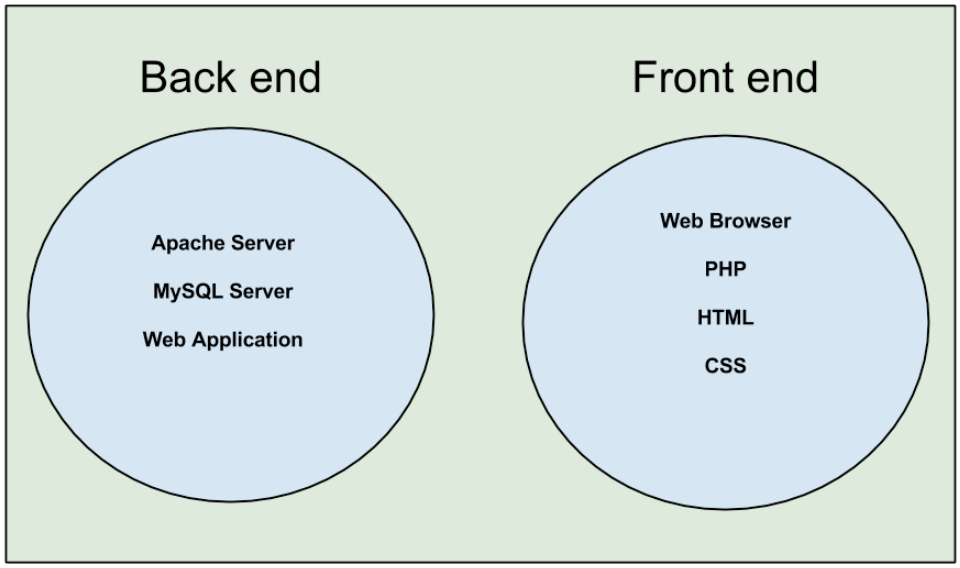
Για παράδειγμα, ο ακόλουθος κώδικας Javascript εμφανίζει ένα πλαίσιο διαλόγου με το κείμενο "Γεια σου, κόσμε!":

```
<script type="text/javascript">
alert('Γεια σου, κόσμε!');
</script>
```

Αν ο κώδικας Javascript περιέχει περισσότερες από μία εντολές, αυτές θα πρέπει να διαχωριστούν μεταξύ τους με το χαρακτήρα του ελληνικού ερωτηματικού ';' (δηλαδή της λατινικής άνω τελείας). Η χρήση του χαρακτήρα αυτού για την τελευταία

εντολή δεν είναι απαραίτητη. Η διαχώριση των εντολών στους νεότερους φυλλομετρητές (browsers) δεν είναι απαραίτητη.

Στο σημείο αυτό που έχει γίνει αναλυτική περιγραφή των εργαλείων και τεχνολογιών που απαρτίζουν την εφαρμογή, ας ξαναδούμε μια γενική άποψη σε γραφική μορφή.



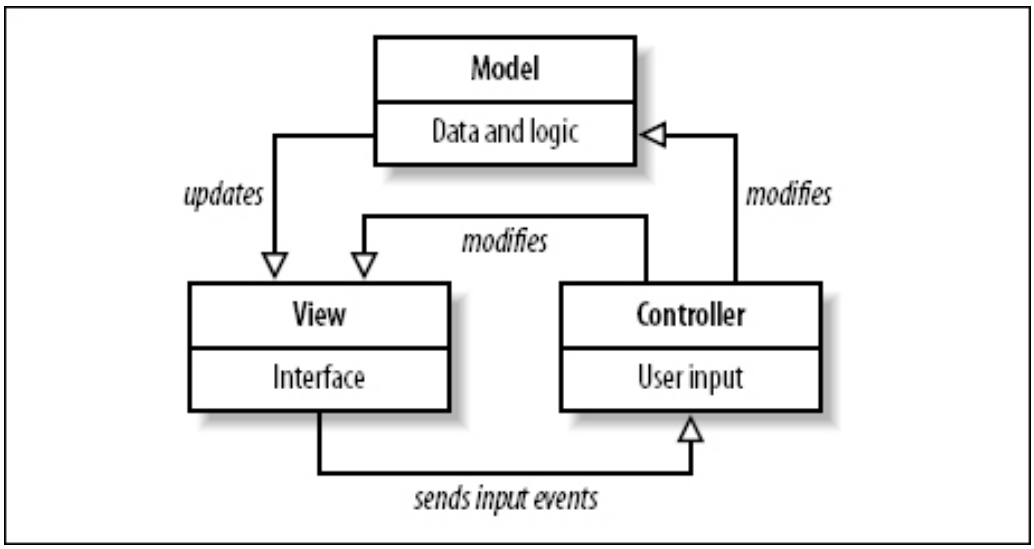
(εικόνα 3-2, γραφική απεικόνηση των χαρακτηριστικών της εφαρμογής που αναφέρθηκαν σε πιο ειδική μορφή)

3.3 **Αρχιτεκτονική Model-View-Controller** (**MVC**)

Γενικά

Το Model-View-Controller (**MVC**) είναι ένα αρχιτεκτονικό πρότυπο που χρησιμοποιείται στην τεχνολογία λογισμικού και συνήθως στη δημιουργία web εφαρμογών. Η επιτυχής χρήση του προτύπου απομονώνει τη λογική της web εφαρμογής από την διεπαφή χρήστη με το οποίο αλληλεπιδρά ο χρήστης, έχοντας έτσι ως αποτέλεσμα μία εφαρμογή πιο ευέλικτη, όπου είναι ευκολότερο να μεταβάλλει κάποιος είτε τον τρόπο παρουσίασης της εφαρμογής της είτε τον υποκείμενο κώδικα που εφαρμόζει λειτουργίες στα δεδομένα, χωρίς τα επιμέρους τμήματα να αλληλοεπηρεάζονται.

Στην MVC αρχιτεκτονική, το **model** αναπαριστά την πληροφορία, τα δεδομένα που η εφαρμογή χρησιμοποιεί. Το **view** αντιστοιχεί σε τρόπο παρουσίασης των δεδομένων του model στο interface του χρήστη π.χ.: text boxes, checkboxes κ.λπ., Τέλος ο **controller** διαχειρίζεται τη σύνδεση των δεδομένων του model με τη λογική του προγράμματος και καθορίζει ποια από τα δεδομένα του model θα παρουσιαστούν από το view στο interface του χρήστη.



εικόνα 3-3: Ένα απλό διάγραμμα που παρουσιάζει τη σχέση-σύνδεση ανάμεσα στα model view controller. Οι συνεχόμενες γραμμές αντικατοπτρίζουν άμεση σύνδεση ενώ οι διακεκομμένες έμμεση.

3.4 Περιγραφή του προτύπου

Το πρότυπο Model-view-controller είναι τόσο ένα αρχιτεκτονικό πρότυπο όσο και ένα σχεδιαστικό πρότυπο, αναλόγως που χρησιμοποιείται.

3.4.1 Σαν αρχιτεκτονικό πρότυπο

Είναι συνηθισμένο μία εφαρμογή να διασπάται σε χωριστά επίπεδα τα οποία τρέχουν σε διαφορετικούς υπολογιστές: Επίπεδο Παρουσίασης(UI), Επίπεδο Λογικής, και επίπεδο Πρόσβασης στα Δεδομένα. Στο MVC το επίπεδο παρουσίασης χωρίζεται περεταίρω σε view και controller.

Το πρότυπο MVC συχνά συναντάται σε web εφαρμογές, όπου το **view** είναι απλά μία html σελίδα, και ο **controller** είναι ο κώδικας που περισυλλέγει δυναμικά δεδομένα και δημιουργεί το περιεχόμενο μέσα στο HTML. Τέλος το **model** αντιπροσωπεύει το περιεχόμενο, που συνήθως βρίσκεται αποθηκεμένο σε κάποια βάση δεδομένων ή σε XML κόμβους(data) καθώς επίσης και τους κανόνες(metadata) που μετατρέπουν το περιεχόμενο αυτό ανάλογα με την αλληλεπίδραση του χρήστη.

Αν και για το πρότυπο MVC υπάρχει μια μεγάλη ποικιλία εκδόσεων, ένας **γενικός κύκλος λειτουργίας** του θα μπορούσε να περιγραφεί όπως παρακάτω:

1. Ο χρήστης αλληλεπιδρά με το User Interface με κάποιο τρόπο.
2. Ο controller χειρίζεται το event εισόδου από την διεπαφή χρήστη, συνήθως μέσω κάποιου event handler.
3. Ο controller ειδοποιεί το model για το event που έλαβε χώρα, συνήθως μεταβάλλοντας την κατάσταση του model με κάποιο τρόπο.
4. Το view χρησιμοποιεί το model(έμμεσα) για να δημιουργήσει το κατάλληλο νέο User Interface. Το view παίρνει επίσης δεδομένα από το model. Το model όμως δεν έχει άμεση γνώση του view.
5. Το user interface περιμένει για επόμενη ενέργεια του χρήστη, η οποία ξεκινά ένα νέο κύκλο.

Αποσυνδέοντας το model και το view, το πρότυπο MVC μειώνει κατά πολύ την πολυπλοκότητα της αρχιτεκτονικής μιας web εφαρμογής ενώ ταυτόχρονα αυξάνει την προσαρμοστικότητά και την ευελιξία της.

3.4.2 Σαν πρότυπο σχεδίασης

Το πρότυπο MVC περιλαμβάνει πολύ περισσότερα τμήματα της αρχιτεκτονικής μιας εφαρμογής από τα συνηθισμένα ενός σχεδιαστικού προτύπου.

Model:

Πρόκειται για μία **domain-specific** αναπαράσταση της πληροφορίας πάνω στην οποία ενεργεί η εφαρμογή. Η λογική της εφαρμογής (domain-logic) προσθέτει νόημα στα ανεπεξέργαστα δεδομένα.

Πολλές εφαρμογές χρησιμοποιούν έναν μόνιμο(σταθερό) αποθηκευτικό μηχανισμό(όπως π.χ.: μία βάση δεδομένων) για να αποθηκεύουν τα δεδομένα τους.

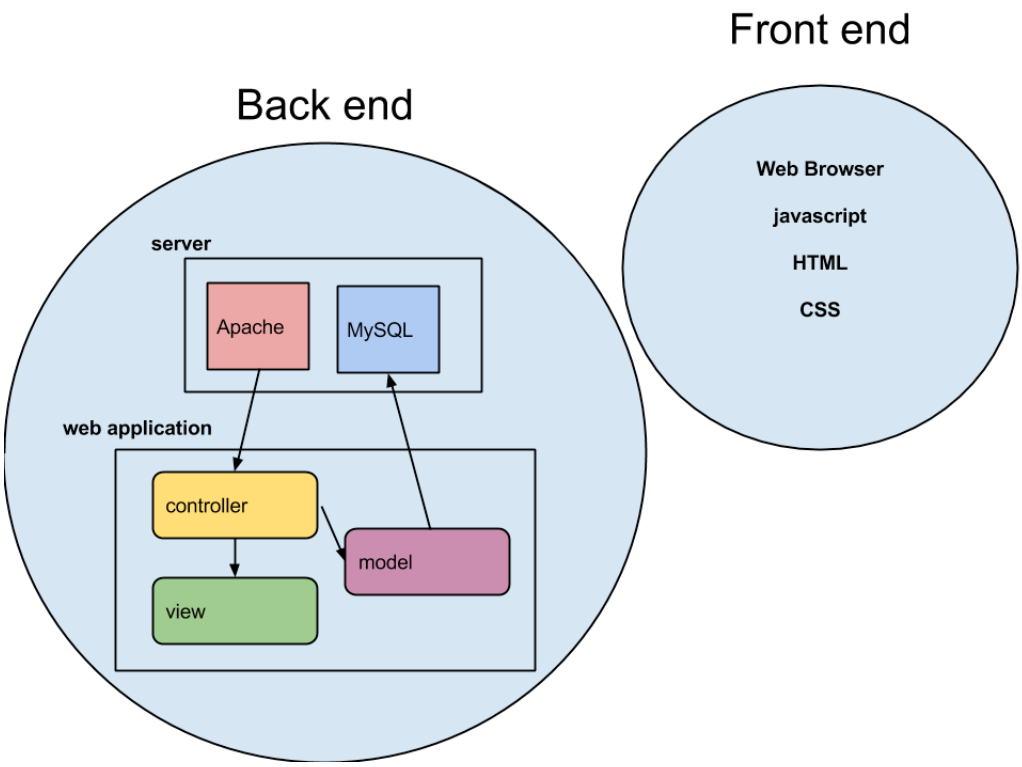
View:

Παρουσιάζει τα δεδομένα του model σε μία μορφή κατάλληλη για αλληλεπίδραση, συνήθως user interface στοιχεία. Είναι δυνατόν πολλαπλά διαφορετικά views να υπάρχουν για ένα μοναδικό model για διαφορετικούς σκοπούς.

Controller:

Επεξεργάζεται και ανταποκρίνεται σε events, που συνήθως προκαλούνται από επιλογές του χρήστη και είναι πιθανόν να επιδρά και να αλλάζει το model.

Μετά την ανάλυση της MVC αρχιτεκτονικής η γενικότερη εικόνα της εφαρμογής αλλάζει.



(εικόνα 3-3, γραφική απεικόνηση των χαρακτηριστικών της εφαρμογής που αναφέρθηκαν συμπεριλαμβανομένης της αρχιτεκτονικής MVC)

4. Στάδια ανάπτυξης της εφαρμογής

4.1 Σχεδίαση της βάσης δεδομένων

Όπως κάθε εφαρμογή, έτσι και η δικιά μας απαιτεί μια βάση δεδομένων. Ο σκοπός της βάσης δεδομένων είναι η φύλαξη δεδομένων και πληροφοριών που απαιτούνται για την λειτουργία της ιστοσελίδας.

Στη συγκεκριμένη περίπτωση έχει κατασκευαστεί μία (1) βάση δεδομένων με όνομα “p4117 6rome_med” όπου αποθηκεύονται πληροφορίες της εφαρμογής, ομαδοποιημένες σε 4 πίνακες (tables) ανάλογα με το είδος και τη χρήση τους.

Παρακάτω φαίνονται τα ονόματα των πινάκων (tables) της βάσης δεδομένων και η χρήση τους.

Product

Στον πίνακα αυτόν αποθηκέυονται όλες οι πληροφορίες που σχετίζονται για ένα προϊόν, που βρίσκεται στην αποθήκη ή ακόμα και αν έχει πωληθεί.

1	id	Int(11)	Είναι το κλειδί βάσει του οποίου ξεχωρίζει κάθε εγγραφή στον πίνακα
2	supplier	Varchar(100)	Το όνομα προμηθευτή του προϊόντος
3	invoicetype	Varchar(100)	Ο τύπος τιμολογίου
4	invoicenumber	Int(11)	Αριθμός τιμολογίου του προμηθευτή
5	invoiceday	Int(11)	Ημέρα καταχώρησης προϊόντος
6	invoicemonth	Int(11)	Μήνας καταχώρησης προϊόντος
7	invoiceyear	Int(11)	Έτος καταχώρησης προϊόντος
8	quantity	Int(11)	Ποσότητα
9	serial	Int(11)	Σειριακός αριθμός προϊόντος
10	type	Varchar(100)	Ο τύπος του προϊόντος
11	model	Varchar(100)	Το μοντέλο
12	category	Varchar(100)	Η γενικότερη κατηγορία προϊόντων στην οποία ανήκει.
13	description	Varchar(100)	Περιγραφή
14	buyprice	Int(11)	Τιμη αγοράς
15	sellprice	Int(11)	Τιμή πώλησης
16	status	Varchar(100)	Η κατάσταση στην οποία βρίσκεται.

			Δηλαδη νέο, επισκευασμένο, πουλημένο, κλπ.
17	comments	text	Κάποια σχόλια που μπορεί να γράψει ο χρήστης
18	lastuser	Varchar(100)	Αποθηκεύει αυτόματα τον τελευταίο χρήστη που έκανε οποιαδήποτε αλλαγή στο προϊόν
19	customerinvoice	Int(11)	Σε περίπτωση πώλησης, ενοικίασης, αποθηκεύει τον αριθμό παραστατικού
20	lastday	Int(11)	Τελευταία μέρα που έγινε οποιαδήποτε μεταβολή
21	lastmonth	Int(11)	Τελευταίος μήνας που έγινε οποιαδήποτε μεταβολή
22	lastyear	Int(11)	Τελευταίο έτος που έγινε οποιαδήποτε μεταβολή

Client

Στον πίνακα αυτόν αποθηκεύονται όλες οι πληροφορίες που σχετίζονται με τους πελάτες της επιχείρησης.

