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

Πτυχιακή εργασία
«Σύστημα διαχείρισης ιστορικού ασθενών με χρήση PHP και MYSQL»

Των φοιτητών:

Δουλάκη Στυλιανού
Αρ. Μητρώου: 03/2188

Νταγιαμά Δημητρίου
Αρ. Μητρώου: 03/2190

Επιβλέπων καθηγητής:

Αθανάσιος Μάργαρης

Θεσσαλονίκη 2009

Πρόλογος

Η συγκεκριμένη εργασία δημιουργήθηκε στα πλαίσια πτυχιακής εργασίας κατά το ακαδημαϊκό έτος 2008 – 2009 για λογαριασμό του Αλεξάνδρειου Τεχνολογικού Εκπαιδευτικού Ιδρύματος Θεσσαλονίκης.

Περίληψη

Η εργασία αυτή είναι μια εφαρμογή για την καταγραφή του ιατρικού προφίλ ασθενών μίας κλινικής ή ενός νοσοκομείου. Σκοπός της εφαρμογής είναι μία πλήρως καταγεγραμμένη εικόνα ενός ασθενή για την καλύτερη αντιμετώπιση κάθε περιστατικού από το γιατρό. Επίσης κάθε διάγνωση καταγράφει τα στοιχεία του ιατρού που την έκανε για μία πιο υπεύθυνη αντιμετώπιση του ασθενούς.

Στο σύστημα υπάρχουν δύο κατηγορίες χρηστών, ο ιατρός και ο διαχειριστής. Οι επιλογές του χρήστη στο κύριο μενού της εφαρμογής είναι διαφορετικές για κάθε κατηγορία χρήστη.

Ο ιατρός μπορεί να δημιουργεί ασθενείς, να αλλάζει τα στοιχεία τους και να τους διαγράφει αν δεν έχουν καταχωρημένο ιστορικό. Μπορεί να καταχωρεί διάγνωση σε κάποιο ασθενή και να προβάλει τις καταχωρημένες διαγνώσεις. Του δίνεται η δυνατότητα να εισάγει δώδεκα διαφορετικές εξετάσεις και να βλέπει τα αποτελέσματα. Έχει το δικαίωμα αναζήτησης ασθενούς στη βάση δεδομένων του συστήματος και προβολή λίστας ασθενών. Τέλος του δίνεται η δυνατότητα να αλλάξει τον κωδικό πρόσβασης της εφαρμογής.

Ο διαχειριστής εκτός από τις δυνατότητες του γιατρού μπορεί να δημιουργεί, να αλλάζει και να διαγράφει γιατρούς, ταμεία ασφάλισης και ιατρικές ειδικότητες. Έχει το δικαίωμα διαγραφής της εξέτασης και διάγνωσης. Του δίνεται η δυνατότητα προβολής στατιστικών και διαχείρισης λογαριασμών χρηστών. Τέλος έχει το δικαίωμα αναζήτησης ιατρών και εκτέλεση κάποιας ενέργειας στη λίστα αποτελεσμάτων.

Summary

This particular project is the implementation of a medical record in regard to the patients of a clinic or hospital. the project aims to attain a complete and integral patients' profile that will help the doctor decide the appropriate remedy in each case. Additionally, each diagnosis itemizes the doctor's details, in each case, thus, resulting to a more reliable and responsible attitude from the part of the doctor.

In the system, one can find two categories of users, the doctor and the patient. The options on the main menu vary according to the user.

The doctor can create new patients' profiles, change their details or delete them if there are no recorded diagnoses. Also, the doctor can inscribe a diagnosis and bring out the pre-registered ones. he has the option to enregister twelve different diagnoses and discern the results. he has the right to search for a patient on the data base and and bring out the list of the patienta. In conclusion, he has the option to change the password of the application.

The administrator, in spite of the doctor's options, can create, change and delete, as well, doctors' profiles, insurance fund and doctor's speciality. also, he can delete the diagnosis. he has the option to set out statistics and manage accounts. Finally, he has the option to search for doctors and executions of actions on the list of the results.

Ευχαριστίες

Θα θέλαμε να ευχαριστήσουμε τη Μπουκουβούλα Σταυρούλα πτυχιούχο ιατρικής του Αριστοτελείου πανεπιστημίου Θεσσαλονίκης. Μας έδωσε αρκετές πληροφορίες σχετικά με τον τρόπο λειτουργίας ενός νοσοκομείου, τους ιατρούς, τις ιατρικές ειδικότητες, τα ταμεία ασφάλισης. Μας βοήθησε σχετικά με τις διαγνώσεις, τις εξετάσεις και τις κατηγορίες κάθε εξέτασης. Συγκέντρωσε πληροφορίες για τις εργαστηριακές εξετάσεις και για τους τύπους των

απεικονιστικών εξετάσεων. Γενικότερα η βοήθειά της ήταν καθοριστική για την ανάπτυξη του συστήματος

Επίσης ευχαριστούμε τον επιβλέποντα καθηγητή, Αθανάσιο Μάργαρη για την όλη υποστήριξη και βοήθειά του για την επιτυχή εκπόνηση της πτυχιακής μας εργασίας

Περιεχόμενα

Κεφάλαιο 1^{ο} : ΕΙΣΑΓΩΓΗ

1.1 Εισαγωγή

Η συγκεκριμένη εργασία δημιουργήθηκε στα πλαίσια πτυχιακής εργασίας κατά το ακαδημαϊκό έτος 2008 – 2009 για λογαριασμό του Αλεξάνδρειου Τεχνολογικού Εκπαιδευτικού Ιδρύματος Θεσσαλονίκης.

1.2 Τεχνολογίες

Για να δημιουργηθεί η εφαρμογή χρησιμοποιήθηκε ο συνδυασμός των εργαλείων phpMyAdmin – MySQL – Apache Server. Τα εργαλεία αυτά περιέχονται στο πακέτο Wamp Server που διανέμεται δωρεάν στο διαδίκτυο. Για την ανάπτυξη της ιστοσελίδας εκτός της PHP και HTML χρησιμοποιήθηκαν οι τεχνολογίες CSS και Javascript. Για την προβολή της ιστοσελίδας χρησιμοποιήθηκε το πρωτόκολλο HTTP και το url είναι μέσω Dyn DNS. Όλες αυτές οι τεχνολογίες αναλύονται διεξοδικά στο επόμενο κεφάλαιο.

1.3 Σκοπός

Σκοπός της εφαρμογής είναι η καταγραφή του ιστορικού ενός ασθενή. Το ιστορικό ενός ασθενή αποτελείται από τις διαγνώσεις που του έχουν γίνει αλλά και από τις εξετάσεις στις οποίες έχει υποβληθεί.

Με το σύστημα αυτό το ιστορικό μπορεί να διατηρηθεί για πάντα και έτσι μπορεί οποιοσδήποτε γιατρός να βγάλει ολοκληρωμένα συμπεράσματα σχετικά με την υγεία του ασθενούς. Με την χρήση αυτού του συστήματος δεν θα υπάρχουν πια χαμένες εξετάσεις και θα υπάρχει μια ολοκληρωμένη μηχανογράφηση σχετικά με την εικόνα ενός ασθενή.

Ο κάθε γιατρός μπορεί να δει αναλυτικά όλες τις εξετάσεις που έγιναν σε κάθε ασθενή είτε αυτές είναι εργαστηριακές (όπως η αιματολογική εξέταση) είτε είναι απεικονιστικές (όπως είναι η ακτινογραφία). Για κάθε διάγνωση το σύστημα κρατά τα στοιχεία του γιατρού ο οποίος έκανε την διάγνωση έτσι ώστε να αναγνωρίζονται

τα λάθη τους και να μην βγάζουν επιπόλαια συμπεράσματα. Κάθε γιατρος είναι σε θέση να δει τόσο τις εξετάσεις και διαγνώσεις που έχει κάνει ο ίδιος αλλά και άλλοι συνάδελφοί του.

1.4 Όροι λειτουργίας της εφαρμογής

1.4.1 Δυνατότητες της εφαρμογής

Το Σύστημα Διαχείρισης ασθενών μπορεί και παρέχει τις εξής λειτουργίες:

- Δημιουργία ασθενούς με όλα τα απαραίτητα στοιχεία που το περιγράφουν.
- Δημιουργία ιατρού με όλα τα απαραίτητα στοιχεία που το περιγράφουν.
- Δημιουργία ταμείου ασφάλισης για τον ασθενή.
- Δημιουργία ιατρικής ειδικότητας για τον ιατρό.
- Τροποποίηση στοιχείων ασθενούς.
- Τροποποίηση στοιχείων ιατρού.
- Αλλαγή του ονόματος του ταμείου ασφάλισης.
- Αλλαγή του ονόματος της ιατρικής ειδικότητας.
- Διαγραφή ασθενούς.
- Διαγραφή ιατρού.
- Διαγραφή ταμείου ασφάλισης.
- Διαγραφή ιατρικής ειδικότητας.
- Προβολή στοιχείων ιατρού.
- Προβολή στοιχείων ασθενή.
- Καταχώρηση διάγνωσης ασθενούς από ιατρό.
- Προβολή διαγνώσεων ασθενούς.
- Διαγραφή διαγνώσεων ασθενούς.
- Δημιουργία εργαστηριακής και απεικονιστικής εξέτασης και καταχώρησή της σε ασθενή
- Προβολή εξετάσεων ασθενούς.
- Διαγραφή εξετάσεων ασθενούς.
- Δημιουργία λογαριασμού ιατρού.
- Διαγραφή λογαριασμού ιατρού.
- Αλλαγή κωδικού πρόσβασης λογαριασμού από γιατρό.

- Αλλαγή κωδικού πρόσβασης λογαριασμού από διαχειριστή συστήματος.
- Απλή αναζήτηση ασθενή ή γιατρού με ένα κριτήριο προς αναζήτηση.
- Προηγμένη αναζήτηση ασθενή ή ιατρού με τουλάχιστον ένα και το πολύ τέσσερα κριτήρια αναζήτησης.

1.4.2 Τύποι χρηστών

Οι χρήστες της εφαρμογής είναι δύο. Ο ιατρός και διαχειριστής του συστήματος (administrator). Τα δικαιώματα που έχει κάθε χρήστης περιγράφονται αναλυτικά παρακάτω.

1.4.2.1 Ιατρός

Όλοι οι ιατροί που αναφέρονται στην εφαρμογή αποτελούν τους εγγεγραμμένους ιατρούς του νοσοκομείου ή της κλινικής. Όλοι οι γιατροί του νοσοκομείου προμηθεύονται ένα όνομα χρήστη κι ένα συνθηματικό κατά την εισαγωγή τους στο σύστημα. Για τη σωστή λειτουργία της εφαρμογής και την πλήρη μηχανογράφηση της πρέπει όλοι οι γιατροί του νοσοκομείου να υπάρχουν στο σύστημα.

Οι ιατροί έχουν δικαίωμα απλής και προηγμένης αναζήτησης ασθενούς. Μπορεί να αλλάξει τον κωδικό πρόσβασης στην εφαρμογή. Έχει δικαίωμα στο να δημιουργεί ασθενείς, να αλλάζει τα στοιχεία τους, να βλέπει τα στοιχεία τους και να τους διαγράφει.

Οι ιατροί μπορούν να κάνουν διάγνωση ενός ασθενούς και να προβάλουν τη διάγνωση. Δεν έχουν δικαίωμα να τη σβήσουν. Τα ίδια με τις διαγνώσεις ισχύουν και για τις εξετάσεις.

1.4.2.2 Διαχειριστής συστήματος

Ο διαχειριστής του συστήματος έχει όλα τα δικαιώματα που θα μπορούσε να έχει στην εφαρμογή. Είναι αυτός που δημιουργεί γιατρούς και τους δίνει όνομα χρήστη και κωδικό πρόσβασης. Μπορεί να βλέπει, να αλλάζει τα στοιχεία των

γιατρών καθώς επίσης και να τους διαγράφει. Έχει την δυνατότητα να δημιουργεί ασθενείς να τους επεξεργάζεται αλλά και να τους σβήνει.

Μπορεί και ο διαχειριστής να κάνει μια διάγνωση αλλά και να καταχωρήσει εξετάσεις σε κάποιον ασθενή. Η διαγραφή εξέτασης ή διάγνωσης είναι μόνο δικαίωμα του γιατρού. Επίσης ο διαχειριστής είναι υπεύθυνος για τα ασφαλιστικά ταμεία. Δημιουργεί τροποποιεί και διαγράφει ασφαλιστικά ταμεία από τη βάση δεδομένων. Αντίστοιχες λειτουργίες μπορεί να κάνει και για τις ιατρικές ειδικότητες.

Έχει τη δυνατότητα να κάνει απλή αναζήτηση ασθενούς ή γιατρού με σκοπό την προβολή, αλλαγή ή διαγραφή του στοιχείου προς αναζήτηση. Η απλή αναζήτηση είναι με βάση ενός κριτηρίου προς αναζήτηση. Υπάρχει ωστόσο και στη διάθεσή του και η προηγμένη αναζήτηση γιατρού ή ασθενή όπου τα κριτήρια για την αναζήτηση μπορεί να είναι από ένα ως τέσσερα.

Τέλος ο διαχειριστής της εφαρμογής είναι αυτός που είναι και υπεύθυνος για τους λογαριασμούς και την πρόσβαση των γιατρών στην εφαρμογή. Μπορεί να αλλάζει τον κωδικό πρόσβασης κάποιου ασθενή χωρίς να χρειάζεται τον παλιό κωδικό. Μπορεί επίσης να σταματά την πρόσβαση του γιατρού στην εφαρμογή απενεργοποιώντας τον λογαριασμό του.

1.5 Χρήσιμες Ιατρικές Έννοιες

1.5.1 Εξέταση

Με τον όρο εξέταση εννοούμε τα αποτελέσματα τα οποία προκύπτουν από κάποιον έλεγχο του ασθενή από τα εργαστήρια του νοσοκομείου ή από κάποιο ακτινολογικό τμήμα ή ακόμη και από κάποιον γιατρό. Αποτελούν έντυπα ή αποτελέσματα κάποιων μετρήσεων που έχουν γίνει στον ασθενή. Δεν αποτελούν την απάντηση για την πάθηση του ασθενή αλλά χρησιμοποιούνται για να γίνει η διάγνωση και να βγάλει ο γιατρός κάποια συμπεράσματα.

Οι εξετάσεις χωρίζονται σε δύο κατηγορίες, τις εργαστηριακές και τις απεικονιστικές. Οι εργαστηριακές εξετάσεις είναι αυτές που γίνονται στα

εργαστήρια με ανάλυση δείγματος από το σώμα του ασθενή και το αποτέλεσμα τους είναι κάποιες μετρήσεις. Απεικονιστικές είναι οι εξετάσεις πού γίνονται συνήθως σε κάποιο ακτινολογικό τμήμα και το αποτέλεσμα που προκύπτει είναι μια εικόνα ή κάποιο γράφημα.

Όπως είναι φυσικό η εφαρμογή αυτή αποτελεί πτυχιακή εργασία και περιέχονται μόνο οι βασικές εργαστηριακές και απεικονιστικές εξετάσεις. Για την απόφαση των ποιων εξετάσεων θα καταχωρούνται και ποιες τιμές πρέπει να παίρνουν συνεργαστήκαμε με φοιτήτρια της ιατρική σχολής του Αριστοτελείου πανεπιστημίου Θεσσαλονίκης η οποία έκανε πρακτική εξάσκηση στο Ιπποκράτειο νοσοκομείο Θεσσαλονίκης.

Οι εργαστηριακές εξετάσεις οι οποίες καταχωρούνται στο σύστημα είναι η αιματολογική εξέταση, ο βιοχημικός έλεγχος, ο ορμονικός έλεγχος, ο ορολογικός έλεγχος, η ουρολογική εξέταση και ο πηκτικός έλεγχος.

Οι απεικονιστικές εξετάσεις οι οποίες καταχωρούνται στο σύστημα σαν φωτογραφίες είναι η ακτινογραφία, ο υπέρηχος, η αξονική τομογραφία, η μαγνητική φωτογραφία, η αγγειογραφία και το σπινθηρογράφημα.

1.5.2 Διάγνωση

Με τον όρο διάγνωση εννοούμε το αποτέλεσμα που προκύπτει από την φυσική εξέταση του ασθενούς και έπειτα από τη μελέτη των εργαστηριακών και απεικονιστικών εξετάσεων. Αποτελεί την πάθηση ή τη νόσο από την οποία πάσχει ο ασθενής. Είναι το συμπέρασμα του γιατρού σχετικά με τον ασθενή.

1.6 Παραδοχές

Για την σωστή λειτουργία του συστήματος δεχόμαστε ότι ο διαχειριστής του συστήματος είναι και αυτός ιατρός. Δεν εννοούμε ότι το επάγγελμα του είναι ιατρός αλλά στο σύστημα είναι καταχωρημένος στον πίνακα doctor και έχει όλες τις πληροφορίες σχετικά με αυτόν σαν γιατρός. Αυτό γίνεται έτσι ώστε ο διαχειριστής να μπορεί να δημιουργήσει ασθενείς, να μπορεί να καταχωρήσει διαγνώσεις,

γενικότερα να μπορεί να εκτελέσει όλες τις ενέργειες που μπορεί να κάνει ένας γιατρός.

Κεφάλαιο 2^{ο} : Τεχνολογίες και Εργαλεία

2.1 PHP

2.1.1 Τι είναι η PHP

Η PHP είναι μια γλώσσα προγραμματισμού που σχεδιάστηκε για τη δημιουργία δυναμικών σελίδων στο διαδίκτυο και είναι επισήμως γνωστή ως: HyperText preprocessor.

Είναι μια server-side (εκτελείτε στον διακομιστή) scripting γλώσσα που γράφεται συνήθως πλαισιωμένη από HTML, για μορφοποίηση των αποτελεσμάτων. Αντίθετα από μια συνηθισμένη HTML σελίδα η σελίδα PHP δεν στέλνεται άμεσα σε έναν πελάτη (client), αλλά πρώτα αναλύεται και μετά αποστέλλεται το παραγόμενο αποτέλεσμα. Τα στοιχεία HTML στον πηγαίο κώδικα μένουν ως έχουν, αλλά ο PHP κώδικας ερμηνεύεται και εκτελείται. Ο κώδικας PHP μπορεί να θέσει ερωτήματα σε βάσεις δεδομένων, να δημιουργήσει εικόνες, να διαβάσει και να γράψει αρχεία, να συνδεθεί με απομακρυσμένους υπολογιστές , κ.ο.κ. Σε γενικές γραμμές οι δυνατότητες που μας δίνει είναι απεριόριστες.

Αρχικά η ονομασία της ήταν PHP/FI από το Forms Interpreter η οποία δημιουργήθηκε το 1995 από τον Rasmus Lerdorf ως μια συλλογή από Perl scripts που τα χρησιμοποιούσε στην προσωπική του σελίδα. Δεν άργησε να τα εμπλουτίσει με λειτουργίες επεξεργασίας δεδομένων με SQL, αλλά τα σημαντικά βήματα που έφεραν και την μεγάλη αποδοχή της PHP ήταν αρχικά η μετατροπή τους σε C και μετέπειτα η δωρεάν παροχή του πηγαίου κώδικα μέσω της σελίδας του ώστε να επωφεληθούν όλοι από αυτό που είχε φτιάξει, αλλά και να τον βοηθήσουν στην περαιτέρω ανάπτυξή της.

2.1.2 Ιστορία και εξέλιξη της PHP

Η ιστορία της PHP ξεκινά από το 1995, όταν ένας φοιτητής, ο Rasmus Lerdorf δημιούργησε χρησιμοποιώντας τη γλώσσα προγραμματισμού Perl ένα απλό script με όνομα php.cgi, για προσωπική χρήση.

Το script αυτό είχε σαν σκοπό να διατηρεί μια λίστα στατιστικών για τα άτομα που έβλεπαν το online βιογραφικό του σημείωμα. Αργότερα αυτό το script το διέθεσε και σε φίλους του, οι οποίοι άρχισαν να του ζητούν να προσθέσει περισσότερες δυνατότητες.

Η γλώσσα τότε ονομαζόταν PHP/FI από τα αρχικά Personal Home Page/Form Interpreter. Το 1997 η PHP/FI έφθασε στην έκδοση 2.0, βασιζόμενη αυτή τη φορά στη γλώσσα C και αριθμώντας περισσότερους από 50.000 ιστοτόπους που τη χρησιμοποιούσαν, ενώ αργότερα την ίδια χρονιά οι Andi Gutmans και Zeev Suraski ξαναέγραψαν τη γλώσσα από την αρχή, βασιζόμενοι όμως αρκετά στην PHP/FI 2.0. Έτσι η PHP έφθασε στην έκδοση 3.0 η οποία θύμιζε περισσότερο τη σημερινή μορφή της.

Στη συνέχεια, οι Zeev και Andi δημιούργησαν την εταιρεία Zend (από τα αρχικά των ονομάτων τους), η οποία συνεχίζει μέχρι και σήμερα την ανάπτυξη και εξέλιξη της γλώσσας PHP. Ακολούθησε το 1998 η έκδοση 4 της PHP, τον Ιούλιο του 2004 διατέθηκε η έκδοση 5, ενώ αυτή τη στιγμή έχουν ήδη διατεθεί και οι πρώτες δοκιμαστικές εκδόσεις της επερχόμενης PHP 6, για οποιονδήποτε προγραμματιστή θέλει να τη χρησιμοποιήσει. Οι περισσότεροι ιστοτόποι επί του παρόντος χρησιμοποιούν κυρίως τις εκδόσεις 4 και 5 της PHP.

2.1.2.1 Personal Home Page: Η πρώτη έκδοση της PHP

Αυτή η έκδοση της PHP αποτελούνταν από μια μηχανή parser που υποστήριζε κάποιες μακροεντολές και βοηθήματα που χρησιμοποιούνταν συχνά στις προσωπικές ιστοσελίδες. Επέτρεπε στους προγραμματιστές να ενσωματώνουν διαδικασίες όπως ένα βιβλίο επισκεπτών και ένα μετρητή αριθμού επισκέψεων στις ιστοσελίδες τους.

2.1.2.2 PHP/FI: Η δεύτερη έκδοση της PHP

Στα μέσα του 1995 ο Lerdorf έγραψε τον αναλυτή (parser) εκ νέου και τον μετονόμασε σε PHP/FI Version 2 (το FI συμβολίζει Form Interpreter). Μπορούσε δηλαδή να διερμηνεύει δεδομένα από μια HTML φόρμα. Εκτός από τις λειτουργίες FI και Personal Home Page Tools η PHP/FI περιελάμβανε υποστήριξη και για την βάση δεδομένων mSQL.

Εξαιτίας αυτών των ισχυρών χαρακτηριστικών της, και με την ενσωματωμένη υποστήριξη βάσης δεδομένων, έγινε γνωστή πολύ σύντομα και πολλοί άρχισαν να δουλεύουν στον κώδικα, προσπαθώντας να τη βελτιώσουν και να της προσθέσουν μεγαλύτερη λειτουργικότητα. Είχε τόση μεγάλη απήχηση που προγραμματιστές συνεισέφεραν τον κώδικά τους σε αυτήν. Μετά από την PHP/FI ήρθε η PHP 3, με πιο ισχυρές και εμπλουτισμένες δυνατότητες.

2.1.2.3 PHP 3: Η τρίτη έκδοση της PHP

Μέχρι τα μέσα του 1997 η χρήση της PHP ήταν περιορισμένη στους λίγους προγραμματιστές που είχαν συνεισφέρει τον κώδικά τους. Όμως με την αναγνώριση και την ιδιαίτερη δημοτικότητα που είχε αποκτήσει η εξέλιξή της σύντομα έγινε προσπάθεια δημιουργίας μιας μεγαλύτερης και πιο οργανωμένης ομάδας προγραμματιστών και χρηστών. Ο parser ξαναγράφτηκε από την αρχή.

Οι Zeev Suraski και Andi Gutmans, σχημάτισαν τον πυρήνα της τρίτης έκδοσης της PHP, γνωστός ως PHP 3 ο οποίος περιελάμβανε έναν αξιοσημείωτο αριθμό νέων χαρακτηριστικών. Μέσα σε όχι και πολύ χρόνο άρχισε να χρησιμοποιείται για τη δημιουργία εφαρμογών στο διαδίκτυο.

Ο συνδυασμός των PHP, Apache, MySQL, σύντομα έγινε ένας επιτυχημένος συνδυασμός για την ανάπτυξη εφαρμογών διαδικτύου. Η βελτιωμένη απόδοση του συστήματος αυτού και το γεγονός ότι και οι τρεις τεχνολογίες διανέμονται ελεύθερα στο διαδίκτυο έκαναν ιδανικό αυτόν τον συνδυασμό.

Αυτό φυσικά δεν σημαίνει ότι η PHP δεν είναι συμβατή με άλλο λογισμικό. Η υποστήριξη που παρέχει η PHP σε έναν τόσο μεγάλο αριθμό συστημάτων διαχείρισης βάσεων δεδομένων είναι ένα από τα δυνατότερα και σημαντικότατα χαρακτηριστικά της.

Η πλέον σύγχρονη έκδοση της PHP είναι η PHP 4. Η έκδοση αυτή παρουσιάστηκε σε ένα πιο σύνθετο περιβάλλον όπου προγραμματισμός στο διαδίκτυο δεν είναι μόνο η συγγραφή πολύπλοκων ιστοσελίδων. Οι εφαρμογές που δημιουργούνται πρέπει να είναι δυναμικές, αλληλεπιδραστικές και να υποστηρίζουν λειτουργίες σε επίπεδο βάσεων δεδομένων.

Οι εφαρμογές σχεδιάζονται με τέτοιο τρόπο ώστε πολλαπλοί χρήστες να μπορούν να στέλνουν και να λαμβάνουν πληροφορία συγχρόνως. Υπό μελέτη είναι και ο χρόνος απόκρισης για την ανάκτηση της πληροφορίας. Σύμφωνα με μια εκτίμηση ο μέγιστος χρόνος απόκρισης στον οποίο οι χρήστες μπορούν να περιμένουν υπομονετικά μια ιστοσελίδα να φορτώσει είναι 10 δευτερόλεπτα.

2.1.2.4 PHP 4: Η 4η έκδοση της PHP

Μερικοί από τους συνηθισμένους, εν τούτοις σημαντικούς, παράγοντες που προκάλεσαν αυτήν την εξέλιξη είναι οι εξής:

• Η PHP 4 είναι διαθέσιμη για ανάκτηση από την επίσημη ιστοσελίδα της (http://www.php.net).
• Ένας πολύ μεγάλος αριθμός εγγράφων που την υποστηρίζουν (εγχειρίδια, άρθρα) υπάρχουν ελεύθερα στο διαδίκτυο.
• Λόγω των πάρα πολλών χρηστών που έχει η PHP υπάρχει πληθώρα από newsgroups και mailing lists όπου μπορεί κάποιος να περιγράψει το πρόβλημά του και να βρει λύση.
• Υπάρχει μια αφοσιωμένη ομάδα από εθελοντές, που διορθώνουν οποιοδήποτε σφάλμα κώδικα έχει αναφερθεί η ανακαλυφθεί (http://www.php.net).
• Αντίθετα με τις άλλες γλώσσες σεναρίων, όπως η ASP(Active Server Pages), η PHP έχει ενσωματωμένο ερμηνευτή (interpreter) ο οποίος μεταφράζει τον κώδικα και μπορεί να ανιχνεύσει τυχόν λάθη που υπάρχουν.

• Ένα ακόμα σημαντικό χαρακτηριστικό της PHP είναι η μεταφερσιμότητα της (portability). Η PHP είναι συμβατή και μπορεί να συνεργαστεί με οποιονδήποτε συνδυασμό λογισμικού (λειτουργικό σύστημα – διακομιστές διαδικτύου – διακομιστές βάσεων δεδομένων).

Ακολουθούν ορισμένα χαρακτηριστικά που προστέθηκαν στην PHP 4.

• Υποστήριξη για Boolean τύπους δεδομένων.

• Υποστήριξη Java και XML.

• Υποστήριξη για FTP.

• Ο τελεστής «===», ο οποίος εκτός από τον κλασικό έλεγχο ισότητας δύο μεταβλητών, ελέγχει και αν έχουν τον ίδιο τύπο.

• Η ικανότητα της ανάκλησης μιας συνάρτησης (function)σε σημείο του προγράμματος όπου δεν έχει υλοποιηθεί ακόμα ο κώδικάς της.

• Υποστήριξη για κλήση διαδικασιών με αναφορά-call by reference (&) . Όπως είναι γνωστό αυτό βοηθά στην σύνδεση δύο μεταβλητών όπου η τιμή της μίας εξαρτάται άμεσα από την τιμή της άλλης και ενημερώνεται άμεσα η μία όταν αλλάξει η άλλη.

• SAPI (**S**erver **A**pplication **P**rogramming **I**nterface). Αυτό το χαρακτηριστικό εμπλουτίζει την υποστήριξη διακομιστών διαδικτύου(Web Servers).

• Υποστήριξη για πολλούς αλγορίθμους κρυπτογράφησης όπως οι DES, MD5, Blowfish kai SHA1. μέσω της βιβλιοθήκης mcrypt, η PHP 4 υποστηρίζει πλήρως την κρυπτογράφηση.

• Στην PHP 4, έχουν ξεπεραστεί όλοι οι περιορισμοί στη σύνταξη των εντολών που υπήρχαν σε παλαιότερες εκδόσεις.

• Οι μέθοδοι GET, POST υποστηρίζουν πλέον πίνακες που έχουν πάνω από μία διάσταση (δισδιάστατους, τρισδιάστατους).

• Το αρχείο αρχικής διαμόρφωσης (php.ini) είναι πολύ πιο κατανοητό και διαμορφώσιμο.

• Στην PHP 4 υποστηρίζεται καλύτερα η δημιουργία κλάσεων και αντικειμένων.

2.1.2.5 PHP5: Η τελευταία έκδοση της PHP

Η πλέον σύγχρονη έκδοση της PHP, έχει πολλά νέα χαρακτηριστικά που επιτρέπουν τις υψηλές επιδόσεις και την προσπέλαση σε ακόμα μεγαλύτερο

αριθμό επεκτάσεων (extensions) και βιβλιοθηκών (libraries). Η PHP αποτελεί πλέον το πρότυπο για τη δημιουργία εφαρμογών διαδικτύου και από έρευνες που έχουν γίνει διαπιστώθηκε ότι χρησιμοποιείται κατά κόρον από πάρα πολλούς χρήστες.

Σε αυτή την έκδοση της PHP έχει προστεθεί η δεύτερη έκδοση της μηχανής ZEND, έχει βελτιστοποιηθεί ο τρόπος με τον οποίο γίνεται η εκτέλεση τον εντολών και τέλος έχουν προστεθεί επιπλέον μηχανισμοί ώστε να αυξηθεί η ασφάλεια της γλώσσας.

Η τελευταία έκδοση η οποία κυκλοφορεί αυτή την στιγμή είναι η έκδοση 5.3.0.

2.1.3 Το πρώτο μας PHP script

Μιας και αυτό θα είναι το πρώτο PHP script που θα γράψουμε το πιο φρόνιμο θα είναι να πούμε ένα "Γεια" σε όλο τον κόσμο. Δημιουργούμε ένα νέο text αρχείο με το notepad και το ονομάζουμε first.php. Μέσα σε αυτό το αρχείο γράφουμε:

Κώδικας:

```
<?PHP
  echo "Hello World!";
?>
```

Το σώζουμε και το τοποθετούμε στον web server μας. Το συγκεκριμένο script θα πρέπει να μας εμφανίσει στον browser μας την φράση Hello World!. Με την ετικέτα <?PHP λέμε στον διακομιστή μας ότι από εκεί αρχίζει το PHP script μας και ότι υπάρχει μετά από αυτό θα πρέπει να περάσει από τον μεταφραστή της PHP πριν στείλει κάτι στον browser του πελάτη. Η μετάφραση σταματάει έως να βρει την ετικέτα τέλους κώδικα η οποία είναι η ?>

Ανάμεσα τοποθετήσαμε την εντολή echo "Hello World!"; όπου το μόνο που λέει στον μεταφραστή είναι να εμφανίσει στην έξοδο την φράση Hello World! Οι κλασικές ετικέτες αρχής και τέλους ενός PHP κώδικα είναι, όπως είπαμε, η <?PHP

και η ?>. Υπάρχει όμως και η σύντμηση αυτών η οποία συντάσσετε ως <? Για την αρχής, ενώ η ετικέτα τέλους μένει ως έχει ?> . Σε αυτήν την περίπτωση δηλαδή το script μας θα ήταν ως εξής.

Κώδικας:

```
<?PHP
  echo "Hello World!";
?>
```

Για το αν θα γίνονται δεκτές οι συντμήσεις των ετικετών ευθύνεται μια παράμετρος στο php.ini αρχείο μας: short_open_tag = On; Αν είναι στο On τότε έχουμε ενεργοποιημένες τις συντμήσεις. Μια άλλη παράμετρος που έχουμε στο php.ini μας για τις ετικέτες αρχής και τέλους του PHP κώδικα είναι και η asp_tags = On; Όπου εάν την έχουμε στο On η PHP καταλαβαίνει και τις ετικέτες τύπου ASP. Δηλαδή στην περίπτωσή μας θα έχουμε:

Κώδικας:

```
<%
  echo "Hello World!";
%>
```

Ένας ακόμα τρόπος για να πούμε ότι ο κώδικας που ακολουθεί είναι PHP είναι η χρησιμοποίηση της ετικέτας SCRIPT ως εξής:

Κώδικας:

```
<SCRIPT
LANGUAGE="php">
echo "Hello World!";
</SCRIPT>
```

Απ' ότι βλέπετε έχουμε πολλούς τρόπους για να περιγράψουμε στον διακομιστή μας ότι ο κώδικας που ακολουθεί θα πρέπει να επεξεργαστεί από τον μεταφραστή της PHP. Εμείς θα χρησιμοποιούμε το κλασικό <?PHP και ?> Για να

βάλουμε σχόλια στον κώδικά μας μπορούμε να βάλουμε // ή # ή /* και */ για πολλές γραμμές σχολίων.

Κώδικας:

```
// αυτό είναι ένα σχόλιο
# αυτό είναι άλλο ένα σχόλιο
/*
εδώ θα
βάλουμε
πάνω από μια γραμμή
με σχόλια
*/
```

Στο πρώτο βοήθημα της σειράς είχαμε πει ότι ο PHP κώδικας περικλείετε από HTML. Ας δούμε πως μπορεί να γίνει το first.php εμπλουτισμένο με HTML.

Κώδικας:

```
<html>
<head>
<title>My First PHP script</title>
</head>
<body>
<i>
<?PHP
   echo "Hello World!";
?>
</i>
</body>
</html>
```

Έτσι θα έχουμε ένα σωστά δομημένο HTML και η φράση Hello World! θα εμφανίζετε με πλαγιαστή γραφή. το τελικό παραγόμενο HTML αρχείο θα είναι ως εξής:

Κώδικας:

```
<html>
<head>
<title>My First PHP script</title>
</head>
<body>
<i>
Hello World!
</i>
</body>
</html>
```

2.1.4 Τύποι Δεδομένων στην PHP

Ας δούμε τώρα ποιους τύπους δεδομένων έχουμε στην PHP.

Τέσσερις βαθμωτούς τύπους:

- boolean του Μπουλ (True/False)
- integer ακέραιοι αριθμοί
- floating-point number (float) - (double) αριθμοί κινητής υποδιαστολής
- string αλφαριθμητικά

Δύο σύνθετους τύπους:

- array πίνακες
- object αντικείμενα

Και δύο ειδικούς τύπους

- resource τιμές αναφοράς με εξωτερικές πηγές
- NULL ειδική μεταβλητή που χαρακτηρίζει το 'τίποτα' ή το 'μη ορισμένο'.

Με την συνάρτηση gettype() η PHP μας λέει τι τύπου είναι η μεταβλητή που έχουμε

Κώδικας:

```
<?php
	$v = 5/2;
	$v1 = -3;
	$v2 = "Testing...";
	$v3 = True;

	echo $v."<br />";
	echo $v1."<br />";
	echo $v2."<br />";
	echo $v3."<br />";
	echo "<br />";
	echo gettype($v)."<br />";
	echo gettype($v1)."<br />";
	echo gettype($v2)."<br />";
	echo gettype($v3);
?>
```

Αυτό θα μας βγάλει σαν αποτέλεσμα:

```
2,5
-3
Testing...
1

Double
Integer
String
Boolean
```

Βλέπουμε ότι το True στο boolean μας το εξαφανίζει ως 1, αντίστοιχα στην τιμή False θα μας εμφάνιζε 0 και όχι κάποιο από τα λεκτικά True/False.

2.1.5 Μεταβλητές (Variables)

Η μεταβλητή είναι μια αναφορά σε μια θέση μνήμης του υπολογιστή μας, που κατά την διάρκεια εκτέλεσης ενός προγράμματος μπορούν να αποθηκευτούν πληροφορίες. Αυτές οι πληροφορίες μπορεί να αλλάζουν ή να λάβουν μέρος σε πράξεις μαζί με άλλες μεταβλητές ή σταθερές τιμές. Κάθε μεταβλητή έχει ένα συγκριμένο όνομα που την χαρακτηρίζει, μια συγκεκριμένη εμβέλειας και μια συγκεκριμένη διάρκεια ζωής.

Στις νεότερες γλώσσες προγραμματισμού, όπως και στην PHP, δε μας ενδιαφέρει σε ποια ακριβός θέση μνήμης έχουν τοποθετηθεί οι πληροφορίες της μεταβλητής μας. Αυτό που μας ενδιαφέρει είναι ότι μόνο με το όνομά της μπορούμε να προσπελάσουμε αυτήν την πληροφορία και να την αξιοποιήσουμε όπως εμείς θέλουμε.

Τα ιδιαίτερα χαρακτηριστικά των μεταβλητών (variables) στην PHP Στην PHP τις μεταβλητές δεν χρειάζεται να τις ορίσουμε πριν τις χρησιμοποιήσουμε και μπορούμε κατά την εκτέλεση να αλλάξουμε τον τύπο δεδομένων τους κάτω από συγκεκριμένους όρους.

Οι μεταβλητές της PHP αντιπροσωπεύονται από ένα σημάδι δολαρίου που ακολουθείται από το όνομα της μεταβλητής. Τα ονόματα των μεταβλητών είναι ευαίσθητα στα κεφαλαία και μικρά (case-sensitive).

Ένα έγκυρο όνομα μεταβλητής αρχίζει με ένα γράμμα ή κάτω παύλα (underscore) ακολουθούμενο από οποιονδήποτε αριθμό ή γράμμα, ή γράμματα, αριθμούς, και κάτω παύλες.

Οι αναθέσεις τιμών σε μια μεταβλητή γίνετε με το όνομά της ακολουθούμενο από το σύμβολο της ανάθεσης που είναι το = και αμέσως μετά την πληροφορία που θέλουμε να τοποθετήσουμε στην θέση μνήμης που αναφέρει.

Σωστά ονόματα μεταβλητών στην PHP με αναθέσεις τιμών

Κώδικας:

```
$var = "my first varriable";
$Var = "my 2nd variable"; //(αυτή είναι διαφορετική από την προηγούμενη γιατί έχει κεφαλαίο V)
$_EnyVar = "an other variable";
$iNum = 12;
$dNum = 15.6;
```

Λάθος ονόματα μεταβλητών στην PHP:

Κώδικας:

```
$2num = 2; //ξεκινάει με αριθμό
Jbl = 3; // δεν έχουμε βάλει το αρχικό $
```

Οι αναθέσεις με αναφορά (reference) γίνονται με το σύμβολο &. Με λίγα λόγια με αυτόν τον τρόπο καταφέρνουμε να συνδέσουμε δύο μεταβλητές και να έχουμε πρόσβαση στην ίδια πληροφορία με διαφορετικά ονόματα μεταβλητών.

Κώδικας:

```
<?php
$foo = 'Bob';              // βάζουμε την τιμή 'Bob' στο $foo
$bar = &$foo;              // Συνδέουμε το $foo μέσω του $bar.
$bar = "My name is $bar";  // αλλάζουμε το $bar...
echo $bar;
echo $foo;                 // βλέπουμε ότι και το $foo άλλαξε...
?>
```

2.1.5.1 Εμβέλεια και Διάρκεια ζωής μιας μεταβλητής

Διάρκεια Ζωής μιας μεταβλητής

Η Διάρκεια Ζωής μιας μεταβλητής ξεκινάει από την στιγμή που παίρνει για πρώτη φορά τιμή (δηλώνετε) και σταματάει να υφίσταται όταν ολοκληρωθεί η διαδικασία μέσα στην οποία δηλώθηκε.

Εμβέλεια μιας μεταβλητής

Η Εμβέλεια μιας μεταβλητής ορίζετε από την εμβέλεια της διαδικασίας μέσα στην οποία έχει ορισθεί. Εάν για παράδειγμα έχει ορισθεί μέσα σε μια procedure ή μέσα σε μια function η εμβέλειά της είναι μόνο μέσα εκεί.

Εάν έχει ορισθεί στον κυρίως κορμό του προγράμματος η εμβέλειά της και πάλι είναι μόνο μέσα εκεί και στα πιθανά κομμάτια τους κορμού από εξωτερικά αρχεία, αλλά όχι και μέσα στις περιεχόμενες procedures ή functions. Ονομάζετε

όμως καθολική (global), για τον λόγω ότι έχουμε τρόπο να την χρησιμοποιήσουμε όπου θέλουμε.

Κώδικας:

```
<?php
$num = 3;
include "config.inc";
?>
```

Η μεταβλητή $num θα υφίσταται και μέσα στο config.inc. Αλλά τι γίνετε όταν έχουνε κάποια function που την έχουμε ορίσει εμείς;

Κώδικας:

```
<?php
$num = 3; /* εμβέλεια στον κορμό του προγράμματος */

function Test()
{
  echo $num; /* εμβέλεια μόνο μέσα στην function */
}

Test();
?>
```

Το παραπάνω παράδειγμα δε θα μας βγάλει τίποτα, μιας και μέσα στην function δεν έχουμε δώσει τιμή στην μεταβλητή $num. Για να δούμε ακόμα ένα παράδειγμα που θα μας φέρει κάποια αποτελέσματα.

Κώδικας:

```
<?php
$num = 3; /* εμβέλεια στον κορμό του προγράμματος */

function Test()
{
  $num = 8; /* εμβέλεια μόνο μέσα στην function */
   echo $num;
}
echo $num;
Test();
echo $num;
?>
```

Με αυτό θα πάρουμε το αποτέλεσμα 383 και αυτό γιατί το $num του κορμού δεν επηρεάζετε από $num της function, το οποίο έχει εμβέλεια και διάρκεια ζωής όσο διαρκεί και η εκτέλεση της function.

Εάν θέλουμε η function να χρησιμοποιήσει την μεταβλητή του κορμού δεν έχουμε παρά να της πούμε ότι αυτή η μεταβλητή είναι καθολική (global) και δεν είναι τοπικής εμβέλειας.

Κώδικας:

```
<?php
$num = 3; /* εμβέλεια στον κορμό του προγράμματος */

function Test()
{
  global $num; /* του λέμε να χρησιμοποιήσει την μεταβλητή του
κορμού */
  $num = 8;
  echo $num;
}
echo $num;
Test();
echo $num;
?>
```

Με αυτόν τον τρόπο το αποτέλεσμα που θα πάρουμε θα είναι το 388, μιας και μέσα στην function με το *global $num;* είπαμε να μη δημιουργήσει μια νέα θέση μνήμης για μια τοπική μεταβλητή, αλλά να χρησιμοποιήσει την ίδια την μεταβλητή

του κορμού. Γι' αυτό και ότι πράξεις κάνουμε μέσα στην function με αυτήν την μεταβλητή θα έχει αντίκτυπο και έξω από αυτήν.