1	id	Int(11)	Είναι το κλειδί βάσει του οποίου ξεχωρίζει κάθε εγγραφή στον πίνακα
2	afm	Varchar(100)	Το ΑΦΜ του πελάτη

3	clienttype	Varchar(100)	Ο τύπος πελάτη, δηλαδή ιδιώτης ή εταιρία
4	name	Varchar(100)	Ονομασία
5	address	Varchar(100)	Διεύθυνση
6	phone	Varchar(100)	Τηλέφωνο
7	fax	Varchar(100)	Φαξ
8	comments	text	Σχόλια που μπορούν να γράψουν οι χρήστες

Supplier

Στον πίνακα αυτόν αποθηκέυονται όλες οι πληροφορίες που σχετίζονται με τους προμηθευατές της επιχείρησης.

1	id	Int(11)	Είναι το κλειδί βάσει του οποίου ξεχωρίζει κάθε εγγραφή στον πίνακα
2	afm	Varchar(100)	Το ΑΦΜ του Προμηθευτή
3	clienttype	Varchar(100)	Ο τύπος προμηθευτή, δηλαδή αναλώσιμα, ηλεκτρολογικά, κλπ.
4	name	Varchar(100)	Ονομασία
5	address	Varchar(100)	Διεύθυνση
6	phone	Varchar(100)	Τηλέφωνο
7	fax	Varchar(100)	Φαξ
8	comments	text	Σχόλια που μπορούν να γράψουν οι χρήστες

Clientinvoice

Στον πίνακα αυτόν αποθηκέυονται όλες οι πληροφορίες που σχετίζονται με τα παραστατικά που εκδίδει η επιχείρηση.

1	id	Int(11)	Είναι το κλειδί βάσει του οποίου ξεχωρίζει κάθε εγγραφή στον πίνακα
2	invoicetype	Varchar(100)	Τύπος παραστατικού, πχ Τιμολόγιο ή Δελτίο Αποστολής κλπ.
3	invoicereason	Varchar(100)	Σκοπός διακίνησης των προϊόντων στο παραστατικό
4	invoiceday	Int(11)	Μέρα έκδοσης
5	invoicemonth	Int(11)	Μήνας έκδοσης
6	invoiceyear	Int(11)	Έτος έκδοσης
7	client	Varchar(100)	Ο πελάτης στον οποίο εκδίδεται το παραστατικό
8	invoicenumber	Int(11)	Ένας μοναδικός αριθμός παραστατικού
9	paytype	Varchar(100)	Τρόπος πληρωμής, πχ μετρητά ή πίστωση
10	cost	Int(11)	Καθαρό κέρδος
11	fpa	Int(11)	ΦΠΑ
12	totalcost	Int(11)	Τελικό κόστος Παραστατικού

4.2 Μορφή της Εφαρμογής

Για να καλυφθεί η λειτουργικότητα της εφαρμογής, όπως αυτή περιγράφηκε παραπάνω σχεδιάστηκαν κατάλληλες σελίδες και τμήματα σελίδων, μεταξύ των οποίων πλοηγείται ο χρήστης αλληλεπιδρώντας με την εφαρμογή ώστε να επιτύχει το επιθυμητό αποτέλεσμα.

Η γενική ιδέα για τη διεπιφάνεια της εφαρμογής μας συνίσταται στη χρήση συστατικών που τοποθετούνται στις σελίδες της και της προσδίδουν την απαιτούμενη λειτουργικότητα.

Συστατικά όπως φόρμες, κουμπιά, μενού επιλογής και πίνακες, γνώριμα στους χρήστες του διαδικτύου, τα οποία δομούνται ακολουθώντας ένα νοητό περίγραμμα (template) ώστε να συνθέσουν μια διεπαφή φιλική στο χρήστη.

Οι φόρμες χρησιμοποιούνται όπου προβλέπεται – αναμένεται είσοδος από το χρήστη, τα κουμπιά για την αλληλεπίδραση του με την εφαρμογή μιας και επιτρέπουν την πυροδότηση γεγονότων, οι πίνακες για παρουσίαση δεδομένων, ενώ τα μενού επιλογής σε σημεία όπου η είσοδος που αναμένεται από το χρήστη είναι τυποποιημένη, οπότε του δίνεται η δυνατότητα να επιλέξει παρά να πληκτρολογήσει.

Στην αρχική σελίδα, λοιπόν, παρουσιάζεται φόρμα εισόδου στην υπηρεσία, όπου οι χρήστες θα συνδεθούν στην εφαρμογή εισάγωντας το όνομα χρήστη και τον μυστικό κωδικό πρόσβασης και εφόσον οι κωδικοί αυτοί επαληθευτούν (τοπικά ή μη), αναγνωρίζεται ο χρήστης και μεταφερόμαστε στη σελίδα του χρήστη, το λεγόμενο dashboard.

4.3 Παρουσίαση της εφαρμογής

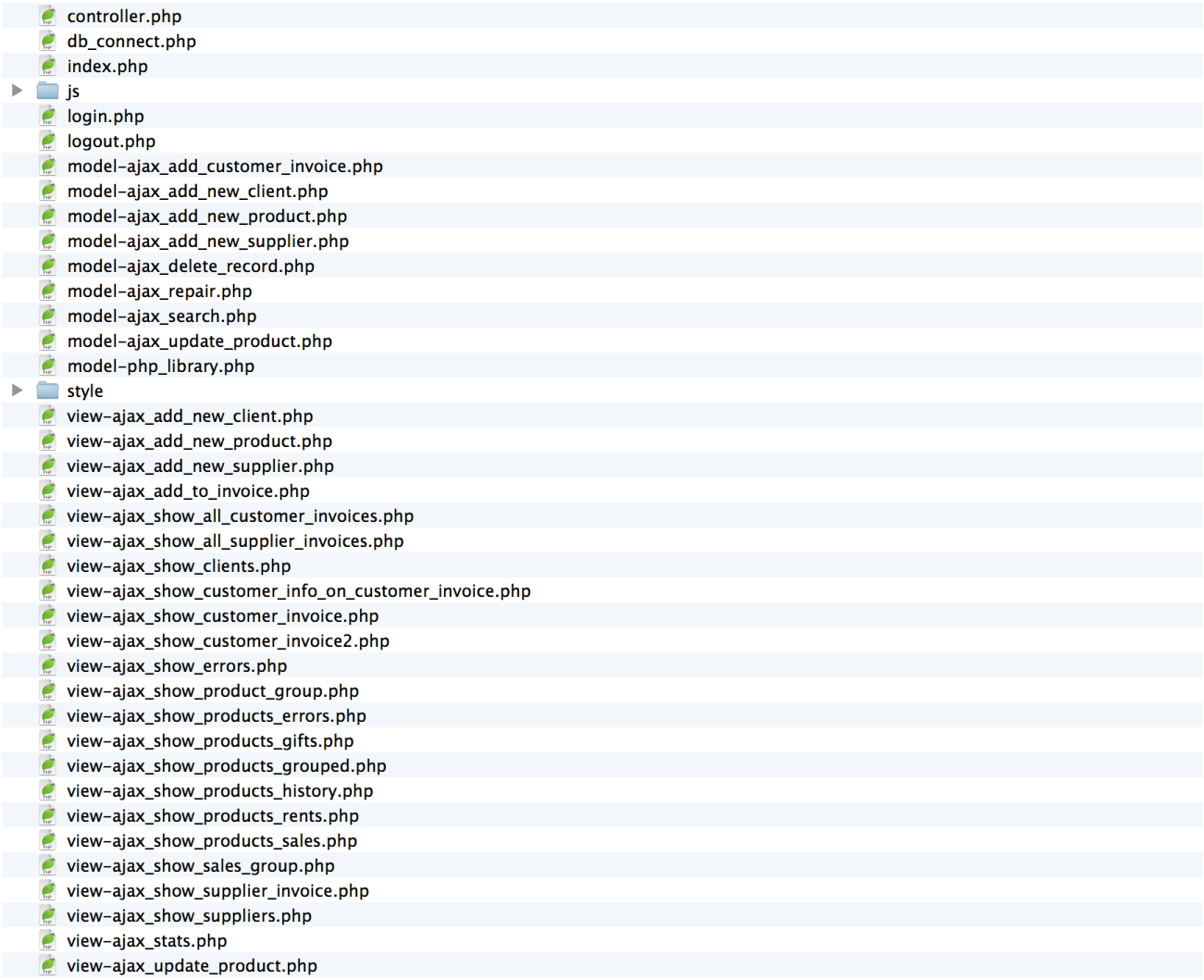
Στο σημείο αυτό, περνάμε στην παρουσίαση και τη διαμόρφωση της web εφαρμογής, συμπληρώνοντας έτσι το πάζλ της ανάπτυξης του και γεμίζοντας τα όποια κενά μπορεί να υπάρχουν στην κατανόηση της. Η παρουσίαση υλοποιείται με αρχεία που βρίσκονται στον κεντρικό κατάλογο της εφαρμογής που αποτελούν τις σελίδες της εφαρμογής.

4.3.1 Παρουσίαση λογικής κώδικα

Όπως σε κάθε MVC αρχιτεκτονική όλες οι client request χειρίζονται από ένα αρχείο (controller.php) το οποίο τις κατευθύνει

στον κατάλληλο προορισμό εντός της εφαρμογής. Μόλις ο χρήστης συνδεθεί στην εφαρμογή από το αρχείο index.php, κατευθύνεται στον conteller.php και θα παραμείνει σε αυτό έως ότου αποσυνδεθεί.

Με μια πρώτη ματιά στον κεντρικό φάκελο της εφαρμογής, βλέπουμε τα παρακάτω αρχεία:

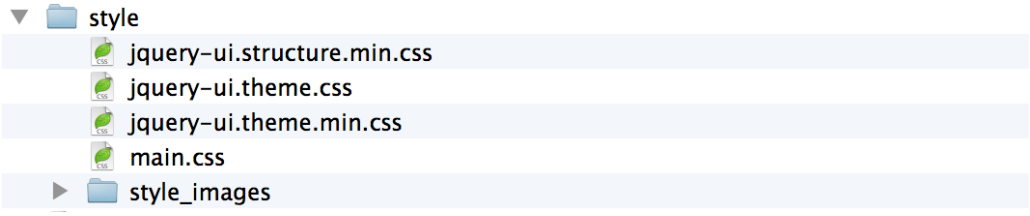


Με λίγη παρατήρηση είναι φανέρο πως υπάρχει ξεκάθαρη διαφοροποίηση μεταξύ των αρχείων. Όπως μάθαμε και στην MVC αρχιτεκτονική η εφαρμογή χωρίζεται σε τρί βασικά τμήματα. Το Model, το View και το Controller.

Όπως παρατηρούμε υπάρχει ένα αρχείο index.php είναι το πρώτο περιβάλλον που βλέπει ο χρήστης πριν συνδεθεί στην εφαρμογή. Αμέσως μετά την σύνδεση προωθείται στο controller.php που αναλαμβάνει να κάνει όλα τα αιτήματα προς τα Model και View.

Στον κεντρικό κατάλογο υπάρχουν μόνο δύο βασική φάκελοι, ο φάκελος " /style" και ο ο φάκελος " /js" , τα περιεχόμενα των οποίων παρουσιάζονται παρακάτω:

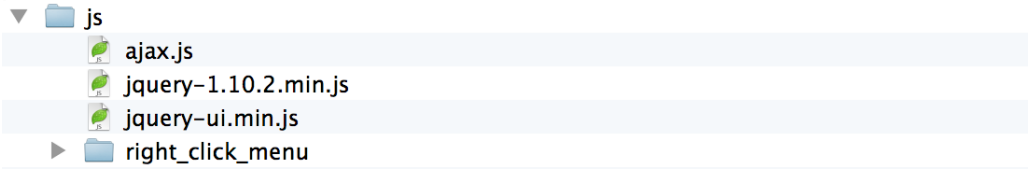
/style



Οτιδήποτε στην εφαρμογή αφορά την μορφοποίηση των δεδομένων και τον τρόπο παρουσίασής τους, βρίσκεται σε αυτόν τον φάκελο. Το αρχείο main.css αναλαμβάνει όλη την μορφοποίηση της εφαρμογής και αποφασίζει τον τρόπο με τον οποίο θα παρουσιάζονται τα δεδομένα.

Τα υπόλοιπα αρχεία μορφοποιούν ελάχιστη πληροφορία που έχει να κάνει με την λειτουργία javascript τμήμάτων κώδικα για απλή επεξεργασία της πληροφορίας στον περιηγητή του χρήστη.

/js



Στον φάκελο αυτό φορτώνονται έτοιμες βιβλιοθήκες της javascript. Οι βιβλιοθήκες αυτές απλοποιούν σε μεγάλο βαθμό απλές λειτουργίες που απαιτούνται μετά την λήψη των δεδομένων από τον σερβερ της εφαρμογής.

Το αρχείο "ajax.js" είναι βιβλιοθήκη που δημιουργήθηκε για τις ανάγκες της συγκεκριμένης εφαρμογής. Ο τίτλος ajax μαρτυρά γεγονός ότι όλες οι λειτουργίες εκτελούνται ασύγχρονα με χρήση της τεχνικής ajax που προσφέρει η βιβλιοθήκη jquery της javascript.

4.3.1.1 Τεχνική Ajax

Γενικά μια ιστοσελίδα είναι ένα σύνολο html οδηγιών που στέλνει ο server στον browser του επισκέπτη όταν αυτός αιτείται να "δει" το περιεχόμενο της ιστοσελίδας αυτής.

Η αποστολή των html οδηγιών γίνεται μια φορά και αφού αυτή ολοκληρωθεί η επικοινωνία server-browser κλείνει. Έτσι κάθε φορά που ο κάτοχος-δημιουργός-διαχειριστής της ιστοσελίδας κάνει αλλαγές στο περιεχόμενο της, οι αλλαγές αυτές δεν είναι άμεσα ορατές από τον επισκέπτη εκτός και αν αυτός ανανεώσει την ιστοσελίδα πατώντας το F5, οπότε και ξανα ανοίγει η επικοινωνία του browser με τον server παραλαμβάνοντας εκ νέου το ανανεωμένο περιεχόμενο.

Η χρήση της τεχνικής AJAX επιτρέπει σε μια ιστοσελίδα να ανανεώνεται ασύγχρονα (asynchronously) ανταλλάσοντας στο παρασκήνιο μικρού όγκου δεδομένα με τον server επιτρέποντας να ανανεώνονται μέρη της ιστοσελίδας (ένα div για παράδειγμα), χωρίς να ανανεώνεται ολόκληρη η σελίδα!

Η τεχνική AJAX χρησιμοποιείται πολύ συχνά σε σελίδες που περιέχουν φόρμες εγγραφής σε κάποια ιστοσελίδα ή forum. Είναι γνωστό ότι στις φόρμες αυτές πρέπει να επιλέξουμε ένα επιθυμητό username. Αν συμπληρώσατε ποτέ μια τέτοια φόρμα και είδατε να εμφανίζεται ένα μήνυμα προειδοποίησης ότι το username που πληκτρολογήσατε υπάρχει ήδη, χωρίς να ξαναφορτωθεί η σελίδα, τότε αυτή η σελίδα χρησιμοποιεί την τεχνική AJAX.

Επίσης το Facebook χρησιμοποιεί ευρέως την τεχνική αυτή σε διάφορα τμήματα της σελίδας, όπως για να εμφανίζει τα νέα posts των φίλων σας που σκρολάρουν συνεχώς στην λίστα επάνω δεξιά ή για να εμφανίσει τα σχόλια κάτω από τα posts σας.

Γενικά μια ιστοσελίδα είναι ένα σύνολο html οδηγιών που στέλνει ο server στον browser του επισκέπτη όταν αυτός αιτείται να "δει" το περιεχόμενο της ιστοσελίδας αυτής. Η αποστολή των html οδηγιών γίνεται μια φορά και αφού αυτή ολοκληρωθεί η επικοινωνία server-browser κλείνει. Έτσι κάθε φορά που ο κάτοχος-δημιουργός-διαχειριστής της ιστοσελίδας κάνει αλλαγές στο περιεχόμενο της, οι αλλαγές αυτές δεν είναι άμεσα ορατές από τον επισκέπτη εκτός και αν αυτός ανανεώσει την ιστοσελίδα πατώντας το F5, οπότε και ξανα ανοίγει η επικοινωνία του browser με τον server παραλαμβάνοντας εκ νέου το ανανεωμένο περιεχόμενο.

Η χρήση της τεχνικής AJAX επιτρέπει σε μια ιστοσελίδα να ανανεώνεται ασύγχρονα (asynchronously) ανταλλάσοντας στο παρασκήνιο μικρού όγκου δεδομένα με τον server επιτρέποντας να ανανεώνονται μέρη της ιστοσελίδας (ένα div για παράδειγμα), χωρίς να ανανεώνεται ολόκληρη η σελίδα!

Η τεχνική AJAX χρησιμοποιείται πολύ συχνά σε σελίδες που περιέχουν φόρμες εγγραφής σε κάποια ιστοσελίδα ή forum. Είναι γνωστό ότι στις φόρμες αυτές πρέπει να επιλέξουμε ένα επιθυμητό username. Αν συμπληρώσατε ποτέ μια τέτοια φόρμα και είδατε να εμφανίζεται ένα μήνυμα προειδοποίησης ότι το username που πληκτρολογήσατε υπάρχει ήδη, χωρίς να ξαναφορτωθεί η σελίδα, τότε αυτή η σελίδα χρησιμοποιεί την τεχνική AJAX.

Επίσης το Facebook χρησιμοποιεί ευρέως την τεχνική αυτή σε διάφορα τμήματα της σελίδας, όπως για να εμφανίζει τα νέα posts των φίλων σας που σκρολάρουν συνεχώς στην λίστα επάνω δεξιά ή για να εμφανίσει τα σχόλια κάτω από τα posts σας.

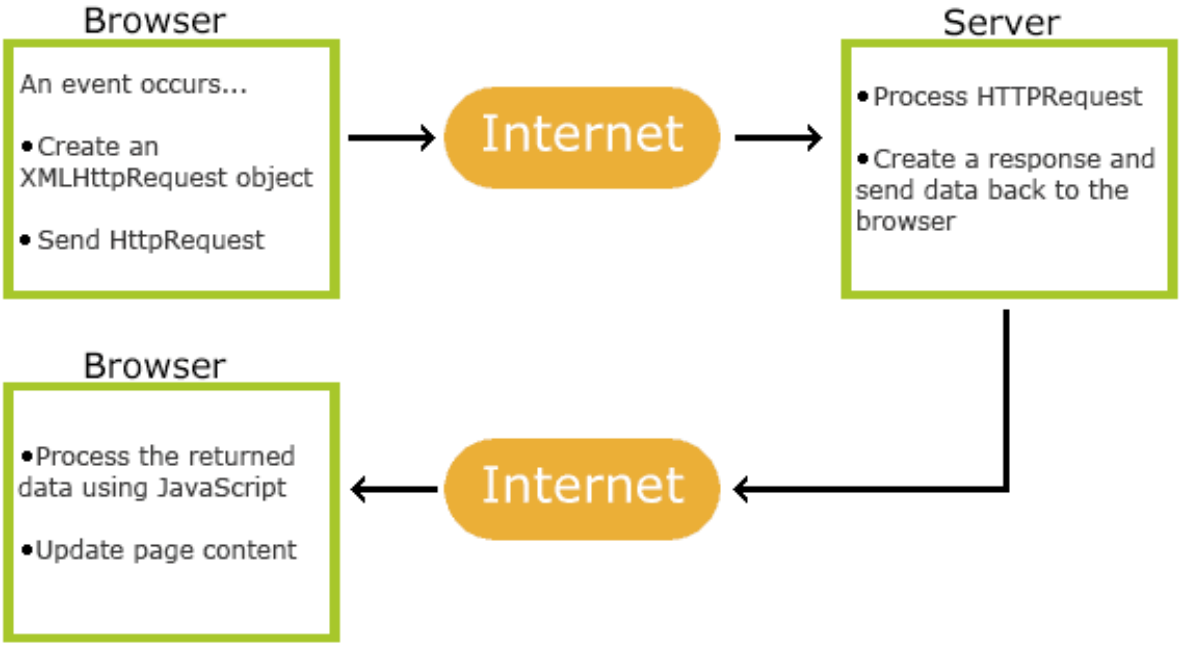
Αυτό που πρέπει να τονίσουμε ώστε να γίνει κατανοητή η χρησιμότητα της τεχνικής AJAX είναι ότι αν ο διαχειριστής της σελίδας προσθέσει για παράδειγμα το όνομα μιας πόλης στην λίστα των διαθέσιμων προορισμών, τότε αυτή η πόλη (αυτή η νέα καταχώρηση) **την ίδια στιγμή γίνεται άμεσα διαθέσιμη στον επισκέπτη** και θα την εμφανίσει στον browser του χωρίς να χρειαστεί να ανανεώσει την σελίδα

Τι ακριβώς όμως είναι η AJAX

Η AJAX δεν είναι γλώσσα προγραμματισμού όπως η JavaScript, αλλά ούτε και γλώσσα χαρακτηρισμού κειμένου όπως η HTML. Δεν θεωρείται καν γλώσσα.

AJAX σημαίνει Asynchronous Javascript And XML και είναι η τεχνική με την οποία μπορούμε να δημιουργήσουμε πολύ πιο γρήγορες και δυναμικές ιστοσελίδες περιορίζοντας τον όγκο δεδομένων που ανταλλάσσει ο server με τον browser του επισκέπτη. Η ιδιαίτερη τεχνική αυτή επιτρέπει την ανανέωση περιεχομένων μιας ιστοσελίδας χωρίς αυτή να ανανεωθεί ολόκληρη.

Η υλοποίηση της τεχνικής AJAX γίνεται με τον συνδυασμό του αντικειμένου (object) XMLHttpRequest (πραγματοποιεί την ασύγχρονη επικοινωνία με τον server), την Javascript/DOM (αλληλεπιδρά με τα δεδομένα και τα εμφανίζει), την CSS (μορφοποιεί τα προς εμφάνιση δεδομένα) και την XML (συχνά χρησιμοποιείται για την μεταφορά δεδομένων).



(εικόνα 4-1, γραφική αναπαράσταση της τεχνικής ajax για λήψη μόνο συγκεκριμένων τμημάτων μιας σελίδας)

Ο κώδικας στο αρχείο ajax.js :

```
function sendform(){//για κατι που δεν χρειάζεται να δω απλα τα αποθηκευω
        //callback handler for form submit
        $("#ajaxform").submit(function(e)
        {
           var postData = $(this).serializeArray();
           var formURL = $(this).attr("action");
           $.ajax(
           {
              url : formURL,
              type: "POST",
              data : postData,
              success:function(data, textStatus, jqXHR)
              {
                 //data: return data from server
                 result = data;
                 if(result == true){
                     $("#add_button").replaceWith($('<p
style="color:green;margin:0px;">Προστέθηκε!</p>'));
                 }
                 if(result == 5){
                     location.href="controller.php";
                 }
              },
              error: function(jqXHR, textStatus, errorThrown)
```

```
                {
                    //if fails
                }

            });
            //e.preventDefault(); //STOP default action
            e.unbind(); //unbind. to stop multiple form submit.
        });

        $("#ajaxform").submit(); //Submit  the FORM
}

function sendform2(){//στέλνω και λαμβάνω δεδομένα
        //callback handler for form submit
        $("#ajaxform").submit(function(e)
        {
            var postData = $(this).serializeArray();
            var formURL = $(this).attr("action");
            $.ajax(
            {
                url : formURL,
                type: "POST",
                data : postData,
                success:function(data, textStatus, jqXHR)
                {
                    //data: return data from server
                    result = data;
                    $('#invoice').html(result);
                },
                error: function(jqXHR, textStatus, errorThrown)
                {
                    //if fails
                }

            });
            //e.preventDefault(); //STOP default action
            e.unbind(); //unbind. to stop multiple form submit.
        });

        $("#ajaxform").submit(); //Submit  the FORM
}
function sendform3(){//για την αναζήτηση
        //callback handler for form submit
        $("#ajaxform2").submit(function(e)
        {
            var postData = $(this).serializeArray();
            var formURL = $(this).attr("action");
            $.ajax(
            {
                url : formURL,
                type: "POST",
                data : postData,
                success:function(data, textStatus, jqXHR)
                {
                    //data: return data from server
                    result = data;
                    $('#main').html(result);
                },
                error: function(jqXHR, textStatus, errorThrown)
                {
                    //if fails
                }

            });
            //e.preventDefault(); //STOP default action
            e.unbind(); //unbind. to stop multiple form submit.
```

```
		});

		$("#ajaxform2").submit(); //Submit  the FORM
}
function reload(){
		 location.reload();
}
function load_add_new_supplier(){
		$(document).ready(function(){
			$('#main').load("view-ajax_add_new_supplier.php").hide().fadeIn('slow');
		});
}
function load_add_new_client(){
		$(document).ready(function(){
			$('#main').load('view-ajax_add_new_client.php').hide().fadeIn('slow');
		});
}
function load_add_new_product(){
		$(document).ready(function(){
			$('#main').load('view-ajax_add_new_product.php').hide().fadeIn('slow');
		});
}
function load_show_all_products(ordering){
		$(document).ready(function(){
			$('#main').load('view-
ajax_show_products_history.php',{ordering:ordering}).hide().fadeIn('slow');
		});
}
function errors(ordering){
		$(document).ready(function(){
			$('#main').load('view-
ajax_show_errors.php',{ordering:ordering}).hide().fadeIn('slow');
		});
}
function load_show_all_gifts(ordering){//grouped
		$(document).ready(function(){
			$('#main').load('view-
ajax_show_products_gifts.php',{ordering:ordering}).hide().fadeIn('slow');
		});
}
function load_show_all_rents(ordering){//grouped
		$(document).ready(function(){
			$('#main').load('view-
ajax_show_products_rents.php',{ordering:ordering}).hide().fadeIn('slow');
		});
}
function load_show_all_sales(ordering){//grouped
		$(document).ready(function(){
			$('#main').load('view-
ajax_show_products_sales.php',{ordering:ordering}).hide().fadeIn('slow');
		});
}
function load_show_all_errors(ordering){//grouped
		$(document).ready(function(){
			$('#main').load('view-
ajax_show_products_errors.php',{ordering:ordering}).hide().fadeIn('slow');
		});
}
function load_show_all_products_grouped(ordering){
		$(document).ready(function(){
			$('#main').load('view-
ajax_show_products_grouped.php',{ordering:ordering}).hide().fadeIn('slow');
		});
}
function load_show_all_product_group(id){//δινω id ενος για να βρει το group
		$(document).ready(function(){
```

```
		$('#main').load('view-ajax_show_product_group.php',{id:id}).hide().fadeIn('slow');
	});
}
function load_show_sales_group(id){//δινω id ενος για να βρει το group
	$(document).ready(function(){
		$('#main').load('view-ajax_show_sales_group.php',{id:id}).hide().fadeIn('slow');
	});
}
function load_show_all_suppliers(ordering){
	$(document).ready(function(){
		$('#main').load('view-
ajax_show_suppliers.php',{ordering:ordering}).hide().fadeIn('slow');
	});
}
function load_show_all_clients(ordering){
	$(document).ready(function(){
		$('#main').load('view-
ajax_show_clients.php',{ordering:ordering}).hide().fadeIn('slow');
	});
}
function delete_record(table,id){
	$(document).ready(function(){
		$('#main').load('model-ajax_delete_record.php',{table:table,
id:id}).hide().fadeIn('slow');
	});
}
function repair(id){
	$(document).ready(function(){
		$('#main').load('repair.php',{id:id}).hide().fadeIn('slow');
	});
}
function load_show_all_suppliers_invoices(ordering){
	$(document).ready(function(){
		$('#main').load('view-
ajax_show_all_supplier_invoices.php',{ordering:ordering}).hide().fadeIn('slow');
	});
}
function load_show_all_customers_invoices(ordering){
	$(document).ready(function(){
		$('#main').load('view-
ajax_show_all_customer_invoices.php',{ordering:ordering}).hide().fadeIn('slow');
	});
}
function load_show_supplier_invoice(invoicenumber){
	$(document).ready(function(){
		$('#main').load('view-
ajax_show_supplier_invoice.php',{invoicenumber:invoicenumber}).hide().fadeIn('slow');

	});
}
function load_show_customer_invoice(){
	$(document).ready(function(){
		$('#container').load('view-
ajax_show_customer_invoice.php').hide().fadeIn('slow');
	});
}
function getval(sel) {//μόλις επιλέξω πελάτη στο παραστατικό βγαίνουν τα στοιχεία του
	customer=sel.value
	$(document).ready(function(){
	    $('#customer_info_on_customer_invoice').load('view-
ajax_show_customer_info_on_customer_invoice.php',{customer:customer}).hide().fadeIn('slow');
	});
}

function load_add_to_invoice(product_id){//βάζει προϊόντα στο καλάθι στο $_SESSION[]
	$(document).ready(function(){
```

```
                $('#supplier_invoice_products').load('view-
ajax_add_to_invoice.php',{product_id:product_id}).hide().fadeIn('slow');

        });
}
function load_customer_invoice_total(){
        $(document).ready(function(){
                $('#container').load("view-
ajax_show_customer_invoice2.php").hide().fadeIn('slow');
        });
}
function load_update_product(product_id){
        $(document).ready(function(){
                $('#main').load('view-
ajax_update_product.php',{product_id:product_id}).hide().fadeIn('slow');
        });
}
function stats(){
        $(document).ready(function(){
                $('#main').load('view-ajax_stats.php').hide().fadeIn('slow');
        });
}
$(function(){/*----------------------------------------------------------------Right click menu για products ;)*/
                $.contextMenu({
                  selector: '.table_row_product', //Προσοχή, αντί για παράμετρο, αλλάζω
selector.
                  callback: function(key, options) {
                  trid = $(this).closest('tr').attr('id');
                  table = "product";
                    if (key=="delete"){
                        delete_record(table,trid);
                    }
                    if (key=="add"){
                        load_add_to_invoice(trid);
                    }
                    if (key=="edit"){
                        load_update_product(trid);
                    }
                    if (key=="repair"){
                        repair(trid);
                    }
                  },
                  items: {
                    "edit": {name: "Edit", icon: "edit"},
                    "add": {name: "add", icon: "add"},
                    "repair": {name: "repair", icon: "repair"},
                    "delete": {name: "Delete", icon: "delete"},
                    "sep1": "---------",
                    "quit": {name: "Quit", icon: "quit"}
                  }
                });
});

$(function(){/*----------------------------------------------------------------Right click menu για
grouped_products ;)*/
                $.contextMenu({
                  selector: '.table_row_grouped_product', //Προσοχή, αντί για παράμετρο,
αλλάζω selector.
                  callback: function(key, options) {
                  trid = $(this).closest('tr').attr('id');
                  table = "product";
                    if (key=="delete"){
                        delete_record(table,trid);
                    }
                    if (key=="look"){
                                load_show_all_product_group(trid);
```