Ένας άλλος τρόπος να κάνουμε το ίδιο πράγμα είναι και ο εξής:

Κώδικας:

```
<?php
$num = 3; /* εμβέλεια στον κορμό του προγράμματος */

function Test()
{
  $GLOBALS['num'] = 8; /* του λέμε να χρησιμοποιήσει την μεταβλητή
του κορμού */
  echo $GLOBALS['num'];
}
echo $num;
Test();
echo $num;
?>
```

Το $GLOBALS είναι ένας πίνακας που περιέχει όλες τις μεταβλητές του κορμού και η εμβέλειά του είναι παντού (superglobal).

Όλες οι μεταβλητές που η εμβέλειά τους είναι παντού (superglobals) και δεν χρειάζεται να ορισθούν ως καθολικές (global) σε ένα υποπρόγραμμα (function ή procedure):

$GLOBALS
$_SERVER
$_GET
$_POST
$_COOKIE
$_FILES
$_ENV
$_REQUEST
$_SESSION .

2.1.5.2 Ο Τύπος Δεδομένων μιας Μεταβλητής

Είδαμε στα προηγούμενα βοηθήματα τους Τύπους Δεδομένων (data types) και τις Μεταβλητές (variables). Ακόμα είπαμε ότι μια μεταβλητή στην PHP μπορεί να έχει αρχικά έναν τύπο δεδομένων, αλλά στην διάρκεια εκτέλεσης του κώδικα μπορεί να αλλάξει. Αυτό που θα δούμε εδώ είναι πως μπορούμε να αναγνωρίσουμε τι τύπο δεδομένων κουβαλάει μια μεταβλητή και πως μπορούμε να τον αλλάξουμε.

Αναγνωρίζοντας τον τύπο δεδομένων με την gettype( )

Η PHP για να μας βοηθήσει στην αναγνώριση του τύπου δεδομένων έχει ενσωματώσει την function gettype( ) που κάνει αυτήν ακριβώς την δουλειά. Παίρνει ως παράμετρο την μεταβλητή και επιστρέφει λεκτικά τον τύπο των δεδομένων που υπάρχουν μέσα της.

Τα λεκτικά που μπορεί να επιστρέψει είναι τα εξής:

- integer
- string
- double
- boolean
- array
- object
- resource
- NULL
- unknown type

Ας δούμε κάποια από αυτά με το παρακάτω παράδειγμα.

Κώδικας:

```
<html>
<head>
<title>Τι τύπο δεδομένων έχει η κάθε μεταβλητή;</title>
</head>
<body>
<?PHP
  $var = 5;
  echo gettype( $var ); // integer
?>
<br>
<?PHP
  $var = "κείμενο";
  echo gettype( $var ); // string
?>
<br>
<?PHP
  $var = 5.0;
  echo gettype( $var ); // double
?>
<br>
<?PHP
  $var = true;
  echo gettype( $var ); // boolean
?>
</body>
</html>
```

Στο παραπάνω παράδειγμα είδαμε ακόμα ότι έχοντας βασικά πάντα την ίδια μεταβλητή μπορέσαμε κατά την διάρκεια εκτέλεσης να τοποθετήσουμε μέσα της όλους τους βαθμωτούς (scalar) τύπους δεδομένων που υπάρχουν στην PHP.

2.2 MySQL και Σχεσιακό μοντέλο Δεδομένων

2.2.1 Το σχεσιακό μοντέλο δεδομένων

Ένα σύστημα διαχείρισης βάσης δεδομένων (ΣΔΒΔ) (*database management system (DBMS)*) αποτελείται από ένα σύνολο δεδομένων και προγράμματα πρόσβασης στα δεδομένα αυτά. Το σύνολο των δεδομένων καλείται βάση δεδομένων (*database*). Στόχος του ΣΔΒΔ είναι η εύκολη και γρήγορη χρήση και ανάκτηση των δεδομένων. Η διαχείριση των δεδομένων περιλαμβάνει:

- τον ορισμό δομών για τη αποθήκευση των δεδομένων
- τον ορισμό μεθόδων για τη διαχείριση των δεδομένων

Ο ορισμός της δομής της βάσης δεδομένων βασίζεται σε ένα μοντέλο δεδομένων το οποίο ορίζει τον τρόπο που περιγράφονται τα δεδομένα, οι σχέσεις τους, η σημασία τους και οι περιορισμοί πάνω στα δεδομένα αυτά.

Το σχεσιακό μοντέλο (*relational model*) δεδομένων παριστάνει δεδομένα και τις σχέσεις τους ως ένα σύνολο πινάκων. Κάθε πίνακας (*table*) αποτελείται από στήλες (*columns*) με μοναδικά ονόματα. Μια γραμμή (*row*) του πίνακα παριστάνει μια σχέση (*relationship*) ανάμεσα σε ένα σύνολο από τιμές. Ο πίνακας που ακολουθεί παριστάνει έναν τηλεφωνικό κατάλογο. Αποτελείται από δύο στήλες και πέντε γραμμές.

Όνομα	Τηλέφωνο
Γιώργος	32560
Μαρία	61359
Θανάσης	98756
Λίνα	78999
Πέτρος	12356

Η SQL (structured query language) αποτελεί σήμερα την πιο διαδεδομένη γλώσσα διαχείρισης σχεσιακών βάσεων δεδομένων. Η SQL παρέχει δυνατότητες για:

- τον ορισμό, τη διαγραφή και τη μεταβολή πινάκων και κλειδιών,
- τη σύνταξη ερωτήσεων (*queries*),
- την εισαγωγή, διαγραφή και μεταβολή στοιχείων,
- τον ορισμό όψεων (*views*) πάνω στα δεδομένα,
- τον ορισμό δικαιωμάτων πρόσβασης,
- τον έλεγχο της ακεραιότητας των στοιχείων,
- τον έλεγχο συναλλαγών (*transaction*)

Η SQL είναι ορισμένη ως διεθνές πρότυπο.

Στην περιγραφή της σύνταξης της SQL θα χρησιμοποιήσουμε τα παρακάτω σύμβολα:

[έκφραση]
η έκφραση εμφανίζεται προαιρετικά

έκφραση1 | έκφραση2
μπορεί να γραφεί η έκφραση1 ή η έκφραση2

έκφραση ...
η έκφραση μπορεί να επαναληφθεί

2.2.2 Τύποι δεδομένων

Τα δεδομένα κάθε στήλης ενός πίνακα πρέπει να έχουν ένα συγκεκριμένο τύπο. Οι βασικοί τύποι που υποστηρίζονται από την SQL είναι οι παρακάτω:

BIT
Ναι ή Όχι

CURRENCY
Τιμή που παριστάνει με ακρίβεια αριθμούς από -922.337.203.685.477,5808 έως 922.337.203.685.477,5807

DATETIME
Χρόνος

SINGLE
Αριθμός κινητής υποδιαστολή μονής ακρίβειας

DOUBLE

Αριθμός κινητής υποδιαστολή διπλής ακρίβειας

SHORT
Ακέραιος 2 byte (-32768 έως 32767)

LONG
Ακέραιος 4 byte (-2.147.483.648 έως 2.147.483.647)

TEXT
Κείμενο μέχρι 255 χαρακτήρες

LONGTEXT
Κείμενο μέχρι 1.2GB

2.2.3 Δημιουργία πινάκων

Νέοι πίνακες δημιουργούνται με την εντολή CREATE TABLE. Αυτή συντάσσεται ως εξής:

CREATE TABLE όνομα_πίνακα (πεδίο1 τύπος [(μέγεθος)] [, πεδίο2 τύπος [(μέγεθος)] [, ...]]

Για παράδειγμα η εντολή:

```
CREATE TABLE Customer
(Name TEXT (20), AccountNum SHORT)
```

δημιουργεί τον πίνακα με όνομα Customer και με δύο στήλες: την Name και την AccountNum.

2.2.4 Αλλαγές σε πίνακες

Η εντολή ALTER TABLE επιτρέπει την προσθήκη νέων στηλών ή τη διαγραφή υπαρχόντων. Η προσθήκη νέων στηλών γίνεται με τη σύνταξη:

ALTER TABLE όνομα_πίνακα ADD COLUMN πεδίο τύπος[(μέγεθος)]

Για παράδειγμα η εντολή:

```
ALTER TABLE Customer
ADD COLUMN Notes TEXT(25)
```

προσθέτει μια νέα στήλη στον πίνακα Customer.

Η διαγραφή υπαρχόντων στηλών γίνεται με τη σύνταξη:

ALTER TABLE όνομα_πίνακα DROP COLUMN πεδίο

Για παράδειγμα η εντολή:

```
ALTER TABLE Customer
DROP COLUMN Notes
```

αφαιρεί τη στήλη Notes από τον πίνακα Customer.

Τέλος, η εντολή DROP TABLE επιτρέπει τη διαγραφή πινάκων.

Για παράδειγμα η εντολή :

```
DROP TABLE Customer
```

θα διαγράψει τον πίνακα Customer

2.2.5 Δείκτες

Η πρόσβαση στα στοιχεία ενός πίνακα γίνεται ταχύτερα όταν αυτά οργανωθούν με τη βοήθεια δεικτών. Ένας δείκτης ορίζεται για μια συγκεκριμένη στήλη ή στήλες και επιτρέπει τη γρήγορη πρόσβαση σε γραμμές με βάση τιμές της

συγκεκριμένης στήλης. Ουσιαστικά όταν ορίζουμε έναν δείκτη το ΣΔΒΔ υλοποιεί μια δομή δεδομένων (π.χ. ταξινομημένο ή κατακερματισμένο πίνακα ή δένδρο) για γρήγορη πρόσβαση στα αντίστοιχα δεδομένα. Δείκτες δημιουργούνται με την εντολή CREATE INDEX. Η σύνταξή της είναι η παρακάτω:

CREATE [UNIQUE] INDEX όνομα_δείκτη
ON όνομα_πίνακα (πεδίο [ASC|DESC][, πεδίο [ASC|DESC], ...])

Η λέξη UNIQUE ορίζει πως δε θα επιτρέπονται διπλές εμφανίσεις μιας τιμής για το συγκεκριμένο δείκτη. Οι λέξεις ASC και DESC ορίζουν αύξουσα ή φθίνουσα ταξινόμηση του πίνακα με βάση το δείκτη.

Παράδειγμα:

```
CREATE INDEX NameIdx
ON Customer (Name)
```

Δημιουργεί το δείκτη NameIdx για τη στήλη Name στον πίνακα Customer. Ένας δείκτης μπορεί να διαγραφεί με τη σύνταξη:

DROP INDEX όνομα_δείκτη ON όνομα_πίνακα

2.2.6 Προσθήκη στοιχείων

Προσθήκη δεδομένων σε έναν πίνακα γίνεται με την εντολή INSERT INTO σύμφωνα με τη σύνταξη:

INSERT INTO όνομα_πίνακα [(πεδίο1[, πεδίο2[, ...]])]
VALUES (τιμή1[, τιμή2[, ...])

Παράδειγμα:

```
INSERT INTO Customer (Name, AccountNum)
VALUES
("Papadopoulos", 1234)
```

2.2.7 Επιλογή SELECT

Επιλογή δεδομένων από έναν ή περισσότερους πίνακες γίνεται με την εντολή SELECT σύμφωνα με την παρακάτω βασική σύνταξη:

SELECT πεδία
FROM πίνακες
[WHERE κριτήρια]

Απλό παράδειγμα:

```
SELECT Name, AccountNum

FROM Customer

WHERE Name LIKE "Pap*"
```

Τα πεδία μπορούν να είναι ονόματα πεδίων ή συγκεντρωτικές συναρτήσεις της SQL πάνω σε πεδία. Τέτοιες συναρτήσεις είναι οι παρακάτω:

Avg
Μέσος όρος
Count
Μέτρηση
Min
Ελάχιστο
Max
Μέγιστο
Sum

Σύνολο

Παράδειγμα:

```
SELECT Sum(AccountBalance)
FROM Customer
```

Ο αστερίσκος ως ορισμός πεδίου επιλέγει όλα τα πεδία. Τα κριτήρια αναζήτησης είναι εκφράσεις πάνω στα πεδία. Ορισμένοι βασικοί τελεστές είναι οι:

- αριθμητικοί + - * / mod
- σύγκρισης < <= > >> = <> like
- λογικοί and or not

Παράδειγμα:

```
SELECT *
FROM Customer
WHERE Balance > 10000 AND Name LIKE "Papad*"
```

Βασικό στοιχείο του σχεσιακού μοντέλου είναι η αποθήκευση Βασικό στοιχείο του σχεσιακού μοντέλου είναι η αποθήκευση δεδομένων σε πίνακες που σχετίζονται μεταξύ τους. Έτσι για παράδειγμα ένας πίνακας μπορεί να φυλάει τα στοιχεία των πελατών και ένας άλλος τους λογαριασμούς. Με τον τρόπο αυτό τα στοιχεία ενός πελάτη που έχει πολλούς λογαριασμούς σε μια τράπεζα φυλάσσονται μόνο μια φορά. Ανάκτηση από δύο τέτοιους πίνακες γίνεται με τον προσδιορισμό τους στο SELECT μαζί με τη χρήση της κατάλληλης έκφρασης που θα ενώσει τους πίνακες.

Παράδειγμα:

```
SELECT Customer, Balance
FROM Customers, Accounts
WHERE Customer.AccountId = Accounts.AccountId
```

2.2.8 Αλλαγή

Αλλαγή σε στοιχεία γίνεται με την εντολή UPDATE σύμφωνα με τη σύνταξη:

```
UPDATE όνομα_πίνακα
SET πεδίο = νέα_τιμή
WHERE κριτήρια;
Παράδειγμα:
UPDATE Accounts
SET Balance=100000
WHERE AccountId = 12233
```

2.2.9 Διαγραφή

Διαγραφή γραμμών από έναν πίνακα γίνεται με την εντολή DELETE σύμφωνα με τη σύνταξη:

```
DELETE
FROM όνομα_πίνακα
WHERE κριτήρια
Παράδειγμα:
DELETE
FROM Accounts
WHERE
AccountId = 123232
```

2.2.10 Σύνδεση με το Διακομιστή Βάσης Δεδομένων

Πριν ξεκινήσουμε να δουλεύουμε με τη βάση δεδομένων μας, θα πρέπει πρώτα να συνδεθούμε με τον διακομιστή. Για το λόγο αυτό η PHP παρέχει τη ρουτίνα mysql_connect(). Η mysql_connect() δεν απαιτεί κανένα όρισμα αλλά δέχεται τρία strings : το hostname, ένα username και ένα password. Εάν δεν δώσουμε κανένα από αυτά τα ορίσματα, η ρουτίνα υποθέτει ότι ο host είναι localhost και ότι τα username και password δεν έχουν οριστεί στο mysqluser πίνακα, εκτός αν έχουν οριστεί στο αρχείο php.ini. Η mysql_connect() επιστρέφει έναν αναγνωριστή συνδέσμου (link identifier) αν η σύνδεση είναι επιτυχής. Την επιστρεφόμενη τιμή μπορούμε να την αποθηκεύσουμε σε μια μεταβλητή ώστε να μπορούμε να συνεχίσουμε να δουλεύουμε με τον διακομιστή βάσης δεδομένων.

Το επόμενο κομμάτι κώδικα χρησιμοποιεί τη mysql_connect() για να συνδεθεί με τον διακομιστή βάσης δεδομένων MySQL :

```
$link = mysql_connect( "localhost" );
if ( ! $link ) die( "Couldn't connect to MySQL" );
```

Στην περίπτωση που χρησιμοποιούμε την PHP σε συνδυασμό με τον Apache, μπορούμε να συνδεθούμε επίσης με τον διακομιστή βάσης δεδομένων με την mysql_pconnect(). Αυτή η ρουτίνα λειτουργεί όπως και η mysql_connect() με τη διαφορά ότι η σύνδεση δεν τερματίζεται όταν το script σταματάει να εκτελείται ή όταν καλούμε τη mysql_close() (η οποία τερματίζει μια συγκεκριμένη σύνδεση με το διακομιστή βάσης δεδομένων). Αντιθέτως, η σύνδεση παραμένει ενεργή, περιμένοντας μια άλλη διεργασία να καλέσει την mysql_pconnect().

2.2.11 Επιλογή μιας Βάσης Δεδομένων

Τώρα που έχουμε συνδεθεί με το διακομιστή βάσης δεδομένων θα πρέπει να επιλέξουμε τη βάση δεδομένων με την οποία θέλουμε να δουλέψουμε. Τη

δυνατότητα αυτή μας τη δίνει η mysql_select_db(). Η συγκεκριμένη μέθοδος απαιτεί το όνομα μιας βάσης δεδομένων και προαιρετικά έναν αναγνωριστή συνδέσμου. Η mysql_select_db() επιστρέφει true αν η βάση δεδομένων υπάρχει και μπορούμε να την προσπελάσουμε.

Στο επόμενο κομμάτι κώδικα διαλέγουμε μια βάση δεδομένων με όνομα DBname :

```
$database = "DBname";
mysql_select_db( $DBname )  or die ( "Couldn't open $DBname" );
```

2.2.12 Διαχείριση Λαθών

Μέχρι τώρα ελέγχαμε τις επιστρεφόμενες τιμές από τις ρουτίνες της MySQL που είχαμε χρησιμοποιήσει και καλούσαμε την die() για τερματισμό της εκτέλεσης του script αν εμφανιζόταν κάποιο πρόβλημα. Αν θέλουμε όμως να τυπώσουμε περισσότερες πληροφορίες στον browser σχετικές με το λάθος τότε μπορούμε να χρησιμοποιήσουμε τις mysql_errno() και mysql_error(), όπου η πρώτη επιστρέφει έναν αριθμό για το λάθος και η δεύτερη ένα string.

Έτσι το προηγούμενο κομμάτι κώδικα θα γίνει τώρα :

```
$database = "DBname";
mysql_select_db( $DBname )

or die ( "Couldn't open
$Dbname".mysql_error() );
```

2.2.13 Προσθήκη Δεδομένων σε Πίνακα

Τώρα που έχουμε πρόσβαση στη βάση δεδομένων μας, μπορούμε να προσθέσουμε δεδομένα σε έναν πίνακα. Για να το κάνουμε αυτό θα πρέπει να κατασκευάσουμε ένα SQL ερώτημα. Η PHP για το σκοπό αυτό μας παρέχει την ρουτίνα mysql_query(). Η mysql_query() απαιτεί ένα string που να περιέχει ένα SQL ερώτημα και, προαιρετικά, ένα αναγνωριστικό συνδέσμου.

Αν ο αναγνωριστικό δεν δοθεί, το ερώτημα στέλνεται στο διακομιστή βάσης δεδομένων με τον οποίο έγινε η τελευταία σύνδεση. Η mysql_query() επιστρέφει μια θετική τιμή αν το ερώτημα είναι πετυχημένο. Αν το ερώτημα μας περιέχει ένα συντακτικό λάθος, ή αν δεν έχουμε άδεια προσπέλασης της βάσης, τότε επιστρέφει false.

Στο παρακάτω κομμάτι κώδικα προσθέτουμε μια γραμμή σε έναν πίνακα με όνομα customer και πεδία τα c_id INT και c_name VARCHAR(20) :

```
$link = mysql_connect( "localhost" );
$query = "INSERT INTO customer ( c_id, c_name )
values( '123', 'Sianos' )";
mysql_query( $query, $link )
or die ( "Couldn't add data to \"customer\" table: "

.mysql_error() );
```

2.2.14 Ανάκτηση Δεδομένων από Πίνακα

Αφού έχουμε εκτελέσει ένα SELECT ερώτημα, μπορούμε να χρησιμοποιήσουμε ένα βρόχο για να προσπελάσουμε κάθε γραμμή που βρέθηκε. Η PHP διατηρεί έναν εσωτερικό δείκτη που κρατά ένα αντίγραφο της θέσης μας μέσα στο σύνολο που βρέθηκε. Όταν προσπελάσουμε μια γραμμή ο δείκτης αυτός θα πάει στην επόμενη.

Μπορούμε εύκολα να πάρουμε έναν πίνακα των πεδίων κάθε γραμμής χρησιμοποιώντας την mysql_fetch_row(). Η ρουτίνα αυτή απαιτεί ένα

αναγνωριστικό αποτελέσματος και επιστρέφει έναν πίνακα που περιέχει κάθε πεδίο της γραμμής. Όταν φτάσουμε στο τέλος του συνόλου που βρέθηκε, η mysql_fetch_row() επιστρέφει false.

Στο παρακάτω κομμάτι κώδικα βλέπουμε πως λειτουργεί η mysql_fetch_row() :

```
$result = mysql_query( "SELECT * FROM customer" );
print "<table border=1>\n";
while ( $a_row = mysql_fetch_row( $result ) )
{
print "<tr>\n";
foreach ( $a_row as $field )
print "\t<td>$field</td>\n";
print "</tr>\n";
}

print "</table>\n";
```

2.2.15 Αλλαγή Δεδομένων

Μπορούμε να αλλάξουμε τα δεδομένα χρησιμοποιώντας τη ρουτίνα mysql_query() σε συνδυασμό με ένα UPDATE. Ένα επιτυχές UPDATE δεν αλλάζει απαραίτητα κάποια γραμμή. Θα πρέπει να χρησιμοποιήσουμε μια ρουτίνα για να καλέσουμε τη mysql_affected_rows() για να διαπιστώσουμε αν αλλάξαμε τα δεδομένα στον πίνακά μας. Η mysql_affected_rows() προαιρετικά δέχεται και ένα αναγνωριστικό συνδέσμου. Αν αυτός λείπει, υπολογίζεται η πιο πρόσφατη σύνδεση. Αυτή η ρουτίνα μπορεί να χρησιμοποιηθεί με κάθε SQL ερώτημα που μπορεί να αλλάξει τη γραμμή ενός πίνακα.