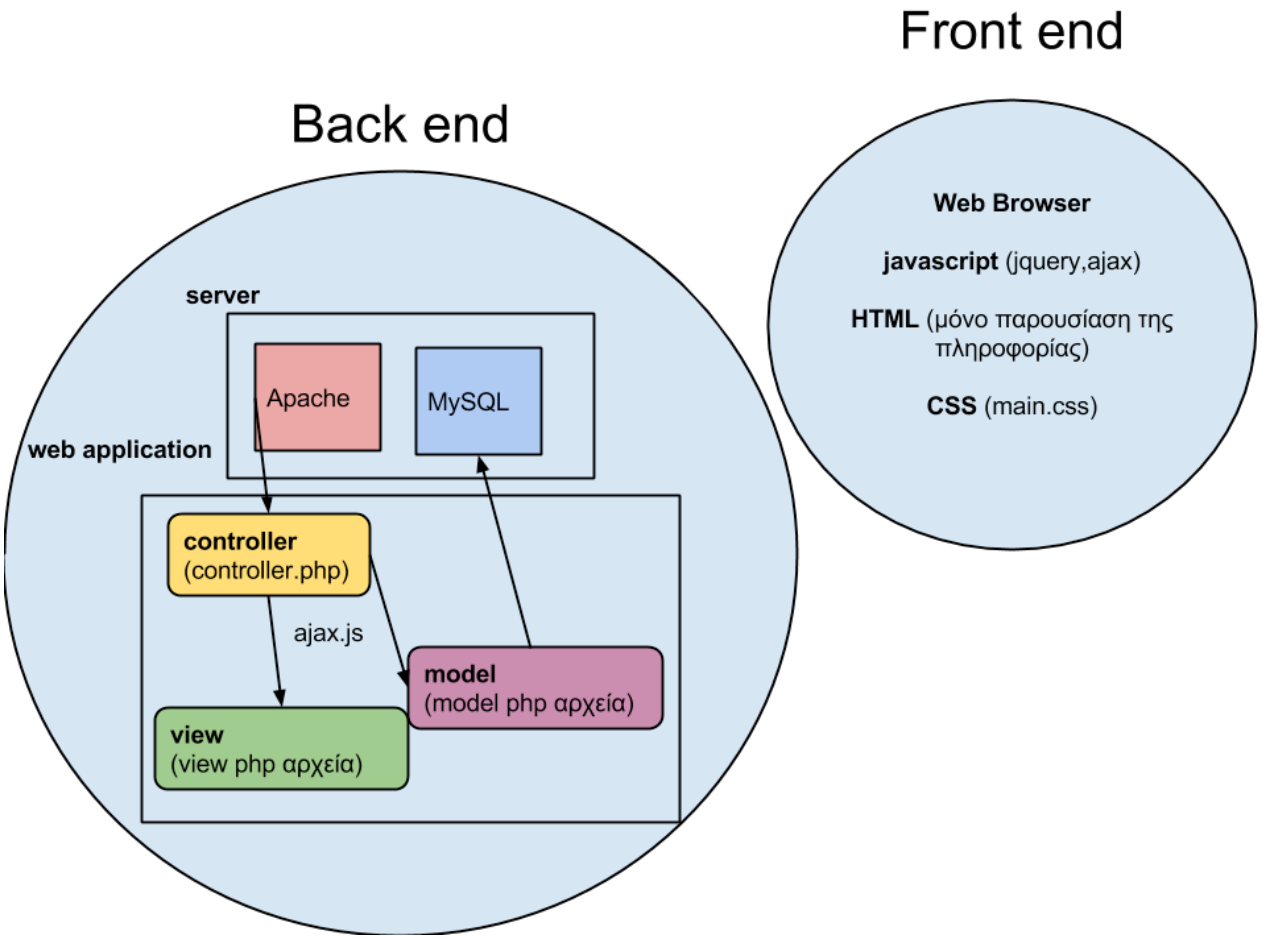
```
                    }
                },
                items: {
                  // "edit": {name: "Edit", icon: "edit"},
                   "look": {name: "Look", icon: "look"},
                  // "add": {name: "add", icon: "add"},
                   "delete": {name: "Delete", icon: "delete"},
                   "sep1": "---------",
                   "quit": {name: "Quit", icon: "quit"}
                }
            });
});

$(function(){/*-----------------------------------------------------------------Right click menu για grouped sales ;)*/
            $.contextMenu({
                selector: '.table_row_grouped_sales',
                callback: function(key, options) {
                trid = $(this).closest('tr').attr('id');
                table = "client";
                    if (key=="delete"){
                        delete_record(table,trid);
                    }
                    if (key=="look"){
                        //alert(trid);
                                load_show_sales_group(trid);
                    }
                },
                items: {
                    "edit": {name: "Edit", icon: "edit"},
                    "look": {name: "Look", icon: "look"},
                    "delete": {name: "Delete", icon: "delete"},
                    "sep1": "---------",
                    "quit": {name: "Quit", icon: "quit"}
                }
            });
});

$(function(){/*-----------------------------------------------------------------Right click menu για suppliers ;)*/
            $.contextMenu({
                selector: '.table_row_supplier',
                callback: function(key, options) {
                trid = $(this).closest('tr').attr('id');
                table = "supplier";
                    if (key=="delete"){
                        delete_record(table,trid);
                    }
                },
                items: {
                    "edit": {name: "Edit", icon: "edit"},
                    "delete": {name: "Delete", icon: "delete"},
                    "sep1": "---------",
                    "quit": {name: "Quit", icon: "quit"}
                }
            });
});

$(function(){/*-----------------------------------------------------------------Right click menu για clients ;)*/
            $.contextMenu({
                selector: '.table_row_client',
                callback: function(key, options) {
                trid = $(this).closest('tr').attr('id');
                table = "client";
                    if (key=="delete"){
                        delete_record(table,trid);
                    }
                },
```

```
                        items: {
                            "edit": {name: "Edit", icon: "edit"},
                            "delete": {name: "Delete", icon: "delete"},
                            "sep1": "---------",
                            "quit": {name: "Quit", icon: "quit"}
                        }
                    });
});

$(function(){/*-----------------------------------------------------------Right click menu για supplier invoice
;)*/
                    $.contextMenu({
                        selector: '.table_row_supplier_invoices',
                        callback: function(key, options) {
                        trid = $(this).closest('tr').attr('id');
                        table = "client";
                            if (key=="delete"){
                                delete_record(table,trid);
                            }
                            if (key=="look"){
                                //alert(trid);
                                        load_show_supplier_invoice(trid);
                            }
                        },
                        items: {
                            "edit": {name: "Edit", icon: "edit"},
                            "look": {name: "Look", icon: "look"},
                            "delete": {name: "Delete", icon: "delete"},
                            "sep1": "---------",
                            "quit": {name: "Quit", icon: "quit"}
                        }
                    });
});
```

Είναι άμεσα αντιληπτό πως από μόνες τους οι συναρτήσεις από τις οποίες απαρτίζεται δεν επιτελούν κάποια λειτουργία, αλλά «δρομολογούν» το αίτημα του χρήστη με ασύγχρονο τρόπο μέσω της μεθόδου .load() που προσφέρει η βιβλιοθήκη jquery στα κατάλληλα αρχεία php, συμβάλλοντας στην υλοποίηση της αρχιτεκτονικής MVC.

Δηλαδή όπως είναι κατανοήτό το αρχείο ajax.js περιλαμβάνει συναρτήσεις οι οποίες συνδέουν τον Controller με τα View και Model.

Σε αυτό το σημείο αξίζει να αναθεωρήσουμε την γενική εικόνα μας για την εφαρμογή.

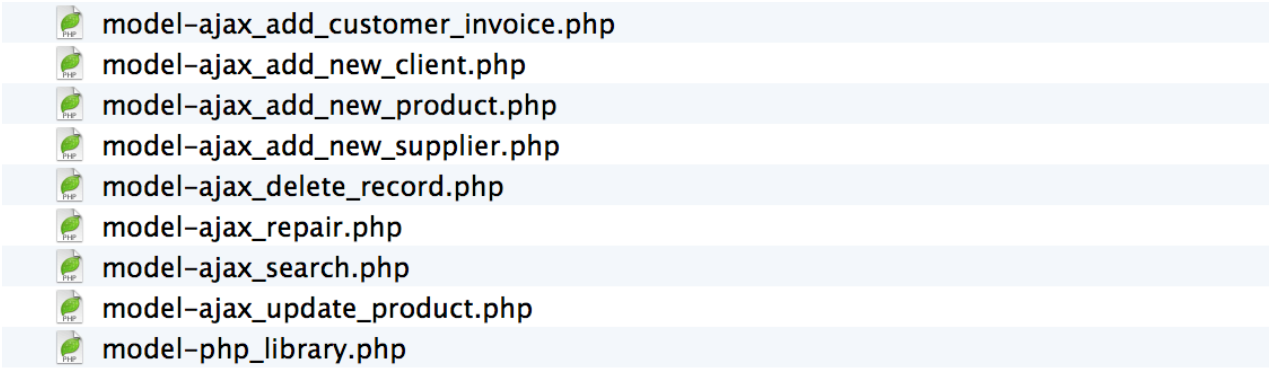


(εικόνα 4-2, Βελτιωμένη εικόνα της εφαρμογής μετά την παρουσίαση των παραπάνω πληροφοριών)

4.3.2 **Η δομή της εφαρμογής όπως προκύπτει βάσει του MVC**

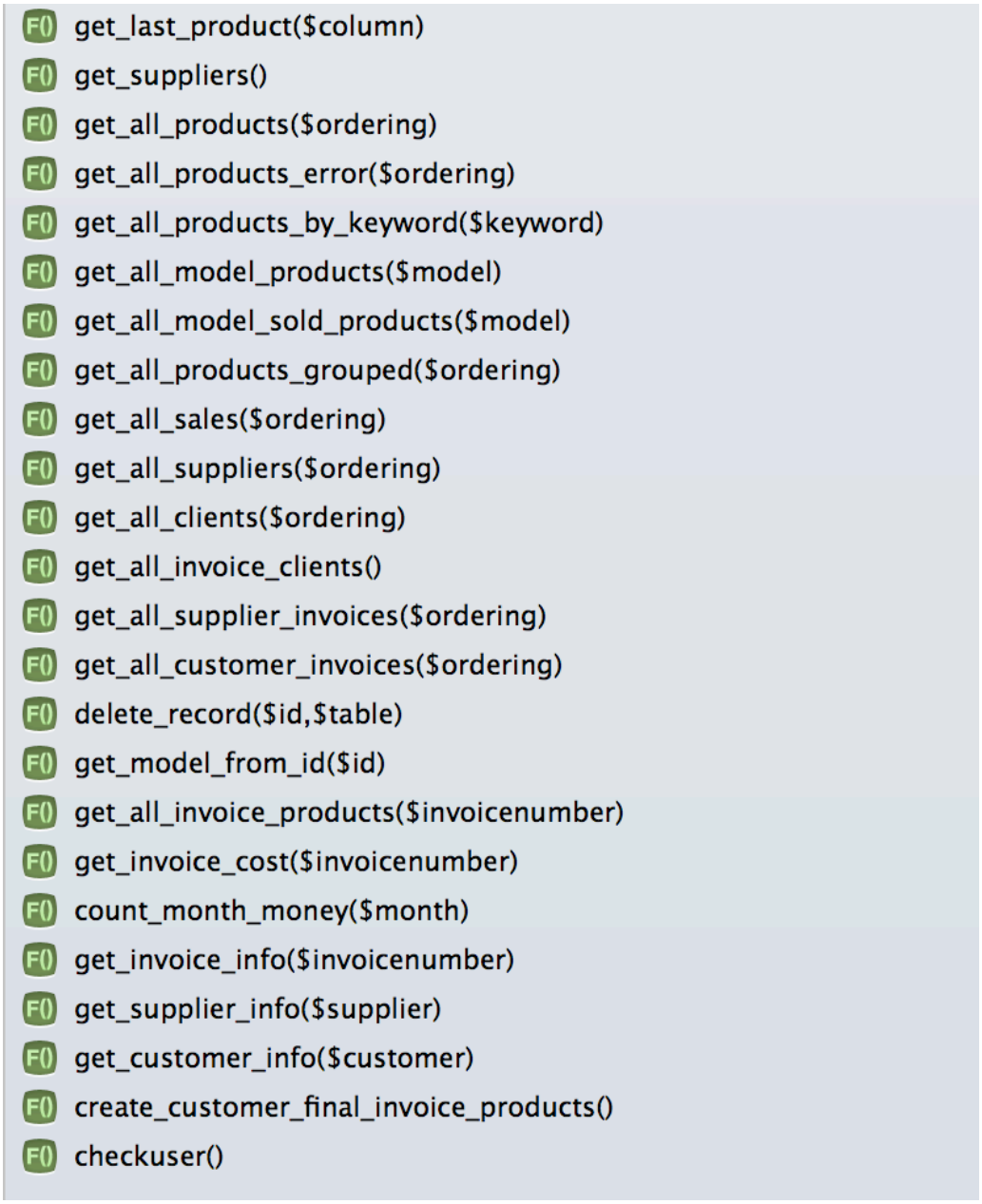
4.3.2.1 **MODEL**

Παρακάτω βλέπουμε αρχεία που στο σύνολό τους υλοποιούν το κομμάτι Model της αρχιτεκτονικής MVC. Ρόλος τους είναι να λαμβάνουν καθώς και να τροποποιούν πληροφορίες στην βάση δεδομένων. Οτιδήποτε έχει να κάνει με εισαγωγή, αλλαγή, διαγραφή από την βάση, έχει να κάνει με τα models.



Ένα αρχείο ιδιαίτερο από τα παραπάνω είναι το “model-php_library.php”. Είναι μια βιβλιοθήκη με συναρτήσεις που περιλαμβάνουν σύνηθη αιτήμα προς την βάση δεδομένων αλλά και λειτουργίες που διευκολύνουν την εκτέλεση των παραπάνω αρχείων.

Ας δούμε τις συναρτήσεις αυτές αναφορικά σε πρώτη φάση:



Το περιεχόμενο του αρχείου “model-php_library.php” είναι το παρακάτω:

```
<?php
require_once('db_connect.php');

function get_last_product($column){ //παίρνω τελευταίο προϊόν από τη βάση για να εμφανίζεται όταν
προσθέτω προϊόν
	global $dbh;// το παίρνω από 'db_connect.php'
	$sql = 'SELECT * FROM product ORDER BY ID DESC LIMIT 1';
	$sth = $dbh->prepare($sql);
	$sth->execute();
	foreach ($sth as $row){
		if($column == 2){
			echo $row['supplier'];
		}
		elseif($column == 3){
			echo $row['invoicetype'];
		}
		elseif($column == 4){
			echo $row['invoicenumber'];
		}
	}
}

function get_suppliers(){ //παίρνω τους προμηθευτές για να φτιάξω option list κατά την προσθήκη τους
	global $dbh;// το παίρνω από 'db_connect.php'
	$sql = 'SELECT name FROM supplier';
	$sth = $dbh->prepare($sql);
	$sth->execute();
	foreach ($sth as $row){
			echo '<option  value="'.$row['name'].'">'.$row['name'].'</option>';
	}
}
function get_all_products($ordering){ //εμφανίζω τα προϊόντα 1-ταξινόμηση κατα 2,3 όρια για βάση δεδομένων
	global $dbh;// το παίρνω από 'db_connect.php'
	$counter=0;
	$sql = 'SELECT * FROM product ORDER BY '.$ordering;
	$sth = $dbh->prepare($sql);
	$sth->execute();
	foreach ($sth as $row){
			$counter+=1;
			echo  '<tr id="'.$row['id'].'" class="table_row_product">';
			echo   '<td>'.$counter.'</td>';
			echo   '<td>'.$row['supplier'].'</td>';
			echo   '<td>'.$row['invoicetype'].'</td>';
			echo   '<td>'.$row['invoicenumber'].'</td>';
			echo
'<td>'.$row['invoiceday'].'/'.$row['invoicemonth'].'/'.$row['invoiceyear'].'</td>';
			echo   '<td>'.$row['serial'].'</td>';
			echo       '<td>'.$row['type'].'</td>';
			echo       '<td>'.$row['model'].'</td>';
			echo       '<td>'.$row['category'].'</td>';
			echo       '<td>'.$row['description'].'</td>';
			echo       '<td>'.$row['buyprice'].'</td>';
			echo       '<td>'.$row['sellprice'].'</td>';
			echo       '<td>'.$row['status'].'</td>';
			echo       '<td>'.$row['comments'].'</td>';
			echo  '</tr>';
	}
}
function get_all_products_error($ordering){ //εμφανίζω τα προϊόντα 1-ταξινόμηση κατα 2,3 όρια για βάση
δεδομένων
	global $dbh;// το παίρνω από 'db_connect.php'
	$counter=0;
	$sql = 'SELECT * FROM product WHERE status="error"';
	$sth = $dbh->prepare($sql);
	$sth->execute();
	foreach ($sth as $row){
			$counter+=1;
			echo  '<tr id="'.$row['id'].'" class="table_row_product">';
			echo   '<td>'.$counter.'</td>';
			echo   '<td>'.$row['supplier'].'</td>';
			echo   '<td>'.$row['invoicetype'].'</td>';
			echo   '<td>'.$row['invoicenumber'].'</td>';
			echo
'<td>'.$row['invoiceday'].'/'.$row['invoicemonth'].'/'.$row['invoiceyear'].'</td>';
```

```
                                        echo   '<td>'.$row['serial'].'</td>';
                                        echo        '<td>'.$row['type'].'</td>';
                                        echo        '<td>'.$row['model'].'</td>';
                                        echo        '<td>'.$row['category'].'</td>';
                                        echo        '<td>'.$row['description'].'</td>';
                                        echo        '<td>'.$row['buyprice'].'</td>';
                                        echo        '<td>'.$row['sellprice'].'</td>';
                                        echo        '<td>'.$row['status'].'</td>';
                                        echo        '<td>'.$row['comments'].'</td>';
                                        echo  '</tr>';
                }
}
function get_all_products_by_keyword($keyword){ //εμφανίζω τα προϊόντα 1-ταξινόμηση κατα 2,3 όρια για
βάση δεδομένων
                global $dbh;// το παίρνω από 'db_connect.php'
                $counter=0;
                $sql = 'SELECT * FROM product WHERE model LIKE "%'.$keyword.'%" OR serial LIKE
"%'.$keyword.'%"';
                $sth = $dbh->prepare($sql);
                $sth->execute();
                foreach ($sth as $row){
                                        $counter+=1;
                                        echo  '<tr id="'.$row['id'].'" class="table_row_product">';
                                        echo   '<td>'.$counter.'</td>';
                                        echo   '<td>'.$row['supplier'].'</td>';
                                        echo   '<td>'.$row['invoicetype'].'</td>';
                                        echo   '<td>'.$row['invoicenumber'].'</td>';
                                        echo
'<td>'.$row['invoiceday'].'/'.$row['invoicemonth'].'/'.$row['invoiceyear'].'</td>';
                                        echo   '<td>'.$row['serial'].'</td>';
                                        echo        '<td>'.$row['type'].'</td>';
                                        echo        '<td>'.$row['model'].'</td>';
                                        echo        '<td>'.$row['category'].'</td>';
                                        echo        '<td>'.$row['description'].'</td>';
                                        echo        '<td>'.$row['buyprice'].'</td>';
                                        echo        '<td>'.$row['sellprice'].'</td>';
                                        echo        '<td>'.$row['status'].'</td>';
                                        echo        '<td>'.$row['comments'].'</td>';
                                        echo  '</tr>';
                }
}

function get_all_model_products($model){//εμφανίζει τα προϊόντα από ένα model
                global $dbh;// το παίρνω από 'db_connect.php'
                $counter=0;
                $sql = 'SELECT * FROM product WHERE (model = "'.$model.'") AND (status="new" OR
status="repaired")';// varchar τιμη σε διπλά αυτακια!!!! sos sos
                //$sth->bindParam(':model', $model);
                $sth = $dbh->prepare($sql);
                $sth->execute();
                foreach ($sth as $row){
                                        $counter+=1;
                                        echo  '<tr id="'.$row['id'].'" class="table_row_product">';
                                        echo   '<td>'.$counter.'</td>';
                                        echo   '<td>'.$row['supplier'].'</td>';
                                        echo   '<td>'.$row['invoicetype'].'</td>';
                                        echo   '<td>'.$row['invoicenumber'].'</td>';
                                        echo
'<td>'.$row['invoiceday'].'/'.$row['invoicemonth'].'/'.$row['invoiceyear'].'</td>';
                                        echo   '<td>'.$row['serial'].'</td>';
                                        echo        '<td>'.$row['type'].'</td>';
                                        echo        '<td>'.$row['model'].'</td>';
                                        echo        '<td>'.$row['category'].'</td>';
                                        echo        '<td>'.$row['description'].'</td>';
                                        echo        '<td>'.$row['buyprice'].'</td>';
                                        echo        '<td>'.$row['sellprice'].'</td>';
                                        echo        '<td>'.$row['status'].'</td>';
                                        echo        '<td>'.$row['comments'].'</td>';
                                        echo  '</tr>';
                }
}
function get_all_model_sold_products($model){//εμφανίζει τα προϊόντα από ένα model
                global $dbh;// το παίρνω από 'db_connect.php'
                $counter=0;
```

```
		$sql = 'SELECT * FROM product WHERE model = "'.$model.'" AND status = "sold"';// varchar τιμη σε
διπλά αυτακια!!!! sos sos
		//$sth->bindParam(':model', $model);
		$sth = $dbh->prepare($sql);
		$sth->execute();
		foreach ($sth as $row){
					$counter+=1;
					echo  '<tr id="'.$row['id'].'" class="table_row_product">';
					echo   '<td>'.$counter.'</td>';
					echo   '<td>'.$row['supplier'].'</td>';
					echo   '<td>'.$row['invoicetype'].'</td>';
					echo   '<td>'.$row['invoicenumber'].'</td>';
					echo
'<td>'.$row['invoiceday'].'/'.$row['invoicemonth'].'/'.$row['invoiceyear'].'</td>';
					echo   '<td>'.$row['serial'].'</td>';
					echo        '<td>'.$row['type'].'</td>';
					echo        '<td>'.$row['model'].'</td>';
					echo        '<td>'.$row['category'].'</td>';
					echo        '<td>'.$row['description'].'</td>';
					echo        '<td>'.$row['buyprice'].'</td>';
					echo        '<td>'.$row['sellprice'].'</td>';
					echo        '<td>'.$row['status'].'</td>';
					echo        '<td>'.$row['comments'].'</td>';
					echo  '</tr>';
		}
}
function get_all_products_grouped($ordering){//εμφανίζει τα προϊόντα ομαδοποιημένα κατά model
		global $dbh;// το παίρνω από 'db_connect.php'
		$counter=0;
		$sql = 'SELECT *,COUNT(*) as count FROM product GROUP BY model ORDER BY '.$ordering;
		//$sql = 'SELECT * FROM product ORDER BY id';
		$sth = $dbh->prepare($sql);
		$sth->execute();
		foreach ($sth as $row){
					$counter+=1;
					echo  '<tr id="'.$row['id'].'" class="table_row_grouped_product">';
					echo   '<td>'.$counter.'</td>';
					echo   '<td>'.$row['supplier'].'</td>';
					echo   '<td>'.$row['invoicetype'].'</td>';
					echo   '<td>'.$row['invoicenumber'].'</td>';
					echo
'<td>'.$row['invoiceday'].'/'.$row['invoicemonth'].'/'.$row['invoiceyear'].'</td>';
					echo   '<td>'.$row['count'].'</td>';
					//echo   '<td>'.$row['serial'].'</td>';
					echo        '<td>'.$row['type'].'</td>';
					echo        '<td>'.$row['model'].'</td>';
					echo        '<td>'.$row['category'].'</td>';
					echo        '<td>'.$row['description'].'</td>';
					echo        '<td>'.$row['buyprice'].'</td>';
					echo        '<td>'.$row['sellprice'].'</td>';
					//echo     '<td>'.$row['status'].'</td>';
					echo        '<td>'.$row['comments'].'</td>';
					echo  '</tr>';
		}
}
function get_all_sales($ordering){//εμφανίζει τα προϊόντα ομαδοποιημένα κατά model
		global $dbh;// το παίρνω από 'db_connect.php'
		$counter=0;
		$sql = 'SELECT *,COUNT(*) as count FROM product WHERE status="sold" GROUP BY model ORDER
BY '.$ordering;
		//$sql = 'SELECT * FROM product ORDER BY id';
		$sth = $dbh->prepare($sql);
		$sth->execute();
		foreach ($sth as $row){
					$counter+=1;
					echo  '<tr id="'.$row['id'].'" class="table_row_grouped_sales">';
					echo   '<td>'.$counter.'</td>';
					echo   '<td>'.$row['supplier'].'</td>';
					echo   '<td>'.$row['invoicetype'].'</td>';
					echo   '<td>'.$row['invoicenumber'].'</td>';
					echo
'<td>'.$row['invoiceday'].'/'.$row['invoicemonth'].'/'.$row['invoiceyear'].'</td>';
					echo   '<td>'.$row['count'].'</td>';
					echo   '<td>'.$row['serial'].'</td>';
```

```
                                        echo        '<td>'.$row['type'].'</td>';
                                        echo        '<td>'.$row['model'].'</td>';
                                        echo        '<td>'.$row['category'].'</td>';
                                        echo        '<td>'.$row['description'].'</td>';
                                        echo        '<td>'.$row['buyprice'].'</td>';
                                        echo        '<td>'.$row['sellprice'].'</td>';
                                        echo        '<td>'.$row['status'].'</td>';
                                        echo        '<td>'.$row['comments'].'</td>';
                                        echo  '</tr>';
                }
}
function get_all_suppliers($ordering){ //εμφανίζω τους προμηθευτές
                global $dbh;// το παίρνω από 'db_connect.php'
                $counter=0;
                $sql = 'SELECT * FROM supplier ORDER BY '.$ordering;
                $sth = $dbh->prepare($sql);
                $sth->execute();
                foreach ($sth as $row){
                                        $counter+=1;
                                        echo  '<tr id="'.$row['id'].'" class="table_row_supplier">'; //class table_row ειναι
για το δεξί κλικ.
                                        echo   '<td>'.$counter.'</td>';
                                        echo   '<td>'.$row['afm'].'</td>';
                                        echo   '<td>'.$row['type'].'</td>';
                                        echo   '<td>'.$row['name'].'</td>';
                                        echo   '<td>'.$row['address'].'</td>';
                                        echo   '<td>'.$row['phone'].'</td>';
                                        echo   '<td>'.$row['fax'].'</td>';
                                        echo        '<td>'.$row['comments'].'</td>';
                                        echo  '</tr>';
                }
}

function get_all_clients($ordering){ //εμφανίζω τους προμηθευτές
                global $dbh;// το παίρνω από 'db_connect.php'
                $counter=0;
                $sql = 'SELECT * FROM client ORDER BY '.$ordering;
                $sth = $dbh->prepare($sql);
                $sth->execute();
                foreach ($sth as $row){
                                        $counter+=1;
                                        echo  '<tr id="'.$row['id'].'" class="table_row_client">'; //class table_row ειναι
για το δεξί κλικ.
                                        echo   '<td>'.$counter.'</td>';
                                        echo   '<td>'.$row['afm'].'</td>';
                                        echo   '<td>'.$row['clienttype'].'</td>';
                                        echo   '<td>'.$row['name'].'</td>';
                                        echo   '<td>'.$row['address'].'</td>';
                                        echo   '<td>'.$row['phone'].'</td>';
                                        echo   '<td>'.$row['fax'].'</td>';
                                        echo        '<td>'.$row['comments'].'</td>';
                                        echo  '</tr>';
                }
}
function get_all_invoice_clients(){ //εμφανίζω τους προμηθευτές
                global $dbh;// το παίρνω από 'db_connect.php'
                $sql = 'SELECT * FROM client';
                $sth = $dbh->prepare($sql);
                $sth->execute();
                foreach ($sth as $row){
                                        echo  '<option value="'.$row['name'].'"
id="'.$row['id'].'">'.$row['name'].'</option>'; //class table_row ειναι για το δεξί κλικ.
                }
}
function get_all_supplier_invoices($ordering){ //
                global $dbh;// το παίρνω από 'db_connect.php'
                $counter=0;
                $sql = 'SELECT DISTINCT invoicenumber,invoiceday,invoicemonth,invoiceyear,supplier,invoicetype
FROM product ORDER BY '.$ordering;
                $sth = $dbh->prepare($sql);
                $sth->execute();
                foreach ($sth as $row){
                                        $counter+=1;
```

```
                                    echo  '<tr id="'.$row['invoicenumber'].'" class="table_row_supplier_invoices">';
//class table_row ειναι για το δεξί κλικ.
                                    echo   '<td>'.$counter.'</td>';
                                    echo   '<td>'.$row['invoicenumber'].'</td>';
                                    echo   '<td>'.$row['invoiceday'].'-'.$row['invoicemonth'].'-
'.$row['invoiceyear'].'</td>';
                                    echo   '<td>'.$row['supplier'].'</td>';
                                    echo   '<td>'.$row['invoicetype'].'</td>';
                                    echo  '</tr>';
            }
}
function get_all_customer_invoices($ordering){ //
            global $dbh;// το παίρνω από 'db_connect.php'
            $counter=0;
            $sql = 'SELECT DISTINCT invoicenumber,invoiceday,invoicemonth,invoiceyear,client,invoicetype,
invoicereason, paytype, cost FROM clientinvoice ORDER BY '.$ordering;
            $sth = $dbh->prepare($sql);
            $sth->execute();
            foreach ($sth as $row){
                                    $counter+=1;
                                    echo  '<tr id="'.$row['invoicenumber'].'" class="table_row_supplier_invoices">';
//class table_row ειναι για το δεξί κλικ.
                                    echo   '<td>'.$counter.'</td>';
                                    echo   '<td>'.$row['invoicenumber'].'</td>';
                                    echo   '<td>'.$row['invoiceday'].'-'.$row['invoicemonth'].'-
'.$row['invoiceyear'].'</td>';
                                    echo   '<td>'.$row['client'].'</td>';
                                    echo   '<td>'.$row['invoicetype'].'</td>';
                                    echo   '<td>'.$row['invoicereason'].'</td>';
                                    echo   '<td>'.$row['paytype'].'</td>';
                                    echo   '<td>'.$row['cost'].'</td>';
                                    echo   '<td>'.$row['cost']*0.23.'</td>';
                                    $total=$row['cost']+$row['cost']*0.23;
                                    echo   '<td>'.$total.'</td>';
                                    echo  '</tr>';
            }
}
function delete_record($id,$table){ //Διαγραφή εγγραφής
            global $dbh;// το παίρνω από 'db_connect.php'
            $sql = 'DELETE FROM table= :table WHERE id= :id';
                                                $sth = $dbh->prepare($sql);
                                                // Bind variables to your statement
                                                $sth->bindParam(':table', $table);
                                                $sth->bindParam(':id', $id);
                                                // Flip the switch
                                                $sth->execute();
}

function get_model_from_id($id){ //βρίσκει το model για δωθέν id
            global $dbh;// το παίρνω από 'db_connect.php'
            $sql = 'SELECT model FROM product WHERE id= :id';
                                                $sth = $dbh->prepare($sql);
                                                // Bind variables to your statement
                                                $sth->bindParam(':id', $id);
                                                // Flip the switch
                                                $sth->execute();
            foreach ($sth as $row){
                                    return $row['model'];
            }
}

function get_all_invoice_products($invoicenumber){//εμφανίζει τα προϊόντα ομαδοποιημένα κατά model
            global $dbh;// το παίρνω από 'db_connect.php'
            $counter=0;
            $sql = 'SELECT *,COUNT(*) as count FROM product WHERE invoicenumber='.$invoicenumber.'
GROUP BY model';
            $sth = $dbh->prepare($sql);
            $sth->execute();
            foreach ($sth as $row){
                                    $counter+=1;
                                    echo  '<tr>';
                                    echo   '<td>'.$counter.'</td>';
                                    echo   '<td>'.$row['model'].'</td>';
                                    echo   '<td>Τεμάχια</td>';
```

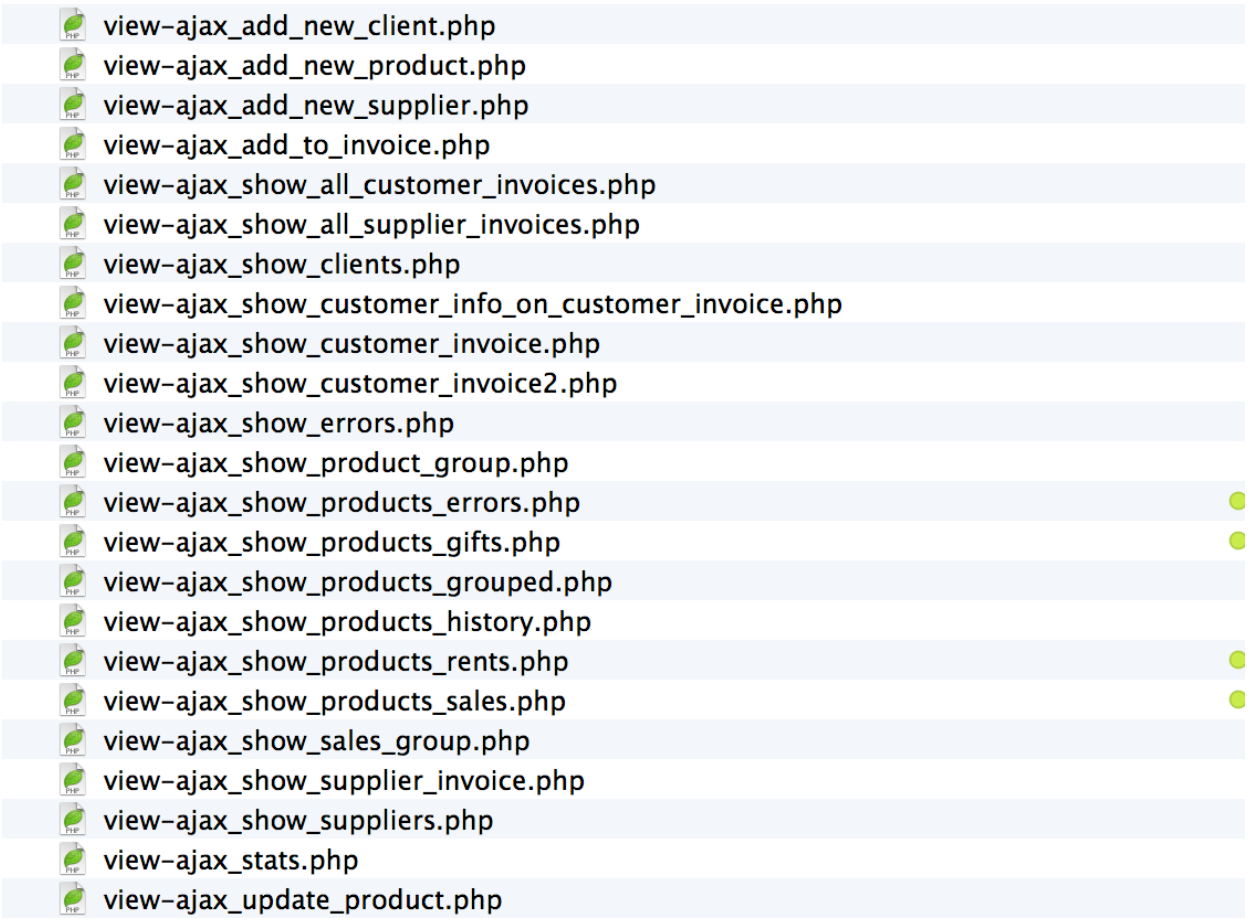
```
                                        echo   '<td>'.$row['count'].'</td>';
                                        echo   '<td>'.$row['buyprice'].'</td>';
                                        echo   '<td>'.$row['buyprice']*$row['count'].'</td>';
                                        echo   '<td>'.$row['buyprice']*$row['count']*0.23.'</td>';
                                        echo  '</tr>';
                }
}

function get_invoice_cost($invoicenumber){//εμφανίζει τα προϊόντα ομαδοποιημένα κατά model
                global $dbh;// το παίρνω από 'db_connect.php'
                $totalcost=0;
                $sql = 'SELECT *,COUNT(*) as count FROM product WHERE invoicenumber='.$invoicenumber.'
GROUP BY model';
                $sth = $dbh->prepare($sql);
                $sth->execute();
                foreach ($sth as $row){
                                        $totalcost+=$row['buyprice']*$row['count'];
                }
                return $totalcost;
}
function count_month_money($month){//εμφανίζει τα προϊόντα ομαδοποιημένα κατά model
                global $dbh;// το παίρνω από 'db_connect.php'
                $totalmoney=0;
                $sql = 'SELECT * FROM product WHERE invoicemonth= "'.$month.'" AND status="sold"';
                $sth = $dbh->prepare($sql);
                $sth->execute();
                foreach ($sth as $row){
                                        $totalmoney=$totalmoney+$row['sellprice'];
                }
                echo $totalmoney;
}

function get_invoice_info($invoicenumber){//εμφανίζει τα προϊόντα ομαδοποιημένα κατά model
                global $dbh;// το παίρνω από 'db_connect.php'
                $sql = 'SELECT * FROM product WHERE invoicenumber='.$invoicenumber.' LIMIT 1';
                $sth = $dbh->prepare($sql);
                $sth->execute();
                $rows = $sth->fetchAll(PDO::FETCH_ASSOC);
      //echo $rows[0]['id'];
      //print_r($rows);
   return $rows;
}

function get_supplier_info($supplier){//εμφανίζει τα προϊόντα ομαδοποιημένα κατά model
                global $dbh;// το παίρνω από 'db_connect.php'
                $sql = 'SELECT * FROM supplier WHERE name= "'.$supplier.'"';
                $sth = $dbh->prepare($sql);
                $sth->execute();
                $rows = $sth->fetchAll(PDO::FETCH_ASSOC);
   return $rows;
}

function get_customer_info($customer){//
                global $dbh;// το παίρνω από 'db_connect.php'
                $sql = 'SELECT * FROM client WHERE name= "'.$customer.'"';
                $sth = $dbh->prepare($sql);
                $sth->execute();
                $rows = $sth->fetchAll(PDO::FETCH_ASSOC);
   return $rows;
}

function create_customer_final_invoice_products(){
                global $dbh;// το παίρνω από 'db_connect.php'
                $_SESSION['cost']=0;
                $_SESSION['fpa']=0;
                $_SESSION['total']=0;
                $x=sizeof($_POST['pr_model']);
                $counter=1;
                //echo $_POST['pr_quantity'][0];
                for ($i = 0; $i < $x; $i++) {
                                $sql = 'SELECT * FROM product WHERE model= "'.$_POST['pr_model'][$i].'"';
                                $sth = $dbh->prepare($sql);
                                $sth->execute();
                                $rows = $sth->fetchAll(PDO::FETCH_ASSOC);
```

```
                        echo '<tr>';
                        echo '<td>'.$counter.'</td>';
                        echo '<td>'.$rows[$i]['serial'].'</td>';
                        echo '<td>Τεμάχια</td>';
                        echo '<td><input type="text" name="pr_model[]"
value="'.$_POST['pr_model'][$i].'"></td>';
                        echo '<td><input type="text" name="pr_quantity[]"
value="'.$_POST['pr_quantity'][$i].'"></td>';
                        echo '<td>'.$rows[$i]['sellprice'].'</td>';
                        echo '<td>'.$rows[$i]['sellprice']*$_POST['pr_quantity'][$i].'</td>';
                        echo '<td>'.$rows[$i]['sellprice']*$_POST['pr_quantity'][$i]*0.23.'</td>';
                        echo '<td><input type="hidden" name="pr_id[]" value="'.$rows[$i]['id'].'"></td>';
                //echo $_POST['pr_model'][$i].' '.$_POST['pr_quantity'][$i].'<br/>';
                echo '</tr>';
                $counter+=1;
                $_SESSION['cost']=$_SESSION['cost']+$rows[$i]['sellprice']*$_POST['pr_quantity'][$i];
                        $_SESSION['fpa']=$_SESSION['fpa']+$rows[$i]['sellprice']*$_POST['pr_quantity'][$i]*0.23;
                        $_SESSION['total']=$_SESSION['cost']+$_SESSION['fpa'];
        }
}

function checkuser(){
        echo $_SESSION['user'];
}
?>
```

4.3.2.2 **VIEW**

Παρακάτω βλέπουμε αρχεία που στο σύνολό τους υλοποιούν το κομμάτι View της αρχιτεκτονικής MVC. Ρόλος τους είναι απλά και μόνο να παρουσιάζουν διάφορες πληροφορίες πολλές από τις οποίες προέρχονται από το Model.

Το View δεν ασχολείται με κανέναν τρόπο με την μορφοποίηση και τον τρόπο παρουσίασης της πληροφορίας.



4.3.2.3 **Controller**

Στην προσωπική μου προσέγγιση ως προς την αρχιτεκτονική MVC, καθώς δεν υπάρχει μία μοναδική, υλοποιώ έναν κεντρικό controller.

Ο Controller λαμβάνει δράση αμέσως μόλις ο χρήστης ταυτοποιηθεί από το index.php.

Το περιεχόμενο του αρχείου controller.php είναι τα παρακάτω:

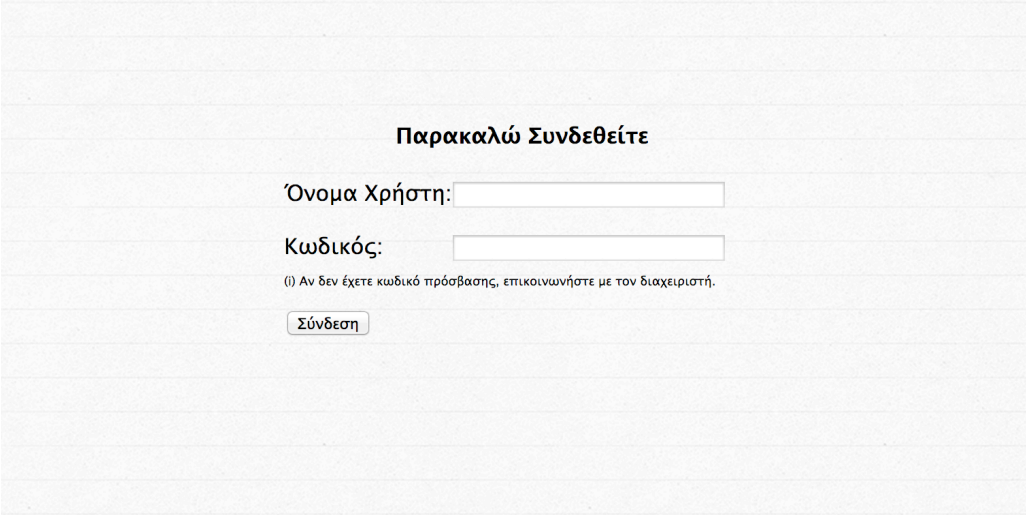
```
<?php
 session_start();
 require_once('model-php_library.php');
?>
<html>
<head>
<title>Med Manager</title>
        <link rel="stylesheet" type="text/css" href="style/main.css">
        <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>

        <script src="http://code.highcharts.com/highcharts.js"></script>
        <script src="http://code.highcharts.com/modules/exporting.js"></script>

        <script type="text/javascript" src="js/jquery-ui.min.js"></script>
```

```
        <link rel="stylesheet" type="text/css" href="style/jquery-ui.structure.min.css">
        <link rel="stylesheet" type="text/css" href="style/jquery-ui.theme.css">

  <script src="right_click_menu/jquery.contextMenu.js" type="text/javascript"></script>
  <link href="right_click_menu/jquery.contextMenu.css" rel="stylesheet" type="text/css"
/>

</head>

<body onload="load_show_all_products_grouped('count')">
        <div id="main_container">
                <div id="top_menu">
                        <div id="logo"><p>MedManager v2014</p></div>
                        <div id="menu">
                           <button onclick="load_show_customer_invoice()">Νέο
Παραστατικό</button><button onclick="gifts('id')">Δωρεές</button><button
onclick="rents('id')">Δανεισμοί</button><button
onclick="load_show_all_sales('id')">Πωλήσεις</button><button
onclick="errors('id')">Βλάβες</button><button
onclick="returns('id')">Επιστροφές</button>
                           <form name="ajaxform" id="ajaxform2" action="search.php"
method="POST">
                                <input id="searchinput" type="text" name="keyword"
value="αναζήτηση...">
                           </form>
                        </div>
                </div><!--top_menu-->
                <div id="left_sidebar">
                        <div class="left_sidebar_category">
                                <p class="left_sidebar_title">Πίνακας Ελέγχου</p>
                                <?php echo 'Γειά σας, <b>'.$_SESSION['user'].'</b>';?>
                                <a href="logout.php">Αποσύνδεση</a>
                        </div>
                        <div class="left_sidebar_category">
                                <p class="left_sidebar_title">Προϊόντα</p>
                                <button
onclick="load_show_all_products('id')">Προβολή Ιστορικού</button>
                                <button
onclick="load_show_all_products_grouped('id')">Προβολή Προϊόντων</button>

                                <button onclick="load_add_new_product()">Προσθήκη
νέου Προϊόντος</button>
                        </div>
                        <div class="left_sidebar_category">
                                <p class="left_sidebar_title">Τιμολόγια</p>

                                <button
onclick="load_show_all_suppliers_invoices('id')">Προμηθευτών</button>
                                <button
onclick="load_show_all_customers_invoices('id')">Πελατών</button>
                        </div>
                        <div class="left_sidebar_category">
                                <p class="left_sidebar_title">Στατιστικά</p>

                                <button onclick="stats()">Πωλήσεων</button>
                        </div>
                        <div class="left_sidebar_category">
                                <p class="left_sidebar_title">Τεχνικό Τμήμα</p>
```

```
                                        <button onclick="errors(id)">Προβολή
βλαβών</button>
                                </div>
                                <div class="left_sidebar_category">
                                        <p class="left_sidebar_title">Προμηθευτές</p>

                                        <button
onclick="load_show_all_suppliers('id')">Προβολή Προμηθευτών</button>
                                        <button onclick="load_add_new_supplier()">Προσθήκη
νέου Προμηθευτή</button>
                                </div>
                                <div class="left_sidebar_category">
                                        <p class="left_sidebar_title">Πελάτες</p>

                                        <button onclick="load_show_all_clients('id')">Προβολή
Πελατών</button>
                                        <button onclick="load_add_new_client()">Προσθήκη
νέου Πελάτη</button>
                                </div>
                        </div>
                        <div id="container">
                                <main id="main">

                                </main>
                                <div id="stats"></div>
                        </div><!--container-->
        </div><!--main_container-->

        <script type="text/javascript" src="js/ajax.js"></script>

        <script type="text/javascript">
                $( "#searchinput" ).keyup(function() {
                  sendform3();
                });
        </script>

        <script type="text/javascript">//όταν κάνω κλικ να γίνεται κενό
                $("#searchinput").focus(
          function(){
            $(this).val('');
          });
        </script>
        <script type="text/javascript">//σταματάω την υποβολή με το enter, στο πεδιο
της αναζήτησης
                $('#searchinput').keypress(function(event) {
            if (event.keyCode == 13) {
              event.preventDefault();
                                return false;
            }
          });
        </script>

</body>
</html>
```

5. Παρουσίαση της εφαρμογής

Το πρώτο στοιχείο που βλέπει κάθε χρήστης που προσπαθεί να συνδεθεί στην εφαρμογή είναι η σελίδα ταυτοποίησης που αναγνωρίζει την ιδιότητα του χρήστη και αναλόγως τον προωθεί στην κατάλληλη σελίδα.

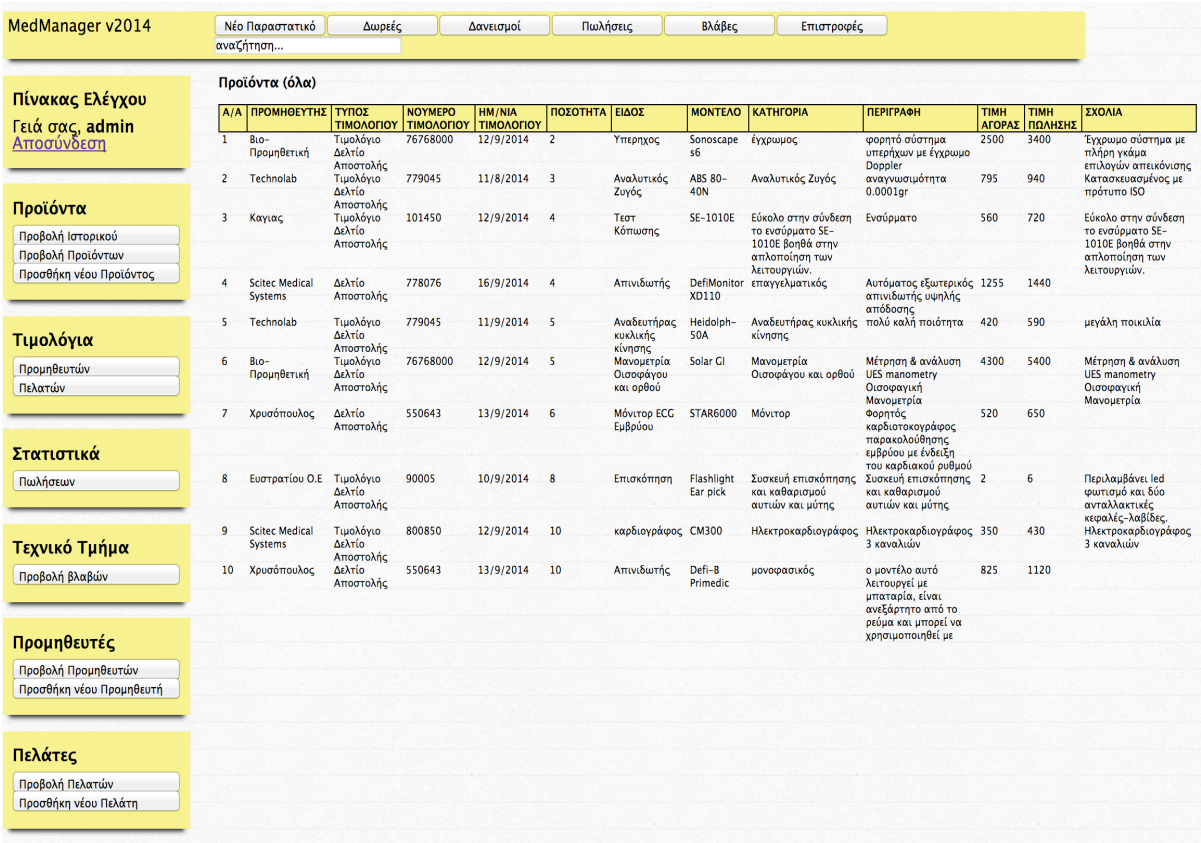


Ας υποθλεσουμε πως συνδεόμαστε στην εφαρμογή ως διαχειριστής. Ο διαχειριστής έχει πλήρη δικαιώματα, συνεπώς θα προωθηθεί στη σελίδα όπου εμφανίζονται όλες οι δυνατές λειτουργίες της εφαρμογής.