Στο παρακάτω κομμάτι κώδικα αλλάζουμε την τιμή του πεδίου c_name του πίνακα customer :

```
$link = mysql_connect( "localhost" );
$query = "UPDATE customer SET c_name = '$name'
wherec_id=001";
$result = mysql_query( $query );
if ( ! $result ) die ("Couldn't update: ".mysql_error());
print "<h1>Table updated ". mysql_affected_rows() .
" row(s) changed</h1><p>";
<input type="text" name="domain">
```

2.2.16 MySQL Γενικά

Πολλοί είναι εκείνοι που έχουν συνδυάσει την άνοδο της "δημοτικότητας" της MySQL με την αντίστοιχη του Linux. Το τελευταίο, αν και εξακολουθεί να παρουσιάζει ασυμβατότητες με κάποια καινούργια περιφερειακά, έχει κερδίσει τη συμπάθεια και "φροντίδα" χρηστών και προγραμματιστών σε ολόκληρο τον πλανήτη. Στηριζόμενο στο ισχυρότερο -σε μια καταναλωτική αγορά- πλεονέκτημά του, το χαμηλό του κόστος υιοθετείται τώρα και από εταιρείες πληροφορικής και internet. Το συγκεκριμένο λειτουργικό σύστημα τους δίνει την ευκαιρία να προσφέρουν στους πελάτες τους οικονομικότερες υπηρεσίες (πχ. φιλοξενία ιστοσελίδων).

Ακολουθώντας το ίδιο μοντέλο χρήσης με το Linux, το GNU (General Public License), το σύστημα MySQL σιγά σιγά καθιερώνεται στη συνείδηση των ειδικών πληροφορικής σαν ένα ποιοτικό προϊόν, αντίστοιχο της Access της Microsoft. Φυσικά δε μπορεί να συγκριθεί με τον MS SQL Server ή την Oracle, για πολύ μεγάλες εγκαταστάσεις και μεγάλες βάσεις.

Συνοπτικά μπορεί κανείς να αναφέρει τα πλεονεκτήματα της MySQL:

- Είναι οικονομική (και μερικές φορές δωρεάν)
- Είναι πιο κατάλληλη για χρήση στο Internet
- Είναι ιδιαίτερα βελτιστοποιημένη για ταχύτητα στην ανάκτηση δεδομένων
- Παρέχει ευκολίες στο backup

- Είναι συμβατή και μεταφέρσιμη σε διάφορες πλατφόρμες και για διάφορα εργαλεία ανάπτυξης

Η MySQL είναι ένα σύστημα διαχείρισης βάσεων δεδομένων. Ακολουθεί το σχεσιακό μοντέλο (relational) και είναι συμβατή με ANSI-SQL. Είναι η πιο δημοφιλής, ανοιχτού κώδικα, SQL βάση δεδομένων που αναπτύσσεται και παρέχεται από την MySQL AB. Η MySQL AB είναι μια εμπορική εταιρεία που παρέχει υπηρεσίες γύρω από τη βάση δεδομένων MySQL.

Το MySQL Database λογισμικό είναι ένα client/server σύστημα που αποτελείται από έναν πολυνηματικό(multithreaded) SQL server που υποστηρίζει διαφορετικά backends, αρκετά διαφορετικά προγράμματα πελάτη και βιβλιοθήκες, εργαλεία διαχειριστή(administrative tools), και ένα ευρύ φάσμα προγραμματιστικών διασυνδέσεων(APIs).

Η έκδοση MySQL Server είναι κατάλληλη για τη διαχείριση μεγαλύτερων βάσεων δεδομένων και την παρουσίαση δεδομένων σε web site στο internet. Παρέχει υψηλή απόδοση και ασφάλεια.

Σε αντίθεση με την Access, η MySQL μπορεί να χρησιμοποιηθεί σε ένα μεγάλο αριθμό λειτουργικών συστημάτων (Windows, Linux, AIX, Solaris κα). Είναι γραμμένη σε C και C++ οπότε με τη χρήση των αντίστοιχών compiler γίνεται δυνατή η χρήση της σε οποιοδήποτε σύστημα.

Η Access έχει ένα προτεινόμενο όριο 20 ταυτόχρονων συνδέσεων από χρήστες του internet σε εφαρμογές Internet και δυναμικών site. Ο όρος ταυτόχρονη σύνδεση, αναφέρεται σε επίπεδο διεργασίας (process) λειτουργικού συστήματος και όχι γενικά σύνδεση επσκέπτη στο site που τη χρησιμοποιεί. Απλά όσο αυξάνονται οι επισκέπτες του site, τόσο αυξάνονται και οι ταυτόχρονες "αιτήσεις" στη βάση πίσω από το site. Η MySQL μπορεί να χειριστεί μέχρι 255 ταυτόχρονες συνδέσεις. Παράλληλα είναι και ταχύτερη από την Access.

Πέρα από τον αριθμό χρηστών, η απόδοσή της είναι καλύτερη και σε μεγαλύτερο όγκο βάσεων δεδομένων. Οι βάσεις Access με μεγάλο μέγεθος (πάνω

από 100MB) αρχίζουν να "σέρνονται" ενώ οι αντίστοιχες MySQL συνεχίζουν με την ίδια απόδοση. Αν και η έκδοση MS Access έχει βελτιωθεί αρκετά, εξακολουθεί να υστερεί της MySQL.

Η MySQL υστερεί σε εφαρμογές με πολύπλοκες σχεσιακές βάσεις. Παράλληλα απουσιάζει υποστήριξη για αποθηκευμένες διαδικασίες (stored procedures, event procedures της Access) καθώς και transaction processing.

Η επερχόμενή της έκδοση 5.1 ερχεται να καλύψει τα κενά που προαναφέρθηκαν

2.2.17 Η ιστορία της SQL και των σχεσιακών βάσεων δεδομένων.

Η SQL βασίστηκε σε ιδέες που προτάθηκαν στις αρχές του 1970 από τον ερευνητή Ted Codd των εργαστηρίων IBM San José Research Laboratories. Αρχικά δημιουργήθηκε με σκοπό να παρέχει μία ημιφυσική γλώσσα για το IBM System Relational database system. Αρχικά οι σχεσιακές βάσεις δεδομένων είχαν πολλούς, και πιο αποδοτικούς, ανταγωνιστές, π.χ. συστήματα διαχείρισης βάσεων δεδομένων τα οποία βασίζονταν σε δικτυακά μοντέλα δεδομένων. Ωστόσο, όλο και πιο αποδοτικά σχεσιακά συστήματα άρχισαν να εμφανίζονται στα τέλη του 1980 κι αυτό οδήγησε στην επικράτηση των σχεσιακών βάσεων δεδομένων και της SQL.

2.3 Apache Server

2.3.1 Γενικά

Ο Apache HTTP Server (εξυπηρετητής), γνωστός και ως απλά Apache είναι ένας web server που έχει παίξει κύριο ρόλο στην ανάπτυξη του World Wide Web και το 2009 έγινε ο πρώτος web server που ξεπέρασε τις εκατό εκατομμύρια ιστοσελίδες. Ο Apache είναι ο μόνος που παρέμεινε “ζωντανός” απέναντι στον Netscape Communications Corporation και ως Sun Java System Web Server) και από τότε εξελίχθητε στο να ανταγωνίζεται άλλους Unix web servers, σε θέματα λειτουργικότητας και απόδοσης. Η πλειοψηφία των web servers που χρησιμοποιούν Apache είναι Linux web servers.

Ο Apache δημιουργήθηκε και συντηρήθηκε από μια ομάδα σχεδιαστών υπό την αιγίδα της Apache Software Foundation. Το λογισμικό είναι διαθέσιμο για μια μεγάλη ποικιλία λειτουργικών συστημάτων όπως Unix, GNU, FreeBSD, Linux, Solaris, Novell NetWare, Mac OS X, Microsoft Windows, OS/2, TPF και eComStation. Ο Apache χαρακτηρίζεται ως δωρεάν λογισμικό ανοιχτού κώδικα.

Από τον Απρίλιο του 1996 ο Apache έγινε ο πιο δημοφιλής HTTP server. Από τον Μάρτιο του 2009 εξυπηρετεί το 46% των δικτυακών τόπων και πάνω από το 66% είναι πολυσύχναστες.

2.3.2 Ιστορία και όνομα

Η πρώτη έκδοση του Apache web server δημιουργήθηκε από τον Rober McCol ο οποίος ήταν σε μεγάλο βαθμό συσχετισμένος με τον web server του National Center for Supercomputing Applications, γνωστός απλώς ως NCSA HTTPd. Όταν ο McCool έφυγε από το NCSA στα μέσα του 1994, η ανάπτυξη του HTTPd σταμάτησε, αφήνοντας μια ποικιλία από patches για βελτιώσεις που κυκλοφορούσαν μέσω ηλεκτρονικού ταχυδρομείου. Τα patches αυτά δόθηκαν και από πολλούς άλλους προγραμματιστές εκτός από τον McCool: Brian Behlendorf, Roy Fielding, Rob Hartill, David Robinson, Cliff Skolnick, Randy Terbush, Robert S. Thau, Andrew Wilson, Eric Hagberg, Frank Peters και Nicolas Pioch και κατά συνέπεια βοήθησαν έτσι ώστε να θεσπιστεί η "ομάδα Apache".

Σύμφωνα με το ίδρυμα του Apache, το όνομα επιλέχθηκε από σεβασμό για την φυλή των Apache. Η πρώτη εξήγηση στηρίχθηκε σε ένα συνέδριο Apache και σε μια συνέντευξή το 2000 από τον Brian Behlendorf, ο οποίος είπε ότι το όνομα σημαίνει "Take no prisoners. Be kind of aggressive and kick some ass". Ο Behlendorf διέψευσε όμως αυτή την εκδοχή στην συνέντευξη του το 2007, αναφέροντας ότι ο Apache server δεν ονομάστηκε προς τιμήν της φυλής, αλλά ότι τόσες πολλές αναθεωρήσεις έγιναν και η ομάδα τον χαρακτήρισε “a patch Web Server”. Και οι δύο εξηγήσεις είναι μάλλον αποδεκτές αν και το λογοπαίγνιο με την εξήγηση έχει πέσει σε δυσμένεια.

Στην δεύτερη έκδοση του Apache server ξαναγράφτηκε μεγάλο τμήμα κώδικα με αυτό που υπήρχε στην πρώτη έκδοση του Apache. Δόθηκε έντονη έμφαση στην ρύθμιση και την ανάπτυξη της φορητότητας, το Apache Portable Runtime. Ο πυρήνας της δεύτερης έκδοσης έχει αρκετές σημαντικές βελτιώσεις σε σχέση με την πρώτη εκδοση του λογισμικου. Μέσα σε αυτές τις βελτιώσεις είναι το UNIX threading, καλύτερη υποστήριξη για μη πλατφόρμες Unix (όπως τα Microsoft Windows), νέο Apache API, και υποστήριξη διευθύνσεων IP 6^{ης} έκδοσης. Η πρώτη διαθέσιμη έκδοση του Apache 2 ήταν τον Μάρτιο του 2000 σε επίπεδο έκδοσης alpha.

2.3.3 Χαρακτηριστικά

Ο Apache server υποστηρίζει ποικίλα χαρακτηριστικά, πολλά από τα οποία επέκτειναν την λειτουργικότητα του πυρήνα. Τέτοια χαρακτηριστικά μπορεί να είναι από προγραμματισμό στην πλευρά του server έως και υποστήριξη authentication. Υπάρχει υποστήριξη και άλλων παρόμοιων γλωσσών όπως η Perl, η Phyton, η tcl και η PHP. Υπάρχει υποστήριξη πολλών στοιχείων authentication, συμπεριλαμβάνεται και η υποστήριξη SSL και TLS.

Υπάρχουν μέθοδοι συμπίεσης έτσι ώστε να βοηθήσουν στη μείωση του μεγέθους των διαδικτυακών τόπων που μεταφέρονται μέσω του πρωτοκόλλου http. Τα αρχεία log του Apache μπορούν να αναλυθούν μέσω κάποιου φυλλομετρητή χρησιμοποιώντας scripts όπως το AWStats/W3Perl ή το Visitors.

Η εικονική φιλοξενία επιτρέπει μία εγκατάσταση του Apache να μπορεί να εξυπηρετεί πολλές διαφορετικές σελίδες. Δηλαδή ένα μηχάνημα με μία και μόνο εγκατάσταση του Apache θα μπορεί να εξυπηρετεί ταυτόχρονα πολλές σελίδες με διαφορετικό όνομα.

Υπάρχει η δυνατότητα ρύθμισης των μηνυμάτων σφάλματος, δυνατότητα διαχείρισης βάσεων δεδομένων. Τέλος ο Apache server υποστηρίζει διάφορα γραφικά περιβάλλοντα για τη διαχείρισή του.

2.3.4 Χρήση

Ο Apache server χρησιμοποιείται κυρίως για να εξυπηρετεί τόσο ιστοσελίδες με στατικό περιεχόμενο αλλά και δυναμικές ιστοσελίδες για τον παγκόσμιο ιστό. Πολλές εφαρμογές στον ιστό είναι έτσι σχεδιασμένες ώστε να ανταποκρίνονται στο περιβάλλον και στα χαρακτηριστικά που προσφέρει ο Apache server.

Είναι στοιχειό του δημοφιλούς WAMP μαζί με την MySQL και τις γλώσσες προγραμματισμού PHP, PERL, Python. Περιέχεται και στις αντίστοιχες εκδόσεις του WAMP για άλλα λειτουργικά συστήματα.

Ο Apache server υπάρχει σαν τμήμα διαφόρων πακέτων λογισμικού, συμπεριλαμβανομένης της βάσης δεδομένων της Oracle και του λογισμικού της IBM, WebSphere Application Server. Το λειτουργικό Mac OS X ενσωματώνει τον Apache server ως υποστήριξη για τον WebObjects Application Server. Υποστηρίζεται ακόμη κατά κάποιο τρόπο από την Borland στο Kylix αλλά και σε εργαλεία ανάπτυξης της Delphi. Επίσης συμπεριλαμβάνεται με τον Novell NetWare 6.5, που είναι ο προεπιλεγμένος web server. Ο Apache server περιλαμβάνεται επίσης σε πολλές διανομές Linux.

Ο Apache χρησιμοποιείται και για πολλά άλλα καθήκοντα, όπου το περιεχόμενο μιας σελίδας θα πρέπει να διατίθενται με ασφαλή και αξιόπιστο τρόπο. Ένα τέτοιο παράδειγμα είναι η κοινή χρήση αρχείων από έναν προσωπικό υπολογιστή μέσω Internet. Ένας χρήστης που έχει εγκαταστήσει τον Apache μπορεί να βάλει αρχεία στον φάκελο αρχείων του λογισμικού και διάφοροι χρήστες να μπορούν να έχουν πρόσβαση σε αυτά.

Οι προγραμματιστές διαδικτυακών εφαρμογών έχουν μία έκδοση του apache server εγκατεστημένη στον υπολογιστή τους έτσι ώστε να μπορούν να δουν την ιστοσελίδα που σχεδιάζουν και να έχουν ευκολότερη διόρθωσή της.

Ο κύριος ανταγωνιστής του apache είναι το λογισμικό της Microsoft, Internet Information Services (IIS). Ακολουθεί το λογισμικό της Sun Microsystems,

Sun Java System Web Server και πολλές άλλες εφαρμογές, όπως ο Zeus Web Server. Ορισμένες από τις μεγαλύτερες ιστοσελίδες στον κόσμο χρησιμοποιούν Apache. Το Google βασίζεται σε μια τροποποιημένη έκδοση του Apache, που ονομάζεται Google Web Server (GWS).

2.3.5 Άδεια χρήσης

Η άδεια χρήσης είναι ένα ξεχωριστό μέρος του Apache HTTP Server, είναι λογισμικό ανοιχτού κώδικα. Ωστόσο ο Apache επιτρέπει την διανομή τόσο σε ανοιχτό αλλά και σε κλειστό πηγαίο κώδικα.

Το όνομα Apache είναι ένα καταχωρημένο και μπορεί μόνο να χρησιμοποιηθεί με ρητή άδεια του κατόχου.

2.4 Πρωτόκολλο HTTP και URL

Το Πρωτόκολλο Μεταφοράς Υπερκειμένου (HyperText Transfer Protocol, HTTP) είναι η κύρια μέθοδος που χρησιμοποιούν τα πρωτόκολλα του Παγκοσμίου Ιστού για να μεταφέρουν δεδομένα από έναν διακομιστή (server) σε ένα πελάτη (client). Η ανάπτυξη του HTTP έγινε υπό την εποπτεία του World Wide Web Consortium και του Internet Engineering Task Force (IETF).

Το HTTP είναι ένα γενικό, αντικειμενοστραφές πρωτόκολλο που μεταβιβάζει πληροφορία μεταξύ των διακομιστών και των πελατών. Πελάτης ονομάζεται ο τελικός χρήστης, και ο εξυπηρετητής ο υπολογιστής ο οποίος έχει την ιστοσελίδα. Η προεπιλεγμένη πορτα (port) στην οποία δουλεύει το πρωτόκολλο είναι η 80.

Ξεκίνησε από την έκδοση HTTP/0.9 κατά τη διάρκεια της πρώιμης ανάπτυξης του διαδικτύου και ακολούθησε η έκδοση HTTP/1.0 το 1995. Η πιο πρόσφατη έκδοσή του, HTTP/1.1, παρέχει περισσότερη λειτουργικότητα και υποστηρίζει πολλαπλές συναλλαγές μεταξύ πελάτη και διακομιστή κατά τη διάρκεια της ίδιας αίτησης.

Μια συναλλαγή βασισμένη στο πρωτόκολλο HTTP αποτελείται από τα ακόλουθα στάδια:

- Σύνδεση – Ο πελάτης αποκαθιστά σύνδεση με τον διακομιστή
- Αίτηση **–** Ο πελάτης στέλνει το μήνυμα της αίτησης στον διακομιστή
- Απάντηση **–** Ο διακομιστής διαδικτύου στέλνει την απάντηση
- Κλείσιμο **–** Η σύνδεση τερματίζεται από τον διακομιστή

Το URL είναι η διεύθυνση ενός αρχείου μέσα στο διαδίκτυο ή σε κάποιο τοπικό δίκτυο και αποτελείται από 3 τμήματα:

- Το πρωτόκολλο που πρέπει να χρησιμοποιήσουμε για να αποκτήσουμε αυτό το αρχείο
- Ο κόμβος στον οποίο είναι τοποθετημένο το αρχείο.
- Το directory (με το πλήρες path ή τμήμα αυτού κατά περίπτωση) που βρίσκεται το αρχείο και φυσικά το όνομα του αρχείου.

Για παράδειγμα στην περίπτωση της εφαρμογής που μελετάμε ας υποθέσουμε ότι ο Apache server είναι στο μηχάνημα με διεύθυνση IP 192.168.1.3 για να αποκτήσουμε πρόσβαση στο login της εφαρμογής θα πρέπει να πληκτρολογήσουμε http://192.168.1.3/login.php. Τα url θέλουν ιδιαίτερη προσοχή κατά την πληκτρολόγηση γιατί είναι case sensitive δηλαδή υπάρχει διαφοροποίηση μεταξύ κεφαλαίων και πεζών γραμμάτων.

Για την προβολή της σελίδας μέσω διαδικτύου και την αναπαράσταση στο TEI χρησιμοποιήθηκε για domain name ένα λογαριασμός μέσω Dyn Dns. Στο adsl δρομολογητή που διαθέτουμε μπαίνει το host name και έπειτα ενημερώνεται το domain name με την διευθυνση ip που έχει εκεινη τη στιγμή ο δρομολογητής και μαζί του και ο apache server στη port 80. Αποτέλεσμα της όλης διαδικασίας είναι σαν URL να βάζουμε http://dmas1985.selfip.net/login.php αντί να γράφαμε

http://94.65.123.67/login.php. Η διεύθυνση ip ειναι δυναμική στις adsl συνδεσεις και έτσι θα έπρεπε να μαθαίνουμε κάθε φορά τη διευθυνση ip του δρομολογητη για να έχουμε πρόσβαση στην εφαρμογή. Με το dyn dns θέλουμε μόνο το domain name.

2.5 HTML

Τα αρχικά HTML προέρχονται από τις λέξεις HyperText Markup Language. Η html δεν είναι μια γλώσσα προγραμματισμού. Είναι μια περιγραφική γλώσσα (markup language), δηλαδή ένας ειδικός τρόπος γραφής κειμένου. Ο καθένας μπορεί να δημιουργήσει ένα αρχείο HTML χρησιμοποιώντας απλώς έναν επεξεργαστή κειμένου. Αποτελεί υποσύνολο της γλώσσας SGML (Standard Generalized Markup Language) που επινοήθηκε από την IBM προκειμένου να λυθεί το πρόβλημα της μη τυποποιημένης εμφάνισης κειμένων στα διάφορα υπολογιστικά συστήματα. Ο φυλλομετρητής (browser) αναγνωρίζει αυτόν τον τρόπο γραφής και εκτελεί τις εντολές που περιέχονται σε αυτόν. Αξίζει να σημειωθεί ότι η html είναι η πρώτη και πιο διαδεδομένη γλώσσα περιγραφής της δομής μιας ιστοσελίδας. Η html χρησιμοποιεί τις ειδικές ετικέτες (τα tags) για να δώσει τις απαραίτητες οδηγίες στον browser. Τα tags είναι εντολές που συνήθως ορίζουν την αρχή ή το τέλος μιας λειτουργίας. Βρίσκονται πάντα μεταξύ των συμβόλων < και >. Π.χ. <BODY> Οι οδηγίες είναι case insensitive, δεν επηρεάζονται από το αν έχουν γραφτεί με πεζά (μικρά) ή κεφαλαία. Ένα αρχείο HTML πρέπει να έχει κατάληξη htm ή html.

Για να μπορούν οι browser να ερμηνεύουν σχεδόν απόλυτα σωστά την html έχουν θεσπιστεί κάποιοι κανόνες. Αυτοί οι κανόνες είναι γνωστοί ως προδιαγραφές. Επομένως σχεδόν κάθε είδος υπολογιστή μπορεί να δείξει το ίδιο καλά μια ιστοσελίδα. Οι πρώτες προδιαγραφές ήταν η html 2.0. Πρόβλημα προέκυψε όταν η Microsoft και η Netscape πρόσθεσαν στην html τέτοιες δυνατότητες που στην αρχή τουλάχιστον ήταν συμβατές μόνο με συγκεκριμένους browser. Ακόμη και σήμερα υπάρχουν διαφορές στην απεικόνιση κάποιας σελίδας από διαφορετικούς browsers. Ιδιαίτερο είναι το πρόβλημα όταν η ιστοσελίδα, εκτός από "καθαρή" HTML περιλαμβάνει και εφαρμογές Javascript.