Στην παρακάτω εικόνα φαίνεται η αρχική οθόνη του χρήστη. Ο σχεδιασμός της εφαρμογής ακουλουθεί συγκεκριμένη φιλοσοφία ώστε να θυμίζει εφαρμογή και όχι κλασσική ιστοσελίδα.

Στο πάνω οριζόντιο μέρος και στην αριστερή πλευρά υπάρχουν εικόνες που βοηθούν τον χρήστη να προβάλει το περιεχόμενο που θέλει, χωρίας ιδιαίτερη αναζήτηση.

Χωρίς ο χρήστης να έχει επιλέξει κάτι από τις παραπάνω επιλογές, η εφαρμογή εμφανίζει αυτόματα μια απογραφή της αποθήκης, με όλα τα διαθέσιμα προϊόντα ομαδοποιημένα κατά μοντέλο κατασκευαστή.



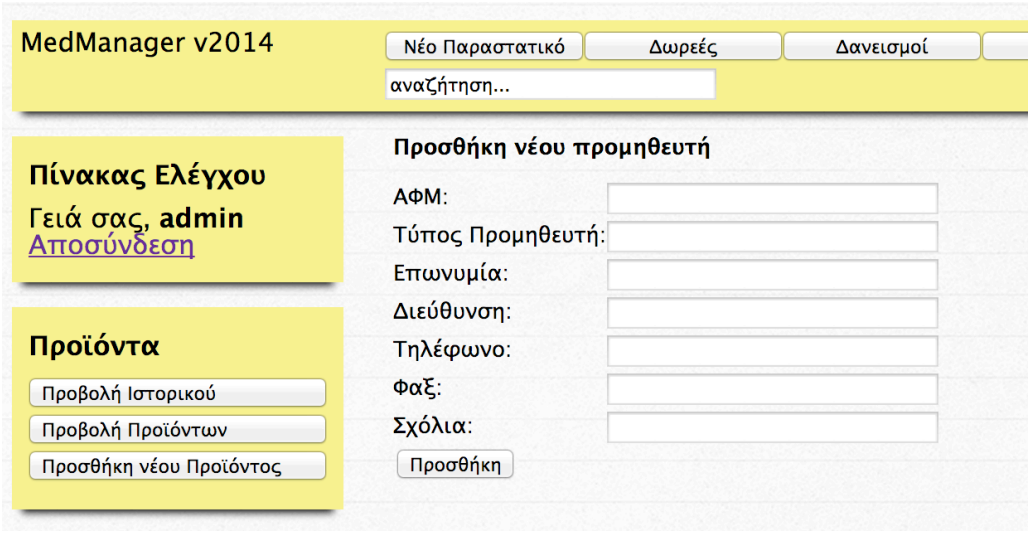
Α/Α	ΠΡΟΜΗΘΕΥΤΗΣ	ΤΥΠΟΣ ΤΙΜΟΛΟΓΙΟΥ	ΝΟΥΜΕΡΟ ΤΙΜΟΛΟΓΙΟΥ	ΗΜ/ΝΙΑ ΤΙΜΟΛΟΓΙΟΥ	ΠΟΣΟΤΗΤΑ	ΕΙΔΟΣ	ΜΟΝΤΕΛΟ	ΚΑΤΗΓΟΡΙΑ	ΠΕΡΙΓΡΑΦΗ	ΤΙΜΗ ΑΓΟΡΑΣ	ΤΙΜΗ ΠΩΛΗΣΗΣ	ΣΧΟΛΙΑ
1	Βιο-Προμηθετική	Τιμολόγιο Δελτίο Αποστολής	76768000	12/9/2014	2	Υπερηχος	Sonoscape s6	έγχρωμος	φορητό σύστημα υπερήχων με έγχρωμο Doppler	2500	3400	Έγχρωμο σύστημα με πλήρη γκάμα επιλογών απεικόνισης
2	Technolab	Τιμολόγιο Δελτίο Αποστολής	779045	11/8/2014	3	Αναλυτικός Ζυγός	ABS 80-40N	Αναλυτικός Ζυγός	αναγνωσιμότητα 0.0001gr	795	940	Κατασκευασμένος με πρότυπο ISO
3	Καγιας	Τιμολόγιο Δελτίο Αποστολής	101450	12/9/2014	4	Τεστ Κόπωσης	SE-1010E	Εύκολο στην σύνδεση το ενσύρματο SE-1010E βοηθά στην απλοποίηση των λειτουργιών.	Ενσύρματο	560	720	Εύκολο στην σύνδεση το ενσύρματο SE-1010E βοηθά στην απλοποίηση των λειτουργιών.
4	Scitec Medical Systems	Δελτίο Αποστολής	778076	16/9/2014	4	Απινιδωτής	DefiMonitor XD110	επαγγελματικός	Αυτόματος εξωτερικός απινιδωτής υψηλής απόδοσης	1255	1440	
5	Technolab	Τιμολόγιο Δελτίο Αποστολής	779045	11/9/2014	5	Αναδευτήρας κυκλικής κίνησης	Heidolph-50A	Αναδευτήρας κυκλικής κίνησης	πολύ καλή ποιότητα	420	590	μεγάλη ποικιλία
6	Βιο-Προμηθετική	Τιμολόγιο Δελτίο Αποστολής	76768000	12/9/2014	5	Μανομετρία Οισοφάγου και ορθού	Solar GI	Μανομετρία Οισοφάγου και ορθού	Μέτρηση & ανάλυση UES manometry Οισοφαγική Μανομετρία	4300	5400	Μέτρηση & ανάλυση UES manometry Οισοφαγική Μανομετρία
7	Χρυσόπουλος	Δελτίο Αποστολής	550643	13/9/2014	6	Μόνιτορ ECG Εμβρύου	STAR6000	Μόνιτορ	Φορητός καρδιοτοκογράφος παρακολούθησης εμβρύου με ένδειξη του καρδιακού ρυθμού	520	650	
8	Ευστρατίου Ο.Ε	Τιμολόγιο Δελτίο Αποστολής	90005	10/9/2014	8	Επισκόπηση	Flashlight Ear pick	Συσκευή επισκόπησης και καθαρισμού αυτιών και μύτης	Συσκευή επισκόπησης και καθαρισμού αυτιών και μύτης	2	6	Περιλαμβάνει led φωτισμό και δύο ανταλλακτικές κεφαλές-λαβίδες.
9	Scitec Medical Systems	Τιμολόγιο Δελτίο Αποστολής	800850	12/9/2014	10	καρδιογράφος	CM300	Ηλεκτροκαρδιογράφος	Ηλεκτροκαρδιογράφος 3 καναλιών	350	430	Ηλεκτροκαρδιογράφος 3 καναλιών
10	Χρυσόπουλος	Δελτίο Αποστολής	550643	13/9/2014	10	Απινιδωτής	Defi-B Primedic	μονοφασικός	ο μοντέλο αυτό λειτουργεί με μπαταρία, είναι ανεξάρτητο από το ρεύμα και μπορεί να χρησιμοποιηθεί με	825	1120	

Τα προϊόντα παρουσιάζονται σε μορφή πίνακα. Κάνοντας κλικ στον αντίστοιχο τίτλο γίνεται αναταξινόμηση για να βρεί ο χρήστης πιο εύκολα αυτό που ψάχνει.

5.1 Προσθήκη προμηθευτή

Για να είναι δυνατή η καταχώρηση προϊόντων ενός προμηθευτή, πρέπει πρώτα αυτός να έχει καταγραφεί στην εφαρμογή, μέσω της φόρμας προσθήκης προμηθευτή.

Η επιλογή «Προσθήκη νέου Προμηθευτή» στην αριστερή στήλη, εμφανίζει την παρακάτω φόρμα:

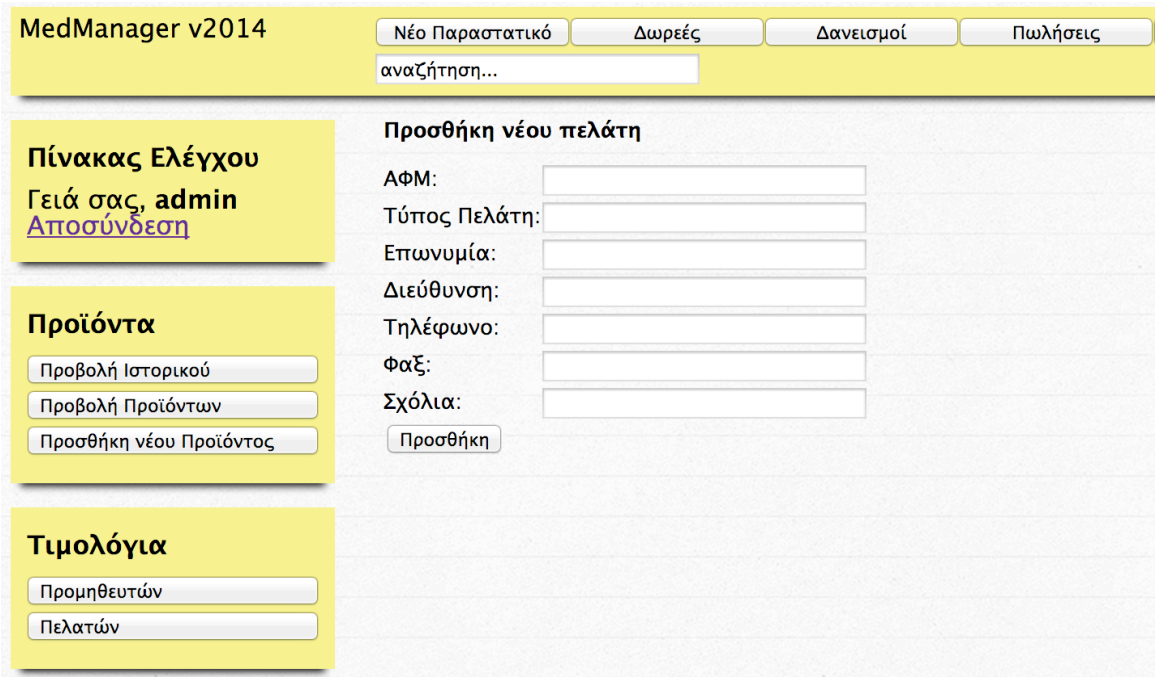


Ο χρήστης καλείται να συμπληρώσει μια φορά τα παραπάνω στοιχεία και πλέον απλά θα επιλέγει δυναμικά τον προμηθευτή όταν εισάγει ένα προϊόν και τα στοιχεία θα συνδέονται αυτόματα με αυτο.

5.2 Προσθήκη πελάτη

Στην περίπτωση όπου αποστέλλεται ένα προϊόν είτε σε πελάτη είτε αυτό επιστρέφεται είτε πωλείται πρέπει να γίνει επιλογή ενός πελάτη από την βάση δεδομένων και να εκδοθεί παραστατικό ανάλογα με τις ανάγκες της κάθε περίπτωσης.

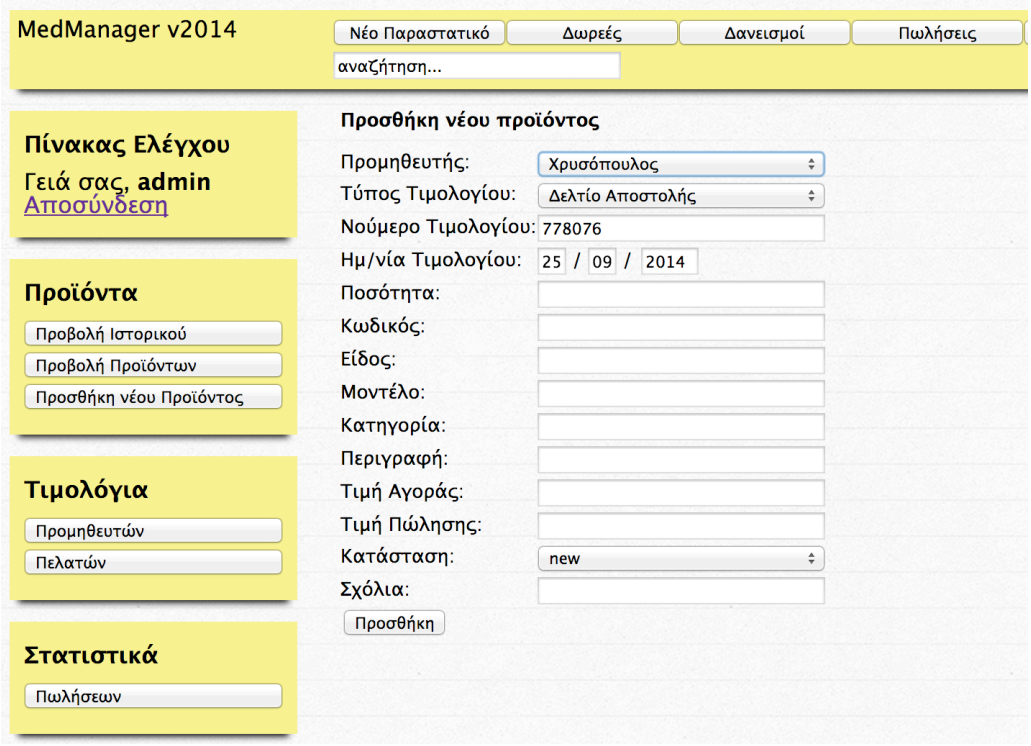
Παρακάτω βλέπετε την φόρμα εισαωγωγής πελάτη στην βάση δεδομένων της εφαρμογής:



Τα στοιχεία που συλλέγονται για τον πελάτη είναι παρόμοια με αυτά που συλλέγονται και για τον προμηθευτή.

5.3 Προσθήκη προϊόντος

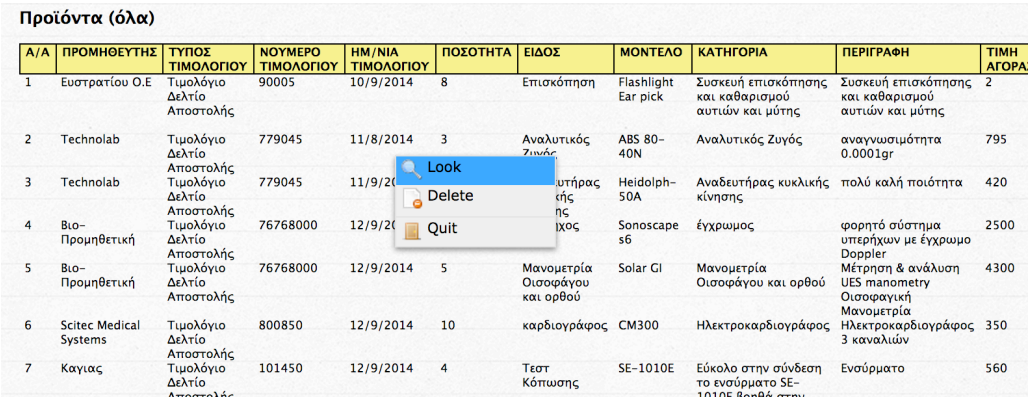
Κατά την προσθήκη προϊόντων βλέπουμε την παρακάτω φόρμα:



Στην φόρμα αυτή, εμφανίζεται αυτόματα μια λίστα με τους προμηθευτές που έχουμε καταχωρήσει και άλλα πεδιά ώστε να αποθηκεύσουμε στοιχεία για το αντικείμενο που εισάγουμε. Ο αριθμός τιμολογίου πρέπει να παραμένει ίδιος για προϊόντα που ανήκουν στο ίδιο τιμολόγιο. Η ημερομηνία εμφανίζεται αυτόματα για να μας διευκολύνει, αλλά υπάρχει και δυνατότητα αλλαγής.

5.4 Προβολή προϊόντος

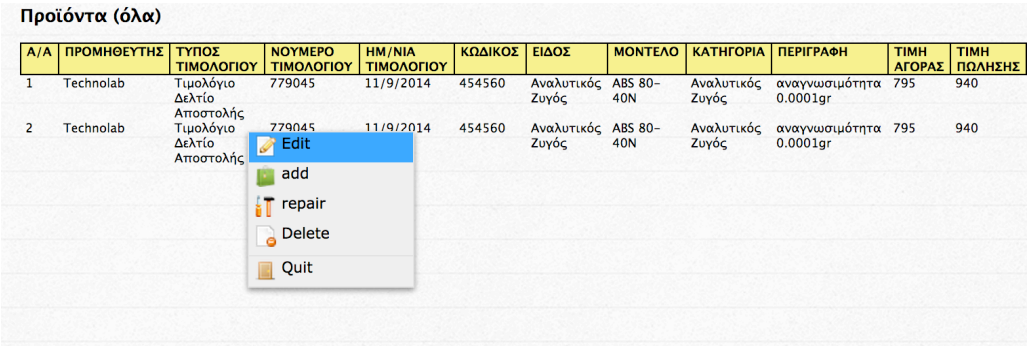
Όπως αναφέραμε αρχικά, τα προϊόντα εμφανίζονται ομαδοποιημένα βάσει του μοντέλου κατασκευαστή και της ποσότητάς τους. Για να προβάλουμε μια κατηγορία και να δούμε και επεξεργαστούμε τα προϊόντα αυτά ανεξάρτητα χρησιμοποιούμε το δεξί κλικ. Μια δυνατότητα της εφαρμογής μας ιδιαίτερα χρήσιμη, που προσαρμόζεται κάθε φορά στο περιεχόμενο που προβάλεται και εμφανίζει αντίστοιχες επιλογές.



Α/Α	ΠΡΟΜΗΘΕΥΤΗΣ	ΤΥΠΟΣ ΤΙΜΟΛΟΓΙΟΥ	ΝΟΥΜΕΡΟ ΤΙΜΟΛΟΓΙΟΥ	ΗΜ/ΝΙΑ ΤΙΜΟΛΟΓΙΟΥ	ΠΟΣΟΤΗΤΑ	ΕΙΔΟΣ	ΜΟΝΤΕΛΟ	ΚΑΤΗΓΟΡΙΑ	ΠΕΡΙΓΡΑΦΗ	ΤΙΜΗ ΑΓΟΡΑ
1	Ευστρατίου Ο.Ε	Τιμολόγιο Δελτίο Αποστολής	90005	10/9/2014	8	Επισκόπηση	Flashlight Ear pick	Συσκευή επισκόπησης και καθαρισμού αυτιών και μύτης	Συσκευή επισκόπησης και καθαρισμού αυτιών και μύτης	2
2	Technolab	Τιμολόγιο Δελτίο Αποστολής	779045	11/8/2014	3	Αναλυτικός Ζυγός	ABS 80-40N	Αναλυτικός Ζυγός	αναγνωσιμότητα 0.0001gr	795
3	Technolab	Τιμολόγιο Δελτίο Αποστολής	779045	11/9/2		υτήρας ής	Heidolph-50A	Αναδευτήρας κυκλικής κίνησης	πολύ καλή ποιότητα	420
4	Βιο-Προμηθετική	Τιμολόγιο Δελτίο Αποστολής	76768000	12/9/2		ιχος	Sonoscape s6	έγχρωμος	φορητό σύστημα υπερήχων με έγχρωμο Doppler	2500
5	Βιο-Προμηθετική	Τιμολόγιο Δελτίο Αποστολής	76768000	12/9/2014	5	Μανομετρία Οισοφάγου και ορθού	Solar GI	Μανομετρία Οισοφάγου και ορθού	Μέτρηση & ανάλυση UES manometry Οισοφαγική Μανομετρία	4300
6	Scitec Medical Systems	Τιμολόγιο Δελτίο Αποστολής	800850	12/9/2014	10	καρδιογράφος	CM300	Ηλεκτροκαρδιογράφος	Ηλεκτροκαρδιογράφος 3 καναλιών	350
7	Καγιας	Τιμολόγιο Δελτίο	101450	12/9/2014	4	Τεστ Κόπωσης	SE-1010E	Εύκολο στην σύνδεση το ενσύρματο SE-	Ενσύρματο	560

Στην περίπτωση αυτή μπορούμε να εμφανίζουμε την κατηγορία ή να την διαγράψουμε.

Όταν όμως πρόκειται για ένα προϊόν και όχι κατηγορία βλέπουμε πολύ περισσότερα στοιχεία:



Α/Α	ΠΡΟΜΗΘΕΥΤΗΣ	ΤΥΠΟΣ ΤΙΜΟΛΟΓΙΟΥ	ΝΟΥΜΕΡΟ ΤΙΜΟΛΟΓΙΟΥ	ΗΜ/ΝΙΑ ΤΙΜΟΛΟΓΙΟΥ	ΚΩΔΙΚΟΣ	ΕΙΔΟΣ	ΜΟΝΤΕΛΟ	ΚΑΤΗΓΟΡΙΑ	ΠΕΡΙΓΡΑΦΗ	ΤΙΜΗ ΑΓΟΡΑΣ	ΤΙΜΗ ΠΩΛΗΣΗΣ
1	Technolab	Τιμολόγιο Δελτίο Αποστολής	779045	11/9/2014	454560	Αναλυτικός Ζυγός	ABS 80-40N	Αναλυτικός Ζυγός	αναγνωσιμότητα 0.0001gr	795	940
2	Technolab	Τιμολόγιο Δελτίο Αποστολής	779045	11/9/2014	454560	Αναλυτικός Ζυγός	ABS 80-40N	Αναλυτικός Ζυγός	αναγνωσιμότητα 0.0001gr	795	940

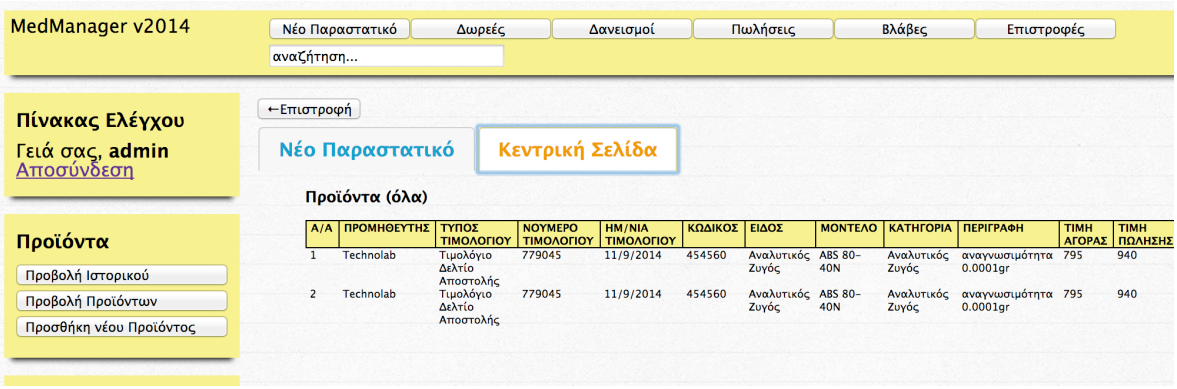
Με την επιλογή edit μπορούμε να ανανεώσουμε τα στοιχεία ενός αντικειμένου.

Με την επιλογή add προσθέτουμε το προϊόν στο παραστατικό, το οποίο λειτουργεί με την ίδια λογική που λειτουργεί και ένα καλάθι αγορών.

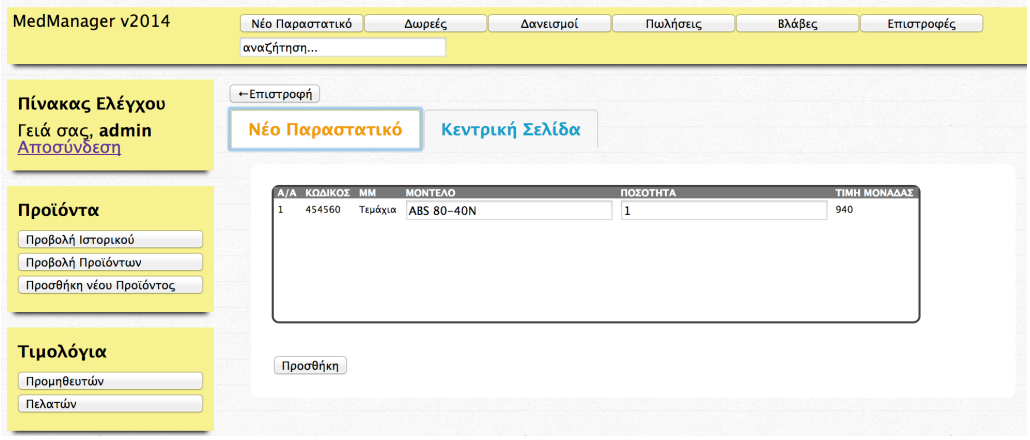
Με την επιλογή edit δηλώνουμε πως το προϊόν αυτό έχει κάποια βλάβη. Η ειδοποίηση αυτή θα εμφανιστεί στο τεχνικό τμήμα. Μπορούμε επίσης να συμπεριλάβουμε και κάποια σχόλια για να διευκολύνουμε τους τεχνικούς και να περιγράψουμε σωστά το πρόβλημα.

5.5 Δημιουργία παραστατικού

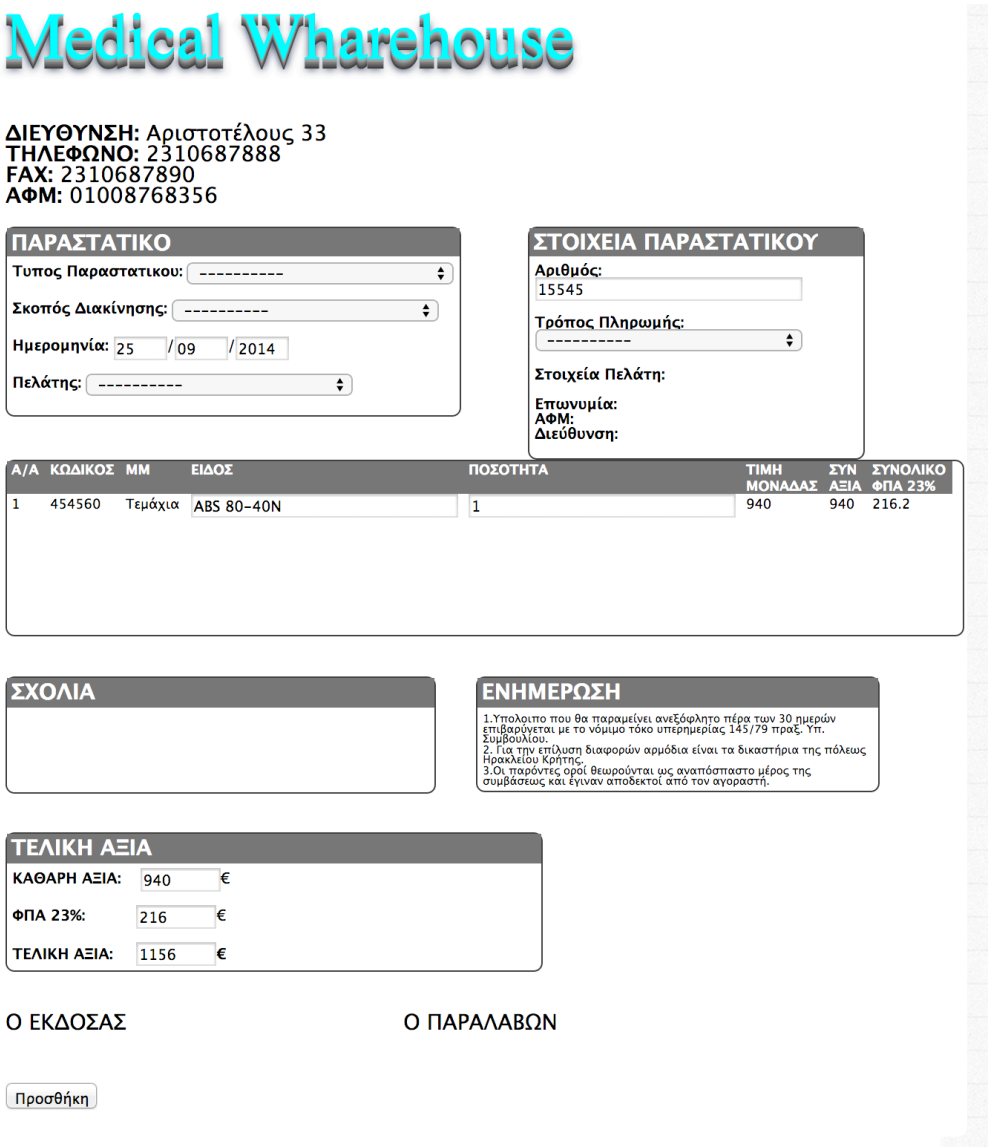
Επιλέγοντας την αντίστοιχη λειτουργία, εμφανίζονται στο περιβάλλον του χρήστη δύο καρτέλες, μία στην οποία βλέπει τα προϊόντα που έχει επιλέξει και μια στην οποία μπορεί να συνεχίζει την περιήγησή στην εφαρμογή προς δική του διευκόλυνση.



Όταν έχει επιλέξει τα προϊόντα και τις ποσότητες που θέλει,



κάνει κλικ στο κουμπί προσθήκη και μεταφέρεται στην επόμενη σελίδα.

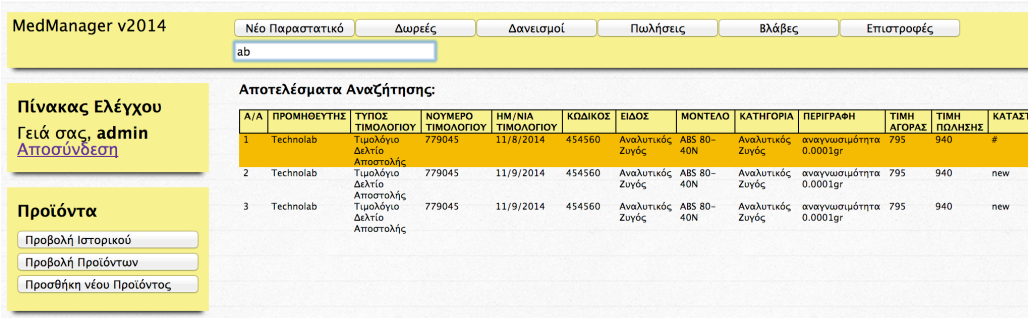


Στην σελίδα αυτή βλέπουμε το γνώριμο σε όλους μας παραστατικό. Το μόνο που έχουμε να κάνουμε είναι να συμπληρώσουμε όλα τα στοιχεία και να το προσθέσουμε.

Έπειτα μπορεί και να εκτυπωθεί. Βρίσκεται όμως αποθηκευμένο και στην βάση δεδομένων της εφαρμογής.

5.6 Αναζήτηση

Η αναζήτηση προϊόντων είναι άλλη μια αξιόλογη λειτουργία της εφαρμογής που αξίζει να σημειωθεί. Το χαρακτηριστικό της που ξεχωρίζει είναι το γεγονός ότι ο χρήστης πληκτρολογεί την λέξει κλειδί και τα αποτελέσματα φαίνονται μπροστά του σε πραγματικό χρόνο, χωρίς να επιλέγει τι είναι αυτό που αναζητά, ή να χρειάζεται να πατήσει κάποιο πλήκτρο για εκίνηση της αναζήτησης ή να μεταφερθεί σε άλλη σελίδα.



Στην παραπάνω εικόνα βλέπουμε στο πεδίο της αναζήτησης, ο χρήστης έχει πληκτρολογήσει μόλις δύο γράμματα. Η μηχανή όμως αναζήτησης έχει ήδη αναζητήσει όλα τα πεδία όλων των εγγραφών και έχει επιστρέψει το πιθανότερο αποτέλεσμα.

6. Βιβλιογραφεία

ΒΙΒΛΙΑ

Pollock John (2001), Οδηγός της JavaScript.

Julie Meloni (2004), Μάθετε PHP, MySQL και APACHE Όλα σε Ένα, Εκδόσεις Γκιούρδας

Welling Luke,Thomson Laura (2002), Ανάπτυξη Web Εφαρμογών με PHP και MySQL

L. Atkinson, Z. Suraski (2004), Πλήρης οδηγός της PHP 5, Εκδόσεις Γκιούρδας

ΚΕΙΜΕΝΟ ΣΕ ΔΙΚΤΥΑΚΟ ΤΟΠΟ

http://www.w3.org/MarkUp/

http://www.w3schools.com/html/default.asp

http://www.webreference.com/js/

http://javascript.internet.com/

http://javascript.com/

http://www.javascriptkit.com/

www.php.net http://www.phpbuilder.com/

http://www.mysql.com/

http://en.wikipedia.org/wiki/Model%E2%80%93View%E2%80%93Controller#cite_n ote-1

http://en.wikipedia.org/wiki/Model%E2%80%93View%E2%80%93Controller

www.pcmag.com/encycopedia , “Static and dynamic Web pages”

http://www.youtube.com/watch?v=gKg_QtlmZfc

https://www.youtube.com/watch?v=qXRcVhWxuaU