Οι τωρινοί κανόνες είναι η έκδοση HTML 4.0 η οποία περιέχει περισσότερα χαρακτηριστικά και αποτελεί μια προσπάθεια να μειωθούν οι ασυμβατότητες των διαφορετικών web browsers. Αυτή η έκδοση κινείται προς μια πιο λογική μέθοδο μορφοποίησης ιστοσελίδων με την χρήση "Style Sheets", που επιτρέπουν την ακρίβεια στην μορφοποίηση που επιθυμούν οι σχεδιαστές ιστοσελίδων και κατά κάποιο τρόπο την διαχωρίζουν από το περιεχόμενο, κάνοντας ευκολότερη την ανανέωση του σχεδιασμού ενός web site.

2.6 CSS

Το CSS *(Cascading Style Sheets-Διαδοχικά Φύλλα Στυλ)* ή (αλληλουχία φύλλων στύλ) είναι μια γλώσσα που ανήκει στην κατηγορία των γλωσσών φύλλων στυλ που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης. Πιο πρακτικά χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στις γλώσσες HTML και XHTML, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστοτόπου.

Το Φύλλο στύλ που εφαρμόζεται σε ένα έγγραφο μπορεί να προέρχεται από :

- το συγγραφέα μιας ιστοσελίδας
- το χρήστη του πλοηγού (browser)
- τον ίδιο τον πλοηγό, αν έχει το δικό του προκαθορισμένο φύλλο στυλ.

Υπάρχει η δυνατότητα να ορίσουμε styles και style sheets με πολλούς τρόπους. Τα Style μπορούν να ορισθούν μέσα σε ένα μόνο στοιχείο HTML, μέσα στο στοιχείο <HEAD> και με εύρος εφαρμογής μια ολόκληρη HTML σελίδα ή σ’ ένα εξωτερικό αρχείο CSS με εύρος εφαρμογής μία ή περισσότερες σελίδες. Πολλά εξωτερικά style sheets μπορούν να χρησιμοποιούνται μέσα απ’ ένα μόνο HTML έγγραφο.

Το πρόβλημα που προκύπτει είναι αν έχουμε μια σελίδα η οποία έχει πολλά css τότε κατά ποιο τρόπο θα βγει το αποτέλεσμα. Σε γενικές γραμμές όλα τα style

θα συγκλίνουν σ' ένα νέο style που αποτελεί σύνθεση όλων των στυλ σύμφωνα με τους παρακάτω κανόνες, όπου ο μεγαλύτερος αριθμός έχει και την υψηλότερη προτεραιότητα:

1. Προεπιλογή του browser.
2. Εξωτερικό Φύλλο Στυλ (External Style Sheet).
3. Εσωτερικό Φύλλο Στυλ (Internal Style Sheet), μέσα στο τμήμα HEAD του HTML εγγράφου.
4. Inline Style, μέσα στο HTML στοιχείο.

2.7 JAVASCRIPT

Η Javascript είναι μια διερμηνευόμενη γλώσσα προγραμματισμού με ορισμένες ιδιότητες και επηρεασμούς από αντικειμενοστραφείς γλώσσες προγραμματσιμού όπως η C++ και η Java, χωρίς όμως σε καμία περίπτωση να μπορεί να χαρακτηριστεί ως μια αντικειμενοστραφής γλώσσα προγραμματισμού. Είναι πολύ δημοφιλής γλώσσα και έτσι έχει ενσωματωθεί σε όλους τους γνωστούς browser.

Η εξέλιξη της Javascript ξεκίνησε από τη Netscape Communications το 1995. Η αρχικλη ιδέα στην ανάπτυξη της Javascript ήταν να φτιαχτεί μια γλώσσα προγραμματισμού που θα ενσωματωνόταν στον δημοφιλή Netscape Navigator εκείνης της εποχής και θα επέτρεπε σε μη έμπειρους προγραμματισμούς να γράψουν κώδικα που θα εισήγαγε κάποια λειτουργικότητα στις στατικές HTML σελίδες χρησιμοποιώντας μία απλή γλώσσα προγραμματισμού. Αυτή τη στιγμή φαίνεται η Javascript να έχει επικρατήσει σε πολύ μεγάλο βαθμό σε σχέση με τις υπόλοιπες σε σχέση με τις υπόλοιπες scripting γλώσσες για client side διαδυκτιακό πργραμματισμό.

Συντακτικά αλλά και ως γλώσσα προγραμματισμού η Javascript κτίστηκε πάνω στο πρότυπο των γλωσσών προγραμματισμού C, C++ και Java. Πολλά από τα δομικά τους στοιχεία θυμίζουν έντονα αυτές τις γλώσσες προγραμματισμού. Έτσι μειώνεται και ο χρόνος εκμάθησης της γλώσσας από τον χρήστη.

Αντίθετα με τις γλώσσες προγραμματισμού από τις οποίες σχεδιάστηκε η Javascript διαχειρίζεται τους τύπους δεδομένων πιο χαλαρά σε σχέση με τη σφιχτή διαχείρηση τύπων δεδομένων που γίνεται στη C, C++, Java. Στη Javascript οι μεταβλητές δεν είναι απαραίτητο να έχουν συγκεκριμένο τύπο ή ακόμη είναι δυνατόν να αλλάζουν τύπο κατά τη διάρκεια της ζωής τους.

Μια συνήθης σύγχυση γύρω από την Javascript είναι ότι αποτελεί μία απλοποιημένη μορφή της Java. Εκτός από μία πιθανή έμπνευση της Javascript από την Java και το γεγονός ότι οι δύο γλώσσες μπορούν να χρησιμοποιηθούν στο Internet η Javascript δεν έχει καμία σχέση με την Java. Η χρήση του ονόματος Javascript έγινε για λόγου προώθησης της γλώσσας Java σε μια εποχή όπου η επέκτασή της ήταν πολύ μεγάλη. Στα πρώτα στάδια κατασκευής της η Javascript λεγόταν Livescript.

2.8 WAMP

Το WAMP είναι ένα πακέτο από ανεξάρτητα προγράμματα που γίνεται εγκατάσταση σε υπολογιστές οι οποίοι χρησιμοποιούν ως λειτουργικό σύστημα το Microsoft Windows. Η συνεργασία αυτών των προγραμμάτων επιτρέπει δυναμικές ιστοσελίδες να εκτελούνται σε ένα δίκτυο υπολογιστών είτε αυτό είναι διαδίκτυο ή κάποιο τοπικό δίκτυο.

Το «WAMP» είναι ένα αρκτικόλεξο. Το «W» αναφέρεται στο λειτουργικό σύστημα (Windows), το «A» αναφέρεται στον Apache server, το «M» αναφέρεται στην MySQL και το «P» στην PHP. Ο Apache server είναι ένας web server που επιτρέπει άτομα που χρησιμοποιούν κάποιον φυλλομετρητή να συνδεθούν σε κάποιο υπολογιστή και να ανακτήσουν πληροφορίες σαν ιστοσελίδες. Η MySQL είναι ένα πρόγραμμα διαχείρισης βάσεων δεδομένων. Η PHP είναι μια scripting γλώσσα η οποία μπορεί να χειρίζεται πληροφορίες που διατηρούνται σε μια βάση δεδομένων και να δημιουργεί ιστοσελίδες εξαρχής κάθε φορά που ο φυλλομετρητής ζητά κάποιο στοιχείο. Στο πακέτο προγραμμάτων υπάρχει και το phpMyAdmin το οποίο παρέχει μία γραφική απεικόνιση για την διαχείρηση της βάσης δεδομένων MySQL.

Υπάρχουν και άλλες εκδόσεις του WAMP για άλλα λειτουργικά συστήματα. Για Linux είναι το LAMP, για Mac είναι το MAMP, για Solaris είναι το SAMP ενώ για FreeBSD είναι το FAMP.

Το WAMP είναι ένα ελεύθερο ανοιχτού κώδικα λογισμικό. Αυτό σημαίνει ότι μία δυναμική σελίδα μπορεί να δημιουργηθεί χωρίς την αγορά κάποιου ειδικού προγράμματος. Πολλές εταιρείες φιλοξενίας ιστοσελίδων εκμεταλλεύτηκαν αυτό το πλεονέκτημα έχοντας προεγκατεστημένα στοιχειά του WAMP και του LAMP στις επιλογές τους. Η άδεια χρήσης του WAMP μπορεί να είναι ελεύθερη, με δωρεά ή εμπορική.

Γενικά, μόνο κάποιος χρήστης με δικαιώματα administrator μπορεί να κάνει εγκατάσταση το πακέτο WAMP. Αυτό σημαίνει ότι το πακέτο μπορεί να γίνει εγκατάσταση μόνο σε κάποιον υπολογιστή που ο χρήστης έχει όλα τα δικαιώματα χρήσης.

2.9 phpMyAdmin

Το phpMyAdmin είναι ένα εργαλείο ανοιχτού κώδικα γραμμένο σε PHP με σκοπό την διαχείριση της MySQL στον παγκόσμιο ιστό. Μπορεί να εκτελέσει διάφορες εργασίες όπως το να δημιουργεί, τροποποιεί, διαγράφει βάσεις δεδομένων, πίνακες, πεδία και γραμμές εκτελώντας SQL ερωτήματα. Μπορεί επίσης να διαχειρίζεται χρήστες και δικαιώματα χρήσης. Το λογισμικό είναι διαθέσιμο σε 54 διαφορετικές γλώσσες. Η πρώτη κυκλοφορία έγινε τον Σεπτέμβριο του 1998.

Παρόμοια εργαλεία είναι το phpPgAdmin, το οποίο παρέχει τις ίδιες περίπου δυνατότητες αλλά για PostgreSQL. Το phpMsAdmin ή το myLittleAdmin είναι εργαλεία για την διαχείρηση του Microsoft SQL Server. Υπάρχει και ένα πιο ελαφρύ εργαλείο για την διαχείρηση βάσης δεδομένων MySQL το phpMinAdmin, το οποίο έχει όλα τα σημαντικά χαρακτηριστικά του phpMyAdmin αλλά αποτελείται από μόνο ένα αρχείο.

2.10 Εγκατάσταση και ρυθμίσεις Wamp Server

Το Wamp server είναι ελεύθερο λογισμικό και μπορούμε να το βρούμε και να το κατεβάσουμε από τη σελίδα του με μια απλή αναζήτηση στο google. Η εγκατάσταση του είναι πάρα πολύ απλή και δεν υπάρχει λόγος αναλυτικής προβολής της εγκατάστασης. Στο ερώτημα της εγκατάστασης για το πρωτόκολλο smtp και το email αφήνουμε τα πεδία ως έχουν αφού δεν χρησιμοποιείται στην εφαρμογή η συνάρτηση mailto(). Το προεπιλεγμένο username είναι το root και password είναι κενό. Τα στοιχεία αυτά αφορούν τη σύνδεση στη βάση μέσω της εφαρμογής αλλά και τη σύνδεση στο phpMyAdmin.

Για να δουλέψει ο apache server του wamp μέσω internet θα πρέπει να κάνουμε forward την port 80, στη διεύθυνση IP του μηχανήματος που διαθέτει τον server. Θα πρέπει η τοπική διεύθυνση IP να είναι στατική έτσι ώστε η κίνηση για την port 80 να προωθηθεί σωστά στον server.

Αφού τελειώσει η εγκατάσταση τότε επιλέγουμε το Wamp start server από το μενού έναρξη και τον φάκελο Wamp. Ο server ξεκινάει και τρέχει κάτω δεξιά στην ώρα. Κάνοντας αριστερό κλικ πάνω του μπορούμε να τον θέσουμε Online πατώντας put Online έτσι ώστε να μπορούν οι χρήστες να έχουν πρόσβαση στην εφαρμογή. Αν θέλουμε να έχουμε πρόσβαση και στο phpMyAdmin μέσω internet θα πρέπει να κανουμε κλικ στο wamp server να πάμε στον apache server έπειτα στο Alias Directories μετά στο http://localhost/phpMyAdmin και να πατήσουμε edit Alias. Το αρχείο που θα ανοίξει είναι αυτό που καθορίζει τις διευθύνσεις IP που θέλουμε να έχουν πρόσβαση στο phpMyAdmin. Το αρχείο που δείχνει πως πρέπει να είναι συμπληρωμένο φαίνεται παρακάτω.

```
Alias /phpmyadmin "c:/wamp/apps/phpmyadmin3.1.1/"

<Directory "c:/wamp/apps/phpmyadmin3.1.1/">
    Options Indexes FollowSymLinks MultiViews
    AllowOverride all
    Order Allow,Deny
    Allow from all
</Directory>
```

ΚΕΦΑΛΑΙΟ 3^{ο} : Υλοποίηση Βάσης Δεδομένων

3.1 Οντότητες και συσχετίσεις

Όταν μοντελοποιούμε την πραγματικότητα, απεικονίζουμε ένα σύστημα που έχει να κάνει με οντότητες και συσχετίσεις. Οντότητα είναι η κάθε περίπτωση μονάδας του πραγματικού συστήματος την οποία ενδιαφερόμαστε να παρακολουθήσουμε πληροφοριακά και η οποία έχει αυτόνομη ύπαρξη μέσα στον κόσμο της υπό ανάπτυξη εφαρμογής. Η κάθε οντότητα έχει μια σειρά από χαρακτηριστικά, τα οποία θέλουμε να γνωρίζουμε και να επεξεργαζόμαστε. Οι τιμές των χαρακτηριστικών καθορίζουν (στον κόσμο της εφαρμογής) την κάθε μια στιγμή οντότητας και την διακρίνουν από τις υπόλοιπες στιγμές της ίδιας οντότητας. Ένα από τα χαρακτηριστικά ή κάποιος συνδυασμός τους διακρίνεται από την ικανότητα που έχει να προσδιορίζει μονοσήμαντα, με την τιμή του, την κάθε στιγμή της οντότητας. Αυτό είναι το κύριο κλειδί της οντότητας . Υπάρχουν περιπτώσεις όπου περισσότερα του ενός χαρακτηριστικά ή συνδυασμοί χαρακτηριστικών μπορούν να παίξουν τον ρόλο του κυρίου κλειδιού για μια οντότητα. Μιλάμε τότε για πολλά υποψήφια κλειδιά. Αν συμβεί αυτό και επιλέξουμε ένα από αυτά να γίνει κύριο κλειδί, τα υπόλοιπα ονομάζονται εναλλακτικά κλειδιά.

Στο χώρο του μοντέλου της εφαρμογής , οι οντότητες δεν είναι απομονωμένες αλλά αλληλεπιδρούν με το περιβάλλον τους. Οι αλληλεπιδράσεις

απεικονίζονται υπό τη μορφή συσχετίσεων που ορίζουν να υπάρχουν ανάμεσα σε δύο η περισσότερες οντότητες.

3.2 Κανονικοποίηση

Στο σχεσιακό περιβάλλον βάσης δεδομένων, όλοι οι πίνακες πρέπει να είναι κανονικοποιημένοι , τουλάχιστον, σε πρώτο βαθμό κανονικοποίησης, δηλαδή να μην περιέχουν επαναλαμβανόμενες ομάδες δεδομένων. Ισοδύναμα η στήλη του κάθε πίνακα δε νοείται να αντιστοιχεί σε πεδίο που παίρνει πολλαπλές και όχι μια τιμή.

Ένας πίνακας που είναι σε πρώτη κανονική μορφή (1NF) είναι πίνακας σύμφωνα με το σχεσιακό μοντέλο , μπορεί όμως να παρουσιάζει προβληματική συμπεριφορά στις εισαγωγές, ενημέρωσης και τις διαγραφές των δεδομένων του. «Προβληματική συμπεριφορά» σημαίνει είτε δυνατότητα να συμβεί λάθος στην καταχώρηση / ενημέρωση των δεδομένων είτε σπατάλη χώρου καταχώρησης των δεδομένων στο δίσκο.

Η θεωρία της κανονικοποίησης αναπτύχθηκε με στόχο να ανιχνεύεται και να προλαμβάνεται μια τέτοιου είδους προβληματική κατάσταση όταν η εφαρμογή είναι ακόμη στο στάδιο του σχεδιασμού της.

Στην κανονικοποίηση με διάσπαση, η κάθε περίπτωση μη κανονικοποιημένου πίνακα αντιμετωπίζεται με διαδοχικές διασπάσεις που καταλήγουν σε ένα ισοδύναμο σύνολο πινάκων όπου ο κάθε ένας είναι σε τρίτη κανονική μορφή (3NF)

3.2.1 Πρώτη κανονική μορφή

Ένα πίνακας είναι σε πρώτη κανονική μορφή (1NF) τότε και μόνον τότε όταν στην κάθε εγγραφή του , η κάθε μια στήλη παίρνει ατομικές τιμές. Με άλλα λόγια, δεν επιτρέπεται να έχουμε πεδία / στήλες στον πίνακα που να παίρνουν πολλαπλές και όχι ατομικές τιμές.

Αν ο πίνακας έλθει σε πρώτη κανονική μορφή δεν υπάρχουν δύο γραμμές που είναι εντελώς όμοιες. Ένα η περισσότερα πεδία / στήλες είναι ικανά με την τιμή τους να προσδιορίζουν μονοσήμαντα την κάθε μια γραμμή. Ακόμη και αν είναι ο πίνακας σε πρώτη κανονική μορφή είναι πολύ πιθανό να υπάρχουν περιττές επαναλήψεις και έτσι υπάρχει η ανάγκη της δεύτερης κανονικής μορφής.

3.2.2 Δεύτερη Κανονική Μορφή

Ένας πίνακας είναι σε δεύτερη κανονική (2NF) μορφή όταν είναι σε πρώτη κανονική μορφή (1NF) και επιπλέον ικανοποιεί την εξής συνθήκη : Το κάθε πεδίο του που δε συμμετέχει στο σχηματισμό του κύριου κλειδιού εξαρτάται από το σύνολο του κυρίως κλειδιού και τίποτα λιγότερο. Αυτή η επιπλέον συνθήκη αφαιρεί έναν σημαντικό αριθμό από περιττές (και επικίνδυνες για λάθη) επαναλήψεις της ίδιας πληροφορίας. Φυσικά , μπαίνει θέμα να εξετάσουμε έναν πίνακα ώστε να είναι σε 2NF μορφή μόνον όταν το κύριο κλειδί είναι σύνθετο. Πίνακες οι οποίοι είναι σε 1NF μορφή και τον οποίων το κύριο κλειδί είναι απλό (αποτελείται από ένα μόνο πεδίο) είναι αυτόματα και 2NF.

Για να φέρουμε έναν πίνακα από 1NF σε 2NF , παίρνουμε τα πεδία του πίνακα που εξαρτώνται από μέρος του σύνθετου κλειδιού μαζί με ένα αντίγραφο του μέρους του κλειδιού (από το οποίο εξαρτώνται τα πεδία) και τα μετακομίζουμε σε έναν άλλο πίνακα.

3.2.3 Τρίτη κανονική μορφή

Ένας πίνακας είναι σε τρίτη κανονική μορφή (3NF) όταν:

- Όλα τα πεδία που δε συμμετέχουν στο σχηματισμό του κύριου κλειδιού του εξαρτώνται συναρτησιακά από το κύριο κλειδί και τίποτα λιγότερο από αυτό (2NF).

- Δεν υπάρχει πεδίο που να μην συμμετέχει στο σχηματισμό του κύριου κλειδιού το οποίο να εξαρτάται συναρτησιακά από άλλο πεδίο που επίσης

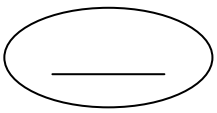
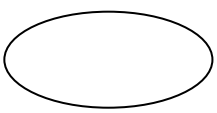
δεν συμμετέχει στο σχηματισμό του κύριου κλειδιού. Εξαίρεση αποτελεί, φυσικά, η εξάρτηση πεδίου από εναλλακτικό κλειδί , όταν υπάρχει.

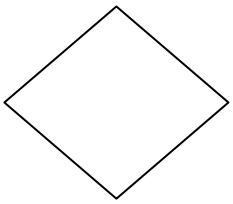
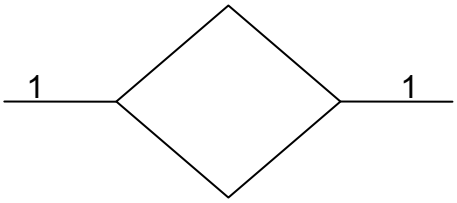
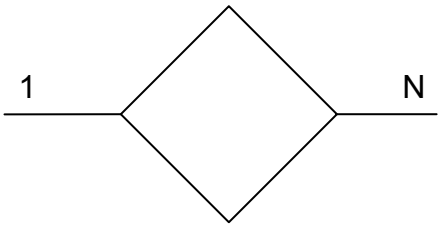
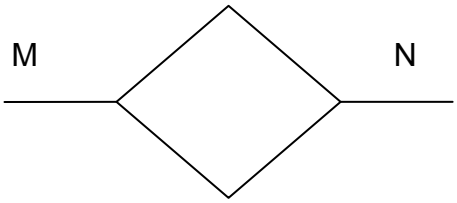
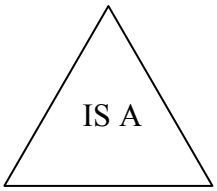
Ισοδύναμα, ένας πίνακας είναι σε 3NF όταν κάθε πεδίο που δε συμμετέχει στο σχηματισμό του κύριου κλειδιού εξαρτάται συναρτησιακά από το κύριο κλειδί, τίποτα λιγότερο από αυτό το κύριο κλειδί και μόνο το κύριο κλειδί . Ένας πίνακας σε 3NF δεν περιλαμβάνει έμμεσες συναρτησιακές εξαρτήσεις μεταξύ των πεδίων του.

Για να μετασχηματιστεί ένας πίνακας από 2NF σε ισοδύναμο σχήμα 3NF, ανιχνεύεται η πιθανή ύπαρξη έμμεσων συναρτησιακών εξαρτήσεων. Οι τελευταίες, αν υπάρχουν, αντιμετωπίζονται κατά τα γνωστά με μετακινήσεις ομάδων πεδίων σε άλλο πίνακα.

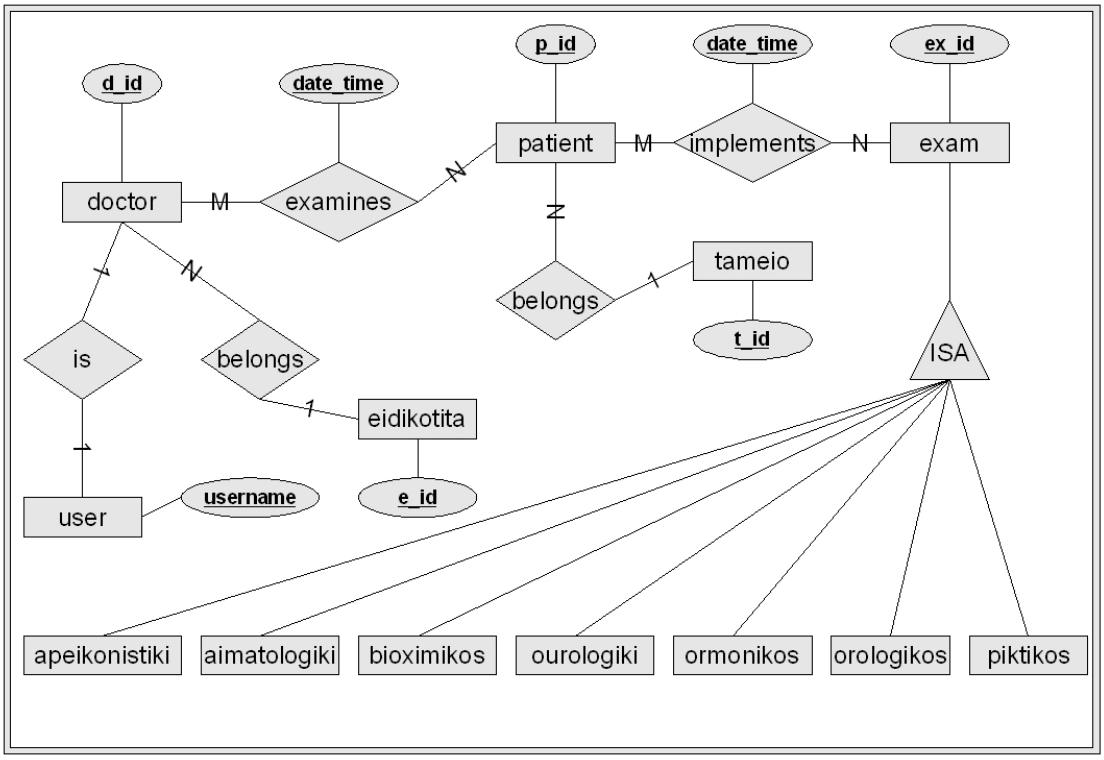
3.3 Διάγραμμα Οντοτήτων – Συσχετίσεων (ER)

Οντότητες και συσχετίσεις παρουσιάζονται διαγραμματικά, για ένα συγκεκριμένο μοντέλο εφαρμογής, με το λεγόμενο διάγραμμα ER (Entry – Relationship Diagram). Αυτή η διαγραμματική τεχνική, επιτρέπει τη σχεδίαση των οντοτήτων μιας βάσης και των συσχετίσεων που υφίστανται ανάμεσα τους χρησιμοποιώντας ειδικά σύμβολά τα οποία παρουσιάζονται στον πίνακα που ακολουθεί :

	Οντότητα
	Κύριο Κλειδί Οντότητας
	Χαρακτηριστικό Οντότητας

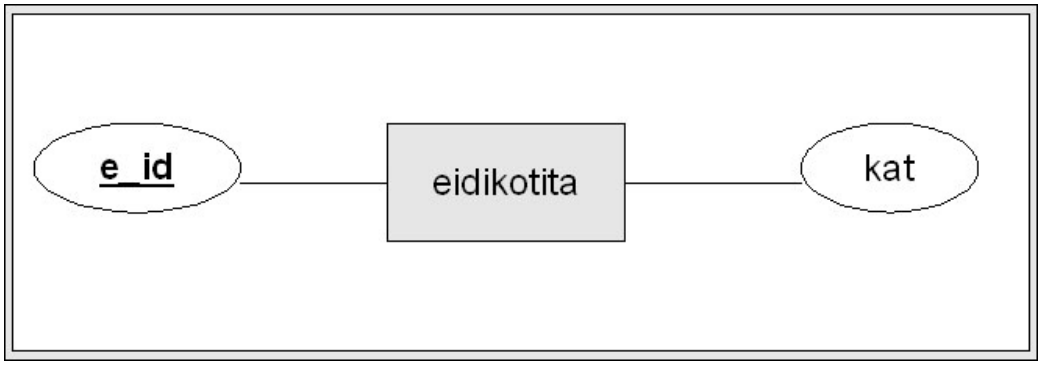
	Συσχέτιση
	Συσχέτιση 1-1
	Συσχέτιση 1-N
	Συσχέτιση M-N
	Ιεραρχία Is A

Παρακάτω φαίνεται το διάγραμμα ER του υπό ανάπτυξη συστήματος. Για την απλούστερη και πιο ουσιαστική απεικόνιση του συστήματος δεν εμφανίζονται τα χαρακτηριστικά κάθε οντότητας.

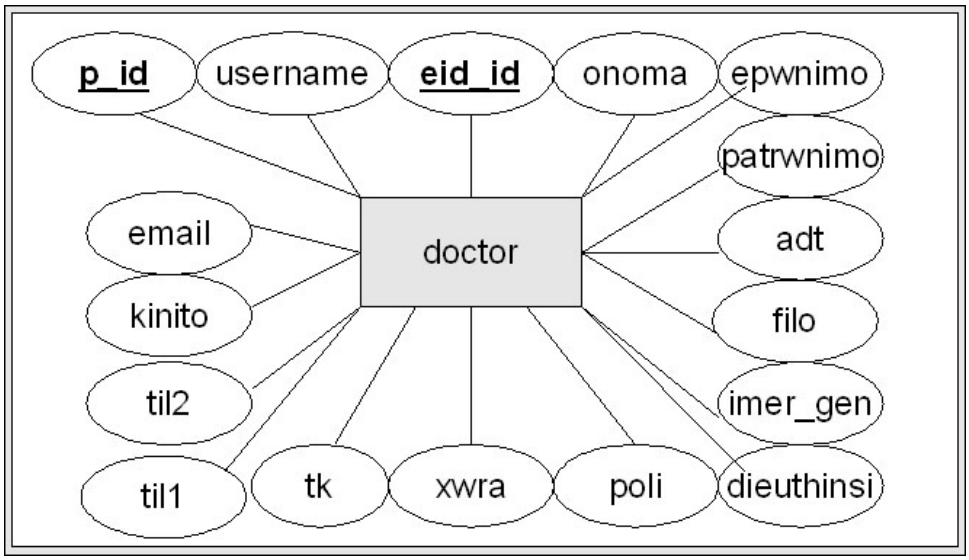


ER Εφαρμογής

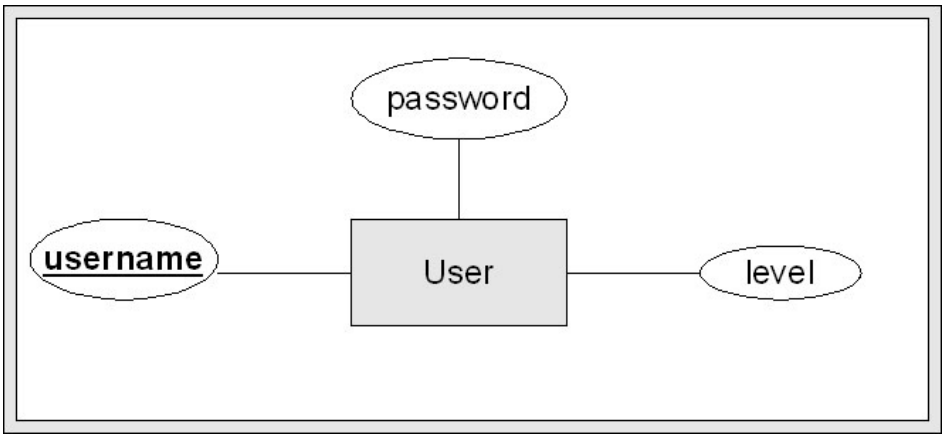
3.3.1 Οντότητα Ειδικότητα



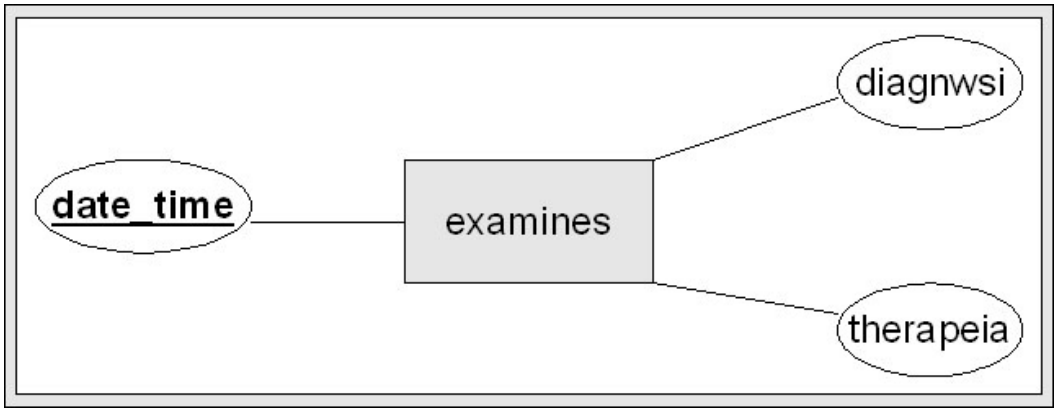
3.3.2 Οντότητα Ιατρός



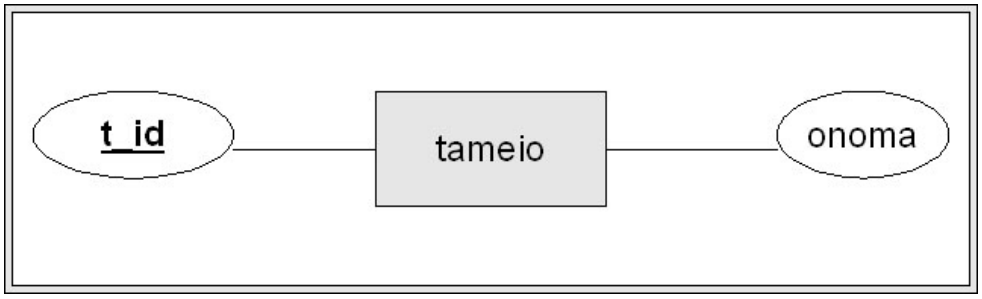
3.3.3 Οντότητα Χρήστης



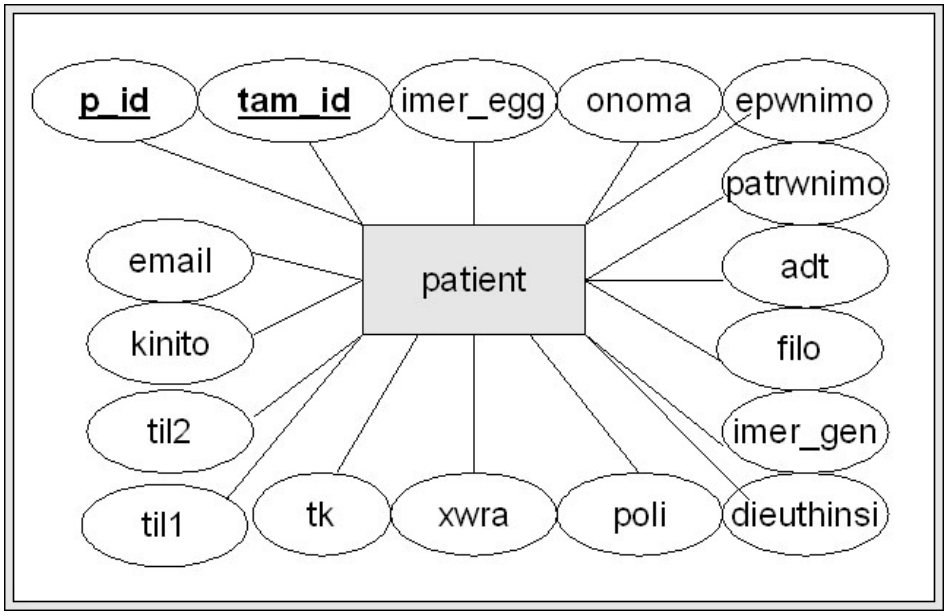
3.3.4 Συσχέτιση Εξετάζει



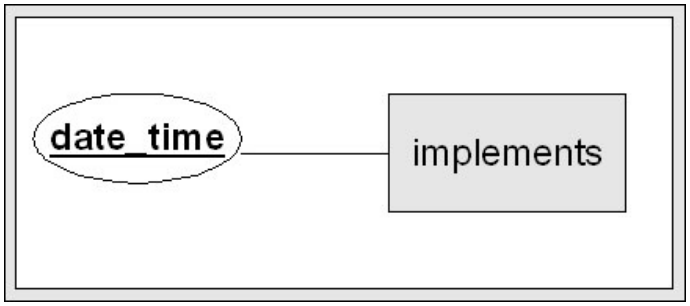
3.3.5 Οντότητα Ταμείο



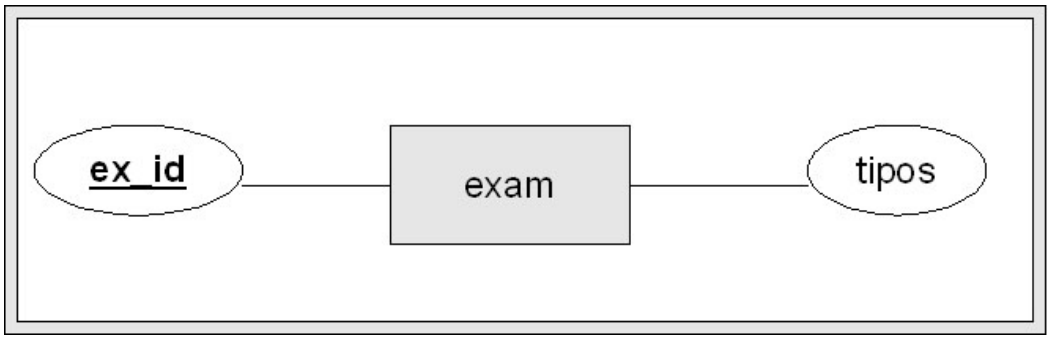
3.3.6 Οντότητα Ασθενής



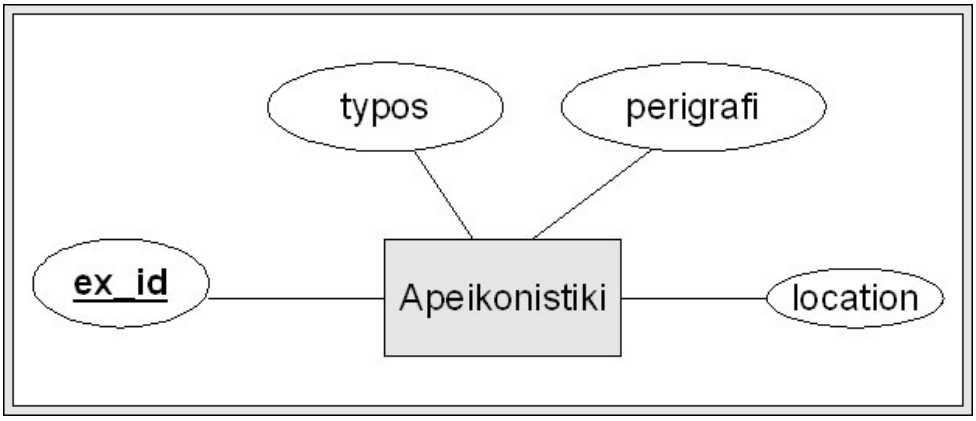
3.3.7 Συσχέτιση Πραγματοποιεί



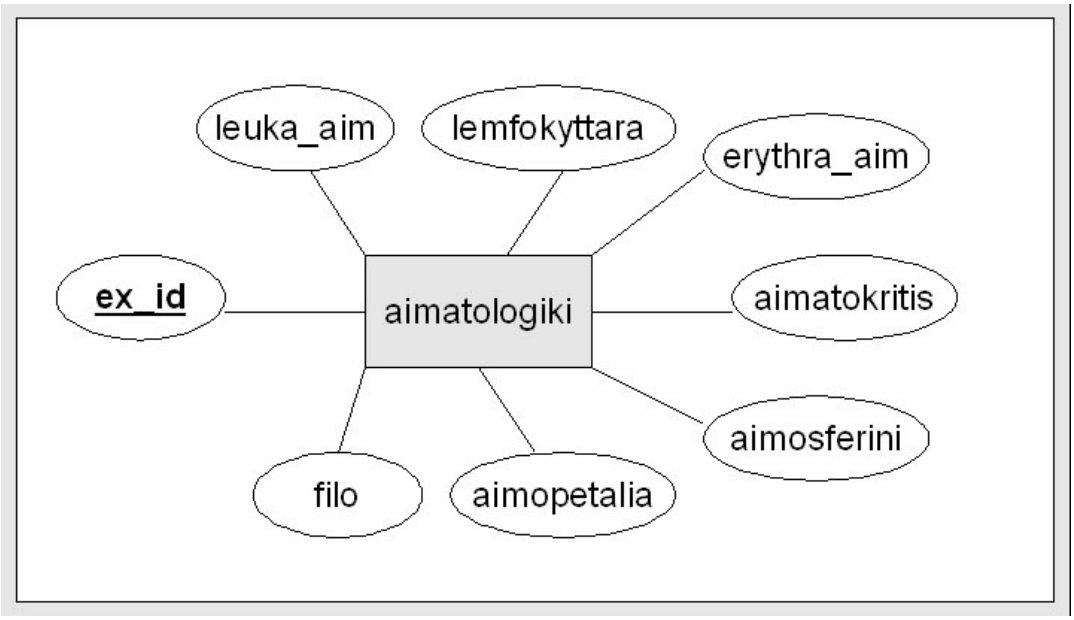
3.3.8 Οντότητα Εξέταση



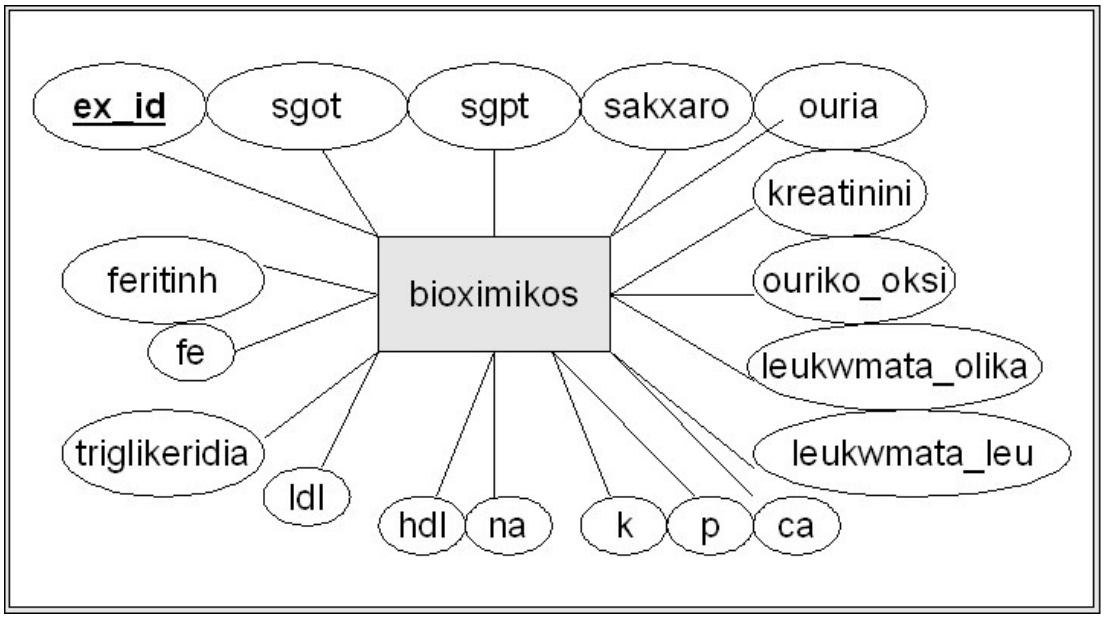
3.3.9 Οντότητα Απεικονιστικές



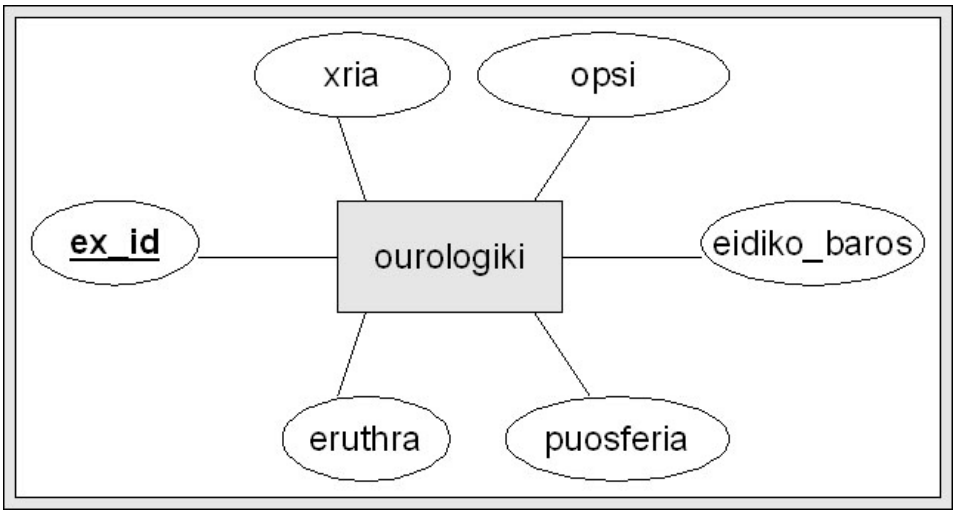
3.3.10 Οντότητα Αιματολογική



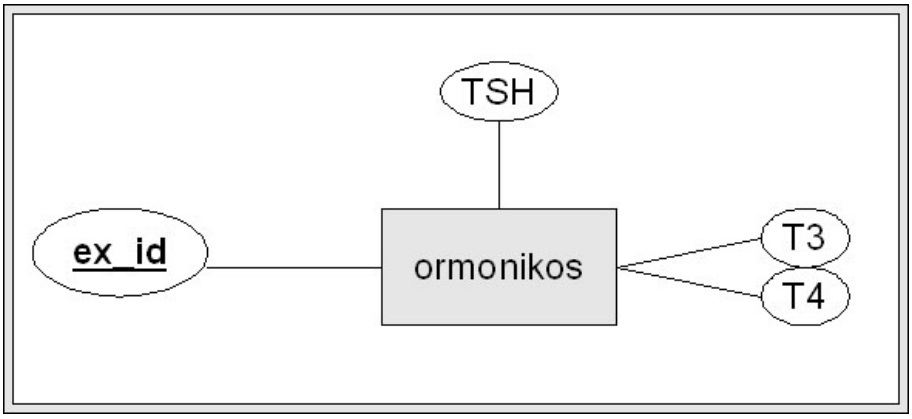
3.3.11 Οντότητα Βιοχημικός



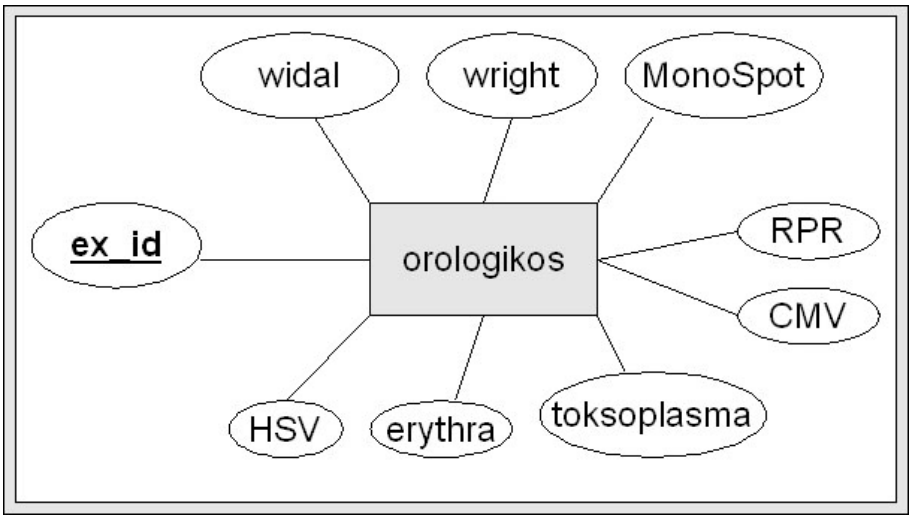
3.3.12 Οντότητα Ουρολογική



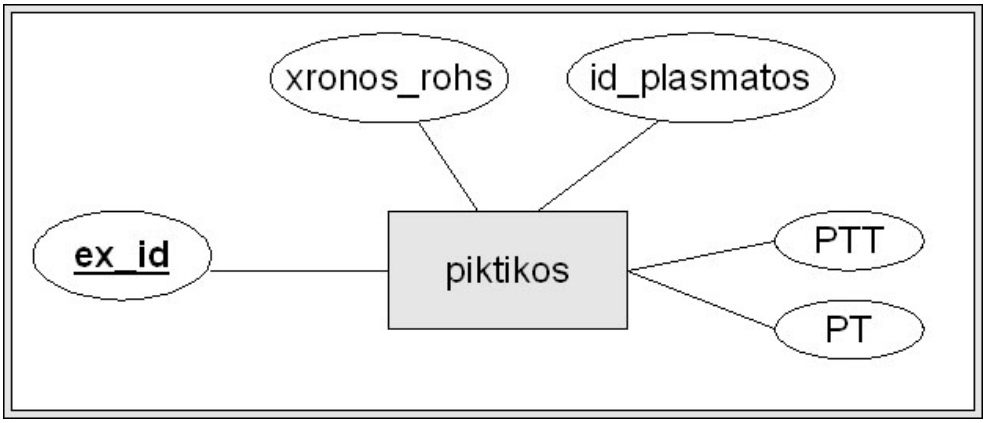
3.3.13 Οντότητα Ορμονικός



3.3.14 Οντότητα Ορολογικός

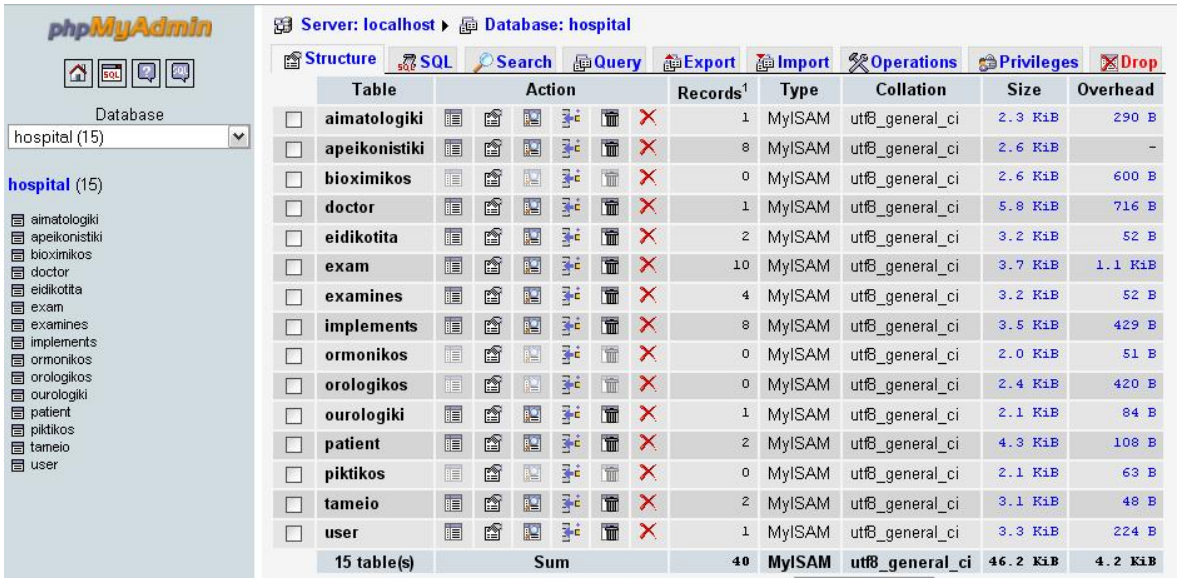


3.3.15 Οντότητα Πηκτικός



3.4 Τεκμηρίωση Πινάκων

Για την υλοποίηση της εφαρμογής χρησιμοποιούνται 15 πίνακες. Παρακάτω εμφανίζονται οι πίνακες μέσα από το phpmyadmin ενώ ακολουθούν τα πίνακες με τα χαρακτηριστικά κάθε οντότητας και τον τύπο του χαρακτηριστικού.



3.4.1 Πίνακας – Οντότητα eidikotita (Ιατρική ειδικότητα)

eidikotitia (**e_id**, kat)

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
e_id	Ακέραιος
kat	Κείμενο

3.4.2 Πίνακας - Οντότητα User (Χρήστης)

user (username, password, level**)**

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
username	Κείμενο
password	Κείμενο
level	Ακέραιος

3.4.3 Πίνακας - Οντότητα Doctor (Ιατρός)

doctor (d_id, username, eid_id, onoma, epwnimo, patrwnimo, adt, filo, imer_gen, dieuthinsi, poli, xwra, tk, til1, til2, kinito, email**)**

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
d_id	Ακέραιος
username	Κείμενο
eid_id	Ακέραιος
onoma	Κείμενο
epwnimo	Κείμενο
patrwnimo	Κείμενο
adt	Κείμενο
filo	Κείμενο
imer_gen	Ημερομηνία
dieuthinsi	Κείμενο
poli	Κείμενο
xwra	Κείμενο
tk	Κείμενο
til1	Κείμενο
til2	Κείμενο
kinito	Κείμενο
email	Κείμενο

3.4.4 Πίνακας - Οντότητα tameio (Ταμείο ασφάλισης)

tameio (**t_id**, onoma)

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
t_id	Ακέραιος
onoma	Κείμενο

3.4.5 Πίνακας - Οντότητα Patient (Ασθενής)

patient (p_id, onoma, epwnimo, patrwnimo, adt, filo, imer_gen, dieuthinsi, poli, xwra, tk, tam_id, til1, til2, kinito, email**)**

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
p_id	Ακέραιος
onoma	Κείμενο
epwnimo	Κείμενο
patrwnimo	Κείμενο
adt	Κείμενο
filo	Κείμενο
imer_gen	Ημερομηνία
dieuthinsi	Κείμενο
poli	Κείμενο
xwra	Κείμενο
tk	Κείμενο
tam_id	Ακέραιος
ımer_gen	Ημερομηνία
til1	Κείμενο
til2	Κείμενο
kinito	Κείμενο
email	Κείμενο

3.4.6 Πίνακας - Συσχέτιση Examines (Εξετάζει)

Examines (d_id, p_id, date_time, diagnwsi, therapeia**)**

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
d_id	Ακέραιος
p_id	Ακέραιος
date_time	Timestamp
diagnwsi	Κείμενο

therapeia	Κείμενο

3.4.7 Πίνακας - Οντότητα Exam (Εξέταση)

Exam (ex_id, tipos**)**

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
ex_id	Ακέραιος
tipos	Κείμενο

3.4.8 Πίνακας - Συσχέτιση Implements (Πραγματοποιεί)

Implements (p_id, ex_id, date_time)

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
p_id	Ακέραιος
ex_id	Ακέραιος
date_time	Timestamp

3.4.9 Πίνακας - Οντότητα aimatologiki (Αιματολογική Εξέταση)

aimatologiki (ex_id, leuka_aim, lemfokyttara, erythra_aim, aimatokritis, aimosferini, aimopetalia)

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
ex_id	Ακέραιος
leuka_aim	Δεκαδικός με ένα δεκαδικό ψηφίο
lemfokyttara	Δεκαδικός με ένα δεκαδικό ψηφίο
erythra_aim	Δεκαδικός με ένα δεκαδικό ψηφίο
aimatokritis	Δεκαδικός με ένα δεκαδικό ψηφίο
aimosferini	Δεκαδικός με ένα δεκαδικό ψηφίο
aimopetalia	Ακέραιος

3.4.10 Πίνακας - Οντότητα apeikonistiki (Απεικονιστική Εξέταση)

Apeikonistiki (ex_id, typos, perigrafi, location**)**

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
ex_id	Ακέραιος
typos	Κείμενο
perigrafi	Κείμενο
location	Κείμενο

3.4.11 Πίνακας - Οντότητα bioximikos (Βιοχημικός Έλεγχος)

Bioximikos (**ex_id**, sgot, sgpt, sakxaro, ouria, kreatinini, ouriko_oks, ca, p, k, na, leukwmata_olika, leukwmata_leukwmatines, hdl, ldl, triglikeridia, fe, fritinh)

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
ex_id	Ακέραιος
Sgot	Ακέραιος
sgpt	Ακέραιος
Sakxaro	Ακέραιος
Ouria	Ακέραιος
Kreatinini	Δεκαδικός με ένα δεκαδικό ψηφίο
Ouriko_oksi	Δεκαδικός με ένα δεκαδικό ψηφίο
ca	Δεκαδικός με ένα δεκαδικό ψηφίο
p	Δεκαδικός με ένα δεκαδικό ψηφίο
k	Δεκαδικός με ένα δεκαδικό ψηφίο
na	Δεκαδικός με ένα δεκαδικό ψηφίο
Leukwmata_olika	Δεκαδικός με ένα δεκαδικό ψηφίο
Leukwmata_leukwmatines	Δεκαδικός με ένα δεκαδικό ψηφίο
hdl	Δεκαδικός με ένα δεκαδικό ψηφίο
ldl	Δεκαδικός με ένα δεκαδικό ψηφίο
triglikeridia	Ακέραιος

fe	Ακέραιος
feritinh	Ακέραιος

3.4.12 Πίνακας - Οντότητα ourologiki (Ουρολογική εξέταση)

Ourologiki (**ex_id**, xria, opsi, eidiko_baros, puosferia, eruthra)

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
ex_id	Ακέραιος
xria	Ακέραιος
opsi	Ακέραιος
eidiko_baros	Ακέραιος
puosferia	Κείμενο
eruthra	Κείμενο

3.4.13 Πίνακας - Οντότητα omronikos (Ορμονικός έλεγχος)

ormonikos (**ex_id**, TSH, T4, T3)

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
ex_id	Ακέραιος
TSH	Δεκαδικός με δύο δεκαδικό ψηφίο
T4	Δεκαδικός με δύο δεκαδικό ψηφίο
T3	Ακέραιος

3.4.14 Πίνακας - Οντότητα Orologikos (Ορολογικός έλεγχος)

orologikos (**ex_id**, widal, wright, MonoSpot, RPR, CMV, toksoplasma, erythra, HSV, EBV)

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
ex_id	Ακέραιος
widal	Ακέραιος
wright	Ακέραιος
MonoSpot	Ακέραιος
RPR	Ακέραιος
CMV	Ακέραιος
toksoplasma	Ακέραιος
erythra	Ακέραιος
HSV	Ακέραιος
EBV	Ακέραιος

3.4.15 Πίνακας - Οντότητα piktikos (Πηκτικός έλεγχος)

piktikos (**ex_id**, xronos_rohs, PT, PTT, id_plasmatos)

Χαρακτηριστικό	Τύπος Χαρακτηριστικού
ex_id	Ακέραιος
xronos_rohs	Ακέραιος
PT	Ακέραιος
PTT	Ακέραιος
id_plasmatos	Ακέραιος

3.5 Ερωτήματα SQL δημιουργίας πινάκων

3.5.1 Δημιουργία πίνακα Ειδικότητα

```
CREATE TABLE Eidikotita
(e_id INTEGER AUTO_INCREMENT,
kat VARCHAR(255) UNIQUE NOT NULL,
PRIMARY KEY (e_id));
```

Το κύριο κλειδί του πίνακα είναι το e_id το οποίο φτιάχνεται αυτόματα με την εντολή auto increment και είναι δεκαδικός. Το όνομα της ειδικότητας δηλαδή το χαρακτηριστικό kat είναι κείμενο και είναι μοναδικό για τον πίνακα ειδικότητα. Επίσης το πεδίο δεν μπορεί να μείνει κενό κατά την εισαγωγή του ονόματος στη βάση.

3.5.2 Δημιουργία πίνακα Χρήστη

```
CREATE TABLE User(
username varchar(255) UNIQUE NOT NULL,
password varchar(255) NOT NULL,
level integer(2) NOT NULL,
PRIMARY KEY (username));
```

Το κύριο κλειδί του πίνακα είναι το username και είναι κείμενο. Υπάρχει το χαρακτηριστικό password το οποίο δεν μπορεί να είναι κενό και αποθηκεύεται ο κωδικός πρόσβασης με το κρυπτογραφημένος με md5. Υπάρχει και το χαρακτηριστικό level που καθορίζει το είδος και την κατάσταση του χρήστη.

3.5.3 Δημιουργία πίνακα Ιατρού

```
CREATE TABLE Doctor
(d_id INTEGER NOT NULL AUTO_INCREMENT,
username VARCHAR(15) UNIQUE NOT NULL,
eid_id INTEGER NOT NULL,
onoma VARCHAR(20) NOT NULL,
epwnimo VARCHAR(30) NOT NULL,
patrwnimo VARCHAR(20),
adt VARCHAR(20) UNIQUE NOT NULL,
filo VARCHAR(20),
imer_gen DATE,
dieuthinsi VARCHAR(40),
poli VARCHAR(20),
xwra VARCHAR(20),
tk VARCHAR(10),
til1 VARCHAR(15),
til2 VARCHAR(15),
kinito VARCHAR(15),
email VARCHAR(30),
PRIMARY KEY (d_id),
FOREIGN KEY (eid_id) REFERENCES Eidikotita(e_id));
```

Το κύριο κλειδί του πίνακα ιατρός είναι το d_id το οποίο είναι ακέραιος και φτιάχνεται αυτόματα. Ξένο κλειδί είναι το eid_id και αναφέρεται στο πίνακα ειδικότητα. Η σχέση αυτή είναι 1-N, ένας ιατρός ανήκει σε μία ειδικότητα και μία ειδικότητα έχει πολλούς ιατρούς. Το χαρακτηριστικό adt αντιπροσωπεύει τον αριθμό ταυτότητας και είναι μοναδικό χαρακτηριστικό στον πίνακα.

3.5.4 Δημιουργία πίνακα Ταμείου

```
CREATE TABLE Tameio
(t_id INTEGER NOT NULL AUTO_INCREMENT,
onoma VARCHAR(30) UNIQUE NOT NULL,
PRIMARY KEY (t_id));
```

Το κύριο κλειδί του πίνακα είναι το t_id το οποίο φτιάχνεται αυτόματα με την εντολή auto increment και είναι δεκαδικός. Το όνομα του ταμείου δηλαδή το χαρακτηριστικό onoma είναι κείμενο και είναι μοναδικό για τον πίνακα tameio. Επίσης το πεδίο δεν μπορεί να μείνει κενό κατά την εισαγωγή του ονόματος στη βάση.

3.5.5 Δημιουργία πίνακα Ασθενούς

```
CREATE TABLE Patient
(p_id INTEGER NOT NULL AUTO_INCREMENT,
onoma VARCHAR(20) NOT NULL,
epwnimo VARCHAR(30) NOT NULL,
patrwnimo VARCHAR(20),
adt VARCHAR(20) UNIQUE NOT NULL,
filo VARCHAR(20),
imer_gen DATE,
dieuthinsi VARCHAR(40),
poli VARCHAR(20),
xwra VARCHAR(20),
tk VARCHAR(10),
tam_id INTEGER NOT NULL,
imer_egg DATE,
til1 VARCHAR(15),
til2 VARCHAR(15),
kinito VARCHAR(15),
email VARCHAR(30),
PRIMARY KEY (p_id),
FOREIGN KEY (tam_id) REFERENCES Tameio(t_id));
```

Το κύριο κλειδί του πίνακα ασθενής είναι το p_id το οποίο είναι ακέραιος και φτιάχνεται αυτόματα. Ξένο κλειδί είναι το tam_id και αναφέρεται στο πίνακα tameio. Η σχέση αυτή είναι 1-N, ένας ασθενής ανήκει σε ένα ταμείο και ένα ταμείο έχει πολλούς ασθενείς. Το χαρακτηριστικό adt αντιπροσωπεύει τον αριθμό ταυτότητας και είναι μοναδικό χαρακτηριστικό στον πίνακα.

3.5.6 Δημιουργία πίνακα Εξετάζει

```
CREATE TABLE Examines
(d_id INTEGER NOT NULL,
p_id INTEGER NOT NULL,
diagnwsi VARCHAR(80),
therapeia VARCHAR(80),
date_time TIMESTAMP,
PRIMARY KEY (d_id, p_id, date_time),
FOREIGN KEY (d_id) REFERENCES Doctor,
FOREIGN KEY (p_id) REFERENCES Patient);
```

Για κύριο κλειδί έχουμε ένα σύνθετο κλειδί το οποίο αποτελείται από τα ξένα κλειδιά d_id, p_id και το date_time. Το date time είναι τύπου timestamp. Για την καταγραφή της διάγνωσης έχουμε το πεδίο diagnwsi και για τη θεραπεία το πεδίο θεραπεία. Ο πίνακας δείχνει μια σχέση M-N. Ένας ιατρός εξετάζει πολλούς ασθενής και ένας ασθενής εξετάζεται από πολλούς ιατρούς.

3.5.7 Δημιουργία πίνακα Εξέταση

```
CREATE TABLE Exam
(ex_id INTEGER NOT NULL AUTO_INCREMENT,
tipos VARCHAR(20) NOT NULL,
PRIMARY KEY(ex_id));
```

Για κύριο κλειδί έχουμε το ex_id το οποίο είναι δεκαδικός και δημιουργείται αυτόματα κατά την εισαγωγή εξέτασης. Ο τύπος αντιπροσωπεύει τον τύπο εξέτασης που έχουμε. Ο πίνακας exam είναι η κεφαλή μιας ιεραρχίας ISA όπου από κάτω κρέμονται όλες οι εξετάσεις.

3.5.8 Δημιουργία πίνακα Πραγματοποιεί

```
CREATE TABLE Implements
(p_id INTEGER NOT NULL,
ex_id INTEGER NOT NULL,
date_time TIMESTAMP,
PRIMARY KEY (p_id,ex_id,date_time),
FOREIGN KEY (p_id) REFERENCES Patients,
FOREIGN KEY (ex_id) REFERENCES Exam);
```

Για κύριο κλειδί έχουμε ένα σύνθετο κλειδί το οποίο αποτελείται από τα ξένα κλειδιά p_id, ex_id και το date_time. Το date time είναι τύπου timestamp. Ο πίνακας δείχνει μια σχέση M-N. Ένας ασθενής πραγματοποιεί πολλές εξετάσεις και μία εξέταση πραγματοποιείται από πολλούς ασθενής.

3.5.9 Δημιουργία πίνακα Αιματολογική

```
CREATE TABLE aimatologiki
(ex_id INTEGER NOT NULL,
leuka_aim FLOAT(3,1),
lemfokyttara FLOAT(3,1),
erythra_aim FLOAT(3,1),
aimatokritis FLOAT(3,1),
aimosferini FLOAT(3,1),
aimopetalia INTEGER(4),
PRIMARY KEY (ex_id),
FOREIGN KEY (ex_id) REFERENCES Exam);
```

Για κύριο κλειδί έχουμε το ex_id το οποίο είναι το ίδιο με το ex_id της εξέτασης. Ομοίως είναι και όλες οι εξετάσεις.

3.5.10 Δημιουργία πίνακα Απεικονιστική

```
CREATE TABLE Apeikonistiki
(ex_id INTEGER NOT NULL,
typos VARCHAR(20),
perigrafi VARCHAR(200),
location VARCHAR(250),
PRIMARY KEY (ex_id),
FOREIGN KEY (ex_id) REFERENCES Exam);
```

3.5.11 Δημιουργία πίνακα Βιοχημικός

```
CREATE TABLE bioximikos
(ex_id INTEGER NOT NULL,
sgot INTEGER(3),
sgpt INTEGER(3),
sakxaro INTEGER(3),
ouria INTEGER(3),
kreatinini FLOAT(2,1),
ouriko_oksi FLOAT(3,1),
ca FLOAT(3,1),
p FLOAT(3,1),
k FLOAT(3,1),
na INTEGER(3),
leukwmata_olika FLOAT(3,1),
leukwmata_leukwmatines FLOAT(3,1),
hdl FLOAT(4,1),
ldl FLOAT(4,1),
triglikeridia INTEGER(3),
fe INTEGER(3),
feritinh INTEGER(3),
PRIMARY KEY (ex_id),
FOREIGN KEY (ex_id) REFERENCES Exam);
```

3.5.12 Δημιουργία πίνακα Ουρολογικής

```
CREATE TABLE ourologiki
(ex_id INTEGER NOT NULL,
xria INT,
opsi INT,
eidiko_baros INTEGER(4),
puosferia VARCHAR(20),
eruthra VARCHAR(20),
PRIMARY KEY (ex_id),
FOREIGN KEY (ex_id) REFERENCES Exam);
```

3.5.13 Δημιουργία πίνακα Ορμονικός

```
CREATE TABLE ormonikos
(ex_id INTEGER NOT NULL,
TSH FLOAT(4,2),
T4 FLOAT(4,2),
T3 INT(3),
PRIMARY KEY (ex_id),
FOREIGN KEY (ex_id) REFERENCES Exam);
```

3.5.14 Δημιουργία πίνακα Ορολογικός

```
CREATE TABLE orologikos
(ex_id INTEGER NOT NULL,
widal INT,
wright INT,
MonoSpot INT,
RPR INT,
CMV INT,
toksoplasma INT,
erythra INT,
HSV INT,
EBV INT,
PRIMARY KEY (ex_id),
FOREIGN KEY (ex_id) REFERENCES Exam);
```

3.5.15 Δημιουργία πίνακα Πηκτκός

```
CREATE TABLE piktikos
(ex_id INTEGER NOT NULL,
xronos_rohs INTEGER(2),
PT INTEGER(2),
PTT INTEGER(3),
id_plasmatos INTEGER(3),
PRIMARY KEY (ex_id),
FOREIGN KEY (ex  id) REFERENCES
```

Κεφάλαιο 4^{ο} : Τεκμηρίωση κώδικα Javascript, CSS και PHP-HTML

Στο κεφάλαιο που ακολουθεί παρουσιάζονται και εξηγούνται σημαντικά τμήματα κώδικα της php τα οποία εμπεριέχονται στην εφαρμογή. Στην ιεραρχία των φακέλων της εφαρμογής υπάρχει ένας φάκελος με όνομα includes. Σε αυτό τον φάκελο περιέχονται αρχεία τα οποία τα κάνουμε include στην εφαρμογή για να μην επαναλαμβάνεται ο ίδιος κώδικας. Τέτοια αρχεία είναι τύπου css, js, php και αρχεία φωτογραφίας gif, jpeg και png.

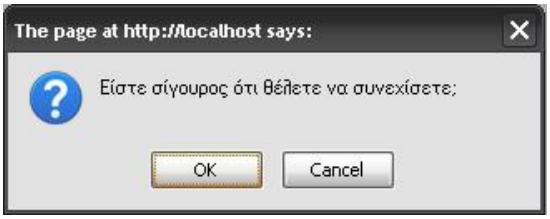
4.1 Τεκμηρίωση Javascript

4.1.1 Αρχείο confirm_continue.js

Το αρχείο αυτό είναι πολύ χρήσιμο κατά την λειτουργία της εφαρμογής. Χρησιμοποιείται σε κρίσιμα σημεία της εφαρμογής όπου χρειάζεται κάποια επιβεβαίωση για την συνέχιση μιας ενέργειας. Συγκεκριμένα χρησιμοποιείται κατά την υποβολή στοιχείων και κατά την διαγραφή κάποιας οντότητας. Παρακάτω παρουσιάζεται ο κώδικας του script.

```
function confirm_continue(){

        var agree=confirm("Είστε σίγουρος ότι θέλετε να συνεχίσετε;");

        if (agree)
                return true ;
        else
                return false ;
}
```

Το αποτέλεσμα αυτού του κώδικα είναι να εμφανιστεί ένα παραθυρο με τις επιλογές OK και Cancel. Αν ο χρήστης πατήσει OK τότε η εφαρμογή συνεχίζει την ενεργεία ενώ αν πατήσει ενώ αν πατήσει Cancel η εφαρμογή δεν προχωρά στην διεκπεραίωση της ενεργείας. Ακολουθεί παραδείγμα σχετικά με την confirm continue.



4.1.2 Αρχείο verify_form.js

Το αρχείο αυτό είναι ένα ακόμη πολύ χρήσιμο αρχείο. Περιέχει συναρτήσεις που αφορούν τις φόρμες εισαγωγής στοιχείων στη βάση δεδομένων. Δεν επιτρέπει το χρήστη να αφήσει κάποιο πεδίο κενό του οποίου η συμπλήρωση

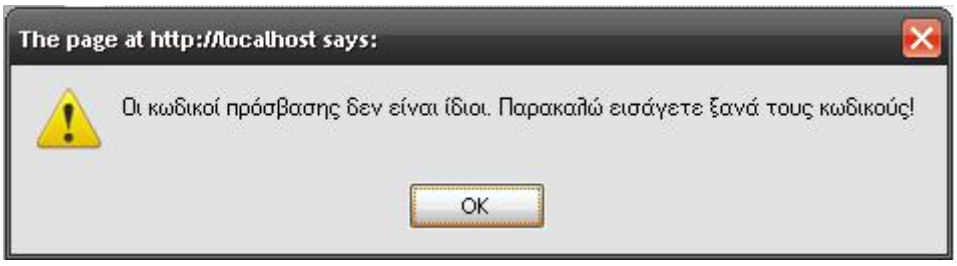
είναι υποχρεωτική. Σε περίπτωση που ο χρήστης αφήσει κάποιο υποχρεωτικό πεδίο κενό επιστρέφεται ένα μήνυμα που αναφέρει ότι η συμπλήρωση του πεδίου είναι απαραίτητη. Επίσης η κάθε συνάρτηση περιέχει και την λειτουργικότητα της confirm_continue(). Πρόσθετες λειτουργίες έχουν προστεθεί σε κάθε συνάρτηση ανάλογα με τις ανάγκες που απαιτεί η κάθε φόρμα. Παρακάτω παρουσιάζεται η συνάρτηση verify_doctor() η οποία είναι υπεύθυνη για την υποβολή των στοιχείων σχετικά με τη δημιουργία γιατρού.

```
function verify_doctor() {

	var themessage = "Η συμπλήρωση των παρακάτω στοιχείων είναι
υποχρεωτική:\n";

	if (document.myform.element_1_1.value=="") {
		themessage = themessage + "-Όνομα\n";
	}
	if (document.myform.element_1_2.value=="") {
		themessage = themessage + "-Επώνυμο\n";
	}
	if (document.myform.element_2.value=="") {
		themessage = themessage + "-Πατρώνυμο\n";
	}
	if (document.myform.element_3.value=="") {
		themessage = themessage + "-Αριθμός Δελτίου Ταυτότητας\n";
	}
	if (document.myform.element_14.value=="") {
		themessage = themessage + "-Ειδικότητα\n";
	}
	if (document.myform.element_11.value=="") {
		themessage = themessage + "-Όνομα χρήστη\n";
	}
	if (document.myform.element_12.value=="") {
		themessage = themessage + "-Κωδικός πρόσβασης\n";
	}
	if (document.myform.element_15.value=="") {
		themessage = themessage + "-Επανάληψη κωδικού πρόσβασης\n";
	}
	if(document.myform.element_15.value!=document.myform.element_12.value
)
	{
		alert("Οι κωδικοί πρόσβασης δεν είναι ίδιοι. Παρακαλώ εισάγετε ξανά
τους κωδικούς!");
		return false;
	}
	if (themessage == "Η συμπλήρωση των παρακάτω στοιχείων είναι
υποχρεωτική:\n") {
		document.myform.submit();
	}
	else {
		alert(themessage);
		return false;
	}
}
```

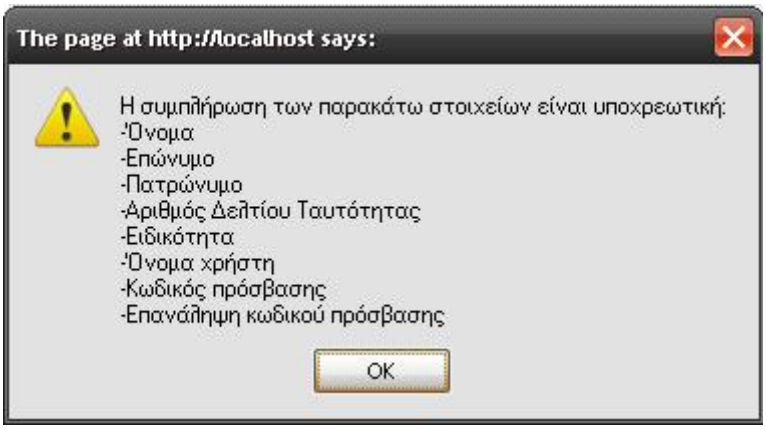
Η συγκεκριμένη συνάρτηση περιέχει και έλεγχο των πεδίων για τον κωδικό πρόσβασης. Γίνεται έλεγχος των πεδίων κωδικός πρόσβασης και επιβεβαίωση

κωδικού πρόσβασης και αν δεν συμπίπτουν επιστρέφεται ένα μήνυμα ότι οι κωδικοί δεν είναι ίδιοι. Αυτό γίνεται έτσι ώστε ο κωδικός πρόσβασης να είναι σίγουρα ο επιθυμητός και να μην έχει γίνει κάποιο λάθος κατά την πληκτρολόγηση.

Ακολουθούν παραδείγματα μηνυμάτων σχετικά με κενά πεδία και σχετικά με κωδικούς πρόσβασης οι οποίοι δεν συμπίπτουν μεταξύ τους.



Μήνυμα σφάλματος όταν δύο password δεν είναι ίδια.



Μήνυμα λάθους όταν υπάρχουν κενά πεδία των οποίων η συμπληρωση είναι υποχρεωτική

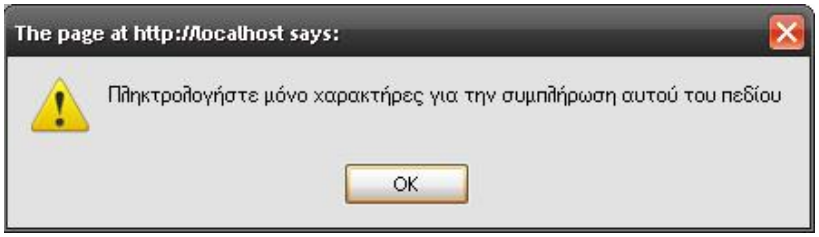
Οι συγκεκριμένες συναρτήσεις ενεργοποιούνται όταν ο χρήστης κάνει υποβολή της φόρμας. Έπειτα για κάθε φόρμα καλείται κάποια συγκεκριμένη συνάρτηση. Αυτό δηλώνεται μέσα στην φόρμα. Πιο συγκεκριμένα για τον γιατρό η φόρμα έχει για event onsubmit=”return verify_doctor();”

Εκτός από τις συναρτήσεις οι οποίες είναι για τα κενά μέσα στο verify_form.js υπάρχουν και τέσσερις ακόμη συναρτήσεις οι οποίες είναι υπεύθυνες για την συμπλήρωση των πεδίων στις φόρμες με τους κατάλληλους χαρακτήρες. Έτσι είναι αδύνατη η εισαγωγή αριθμών για όνομα. Στη συνάρτηση

ορίζεται ένα εύρος χαρακτήρων οι οποίοι είναι αποδεκτοί για την εισαγωγή στο κατάλληλο πεδίο. Αν ο χρήστης πληκτρολογήσει μη αποδεκτούς χαρακτήρες τότε το πεδίο αδειάζει και ο κέρσορας μεταφέρεται και πάλι στο συγκεκριμένο πεδίο. Η συνάρτηση ενεργοποιείται όταν ο χρήστης προσπαθήσει να εισάγει δεδομένα στο κάποιο πεδίο. Πιο συγκεκριμένα σε κάθε πεδίο της φόρμας έχει για event: onBlur="validateCharacter(this)". Αυτό καλεί την συνάρτηση validateCharacter() η όποια θέτει σαν αποδεκτό στοιχείο κάποιο χαρακτήρα. Ακολουθεί κώδικας της validateCharacter().

```
< function validateCharacter(field) {
        var valid =
"άέήίόύώΆΈΉΊΌΎΏαβγδεζηθικλμνξοπρστυφχψωΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣ
ΤΥΦΧΨΩabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXY
Z"
        var ok = "yes";
        var temp;
                for (var i=0; i<field.value.length; i++) {
                                temp = "" + field.value.substring(i, i+1);
                                if (valid.indexOf(temp) == "-1") ok = "no";
                }
                if (ok == "no") {
                                alert("Πληκτρολογήστε μόνο χαρακτήρες για την
συμπλήρωση αυτού του πεδίου");
                                field.focus();
                                field.select();
                                field.value="";
           }
}
```

Οι υπόλοιπες συναρτήσεις με ίδιο σκοπό κάνουν έλεγχο για το αν αυτό που συμπληρώνεται είναι ακέραιος, δεκαδικός, χαρακτήρας ή email. Ακολουθεί ένα μήνυμα σφάλματος για λάθος πληκτρολόγηση στην εισαγωγή στοιχείων.



4.1.3 Αρχεια apli_buttons_actions.js και proigmeni_buttons_actions.js

Έπειτα από την αναζήτηση (απλή ή και προηγμένη) κάποιου ασθενή ή γιατρού παίρνουμε μια σειρά από αποτελεσμέτα ανάλογα με τα κριτήρια της αναζήτησης. Τα απότελέσματα που παίρνουμε έιναι μία διαφορετική φόρμα ανά γραμμή. Στο τέλος κάθε φόρμας αντί να έχουμε ένα κουμπί υποβολής το οποίο να κάνει μία συγκεκριμένη ενέργεια έχουμε περισσότερα. Σχετική εικόνα φαίνεται παρακάτω

Δίπλα από την εγγραφή που παρουσιάζεται υπάρχουν τρία εικονίδια. Το πρώτο προβάλει τα στοιχεία του γιατρού, το δεύτερο επεξεργάζεται τα στοιχεία του γιατρού και το τρίτο διαγράφει το γιατρό. Τις λειτουργίες αυτές τις παρουσιάζουμε στη σχετική ενότητα. Ο κώδικας για την φόρμα παρουσίαζεται και εξηγείται παρακάτω.

```
$i=1;
while($result = mysql_fetch_array( $data )){
        echo '<form id="'.$i.'" name="'.$i.'" method="post" >';
        echo '<input name="d_id" type="hidden" value="'.$result['d_id'].'">';
        echo '<input name="onoma" id="edit" type="text" value="'.$result['onoma'].'"
        size="10" disabled>';
        echo '<input name="epwnimo" id="edit" type="text" value="'.$result['epwnimo'].'"
        size="15" disabled>';
        echo '<input name="patrwnimo" id="edit" type="text"
        value="'.$result['patrwnimo'].'" size="10" disabled>';
        echo '<input name="adt"  type="text" id="edit" value="'.$result['adt'].'" size="10"
        disabled>';
        echo '<input type="image" src="../../includes/view.png" align="absmiddle"
        onclick="return d_provoli('.$i.');">';
        echo '<input type="image" src="../../includes/edit.png" align="absmiddle"
        onclick="return d_allagi('.$i.');">';
        echo '<input type="image" src="../../includes/delete.png" align="absmiddle"
        onclick="return d_diagrafi('.$i.');">';
        $i = $i + 1;
        echo '</form>';
}
```

Όπως φαίνεται και στον κώδικα κάθε φόρμα παίρνει το id μέσω κάποιου μετρητή. Ο μετρητής αυτός $i ξεκινά από την τιμή 1, αυτό γίνεται διότι υπάρχει ακόμη μία φόρμα πιο πάνω. Στο στοιχείο του DOM document.forms[ ] δεν μας ενδιαφέρει η πάνω φόρμα, δηλαδή η document.forms[0] γι αυτό και σαν πρώτη τιμή του μετρητή είναι η 1. Τα κουμπιά που είναι στη φόρμα δεν είναι τύπου submit αλλά είναι εικονίδια τα οποία αν τα κάνουμε κλικ καλούν κάποια συγκεκριμένη συνάρτηση έτσι ώστε να εκτελέσουν την σχετική τους λειτουργία. Ακολουθεί τμήμα κώδικα συνάρτησης javascript. Τα παραδείγματα της ενότητας αφορούν τον γιατρό.

```
function d_provoli(id){

   var agree=confirm("Είστε σίγουρος ότι θέλετε να συνεχίσετε;");
      if (agree){
         document.forms[id].action="../../provoli/Provoli
         Giatrou/form_view_doctor.php";
         document.forms[id].submit();
         return true;
      }
      else
         return false ;
}
```

Η συνάρτηση εφόσον επιβεβαιώσει την πρόθεση του χρήστη θέτει σαν action το κατάλληλο path και κάνει υποβολή την φόρμα. Παραπάνω φαίνεται η συνάρτηση σχετικά με την προβολή στοιχείων γιατρού έπειτα από αναζήτηση. Οι ίδιες ακριβως συναστησεις είναι και στην προηγμένη αναζήτηση με διαφορετικό path λόγω ιεραρχίας των φακέλων.

4.1.4 Αρχείο checkExams.js

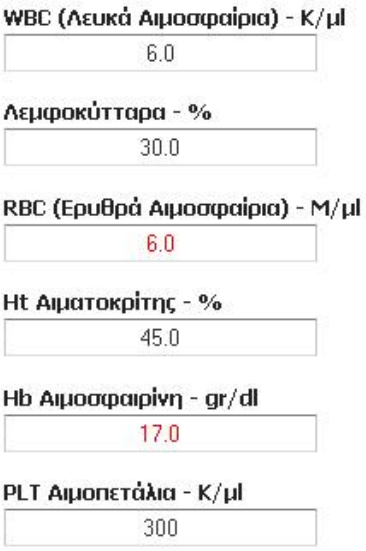
Το αρχείο αυτό χρησιμοποιείται κατά την προβολή των εργαστηριακών εξετάσεων του ασθενούς. Όταν κάποια τιμή κάποιας μέτρησης δεν είναι φυσιολογική τιμή τότε το χρώμα του κειμένου της συγκεκριμένης μέτρησης γίνεται κόκκινο και τονίζεται σαν bold. Με αυτό τον τρόπο από τη μία τονίζεται κάποιο πρόβλημα που ίσως αντιμετωπίζει ο ασθενής και από την άλλη κάνει πιο εύκολη

του δουλειά του γιατρού έτσι ώστε να μην χρειάζεται να συγκρίνει τιμές μετρλησεων με τις φυσιολογικές τιμές. Η φόρμα της προβολής εργαστηριακών εξετάσεων εξηγούνται σε σχετική ενότητα της τεκμηρίωσης. Ακολουθεί κώδικας σχετικά με το πώς επιτυγχάνεται αυτό και αφορά την αιματολογική εξέταση.

```
function aimatologiki_values(){

	if(document.getElementById("element_1").value < 4 ||
document.getElementById("element_1").value > 11 ){
		document.getElementById("element_1").style.color='red';
		document.getElementById("element_1").fontWeight='bold';
	}

	if(document.getElementById("element_2").value < 20 ||
document.getElementById("element_2").value > 45 ){
		document.getElementById("element_2").style.color='red';
		document.getElementById("element_2").fontWeight='bold';
	}

	if(document.getElementById("element_3").value < 4 ||
document.getElementById("element_3").value > 5.6 ){
		document.getElementById("element_3").style.color='red';
		document.getElementById("element_3").fontWeight='bold';
	}

	if(document.getElementById("element_4").value < 37 ||
document.getElementById("element_4").value > 47 ){
		document.getElementById("element_4").style.color='red';
		document.getElementById("element_4").fontWeight='bold';
	}

	if(document.getElementById("element_5").value < 11.8 ||
document.getElementById("element_5").value > 16.8 ){
		document.getElementById("element_5").style.color='red';
		document.getElementById("element_5").fontWeight='bold';
	}

	if(document.getElementById("element_6").value < 150 ||
document.getElementById("element_6").value > 400 ){
		document.getElementById("element_6").style.color='red';
		document.getElementById("element_6").fontWeight='bold';
	}
}
```

Ο κώδικας παίρνει κάθε στοιχείο της φόρμας μέσω της μεθόδου getElementById(), κάνει έλεγχο της τιμής του πεδίου και αν ικανοποιεί την συνθήκη τότε οι

χαρακτήρες γίνονται κόκκινοι. Παρακάτω παρουσιάζεται το αποτέλεσμα αυτής της διαδικασίας. Παράδειγμα μας είναι πάντα η αιματολογική εξέταση.



Με παρόμοιο τρόπο η μέθοδος λειτουργεί και για τις άλλες εργαστηριακές εξετάσεις. Σχετικά με την προβολή των εξετάσεων υπάρχει αναλυτική περιγραφή σε σχετική ενότητα.

4.2 CSS (Cascading Style Sheets)

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκαν πέντε διαφορετικά external css κάθε ένα για διαφορετικό λόγο. Αυτά τα αρχεία είναι: sdmenu.css, view.css, view_confirm.css, view_main.css, view_results.css.

Το sdmenu.css χρησιμοποιείται για την μορφοποίηση του μενού επιλογών που βρίσκεται κάθετα στα αριστερά στην εφαρμογή. Το μενού χωρίζεται σε κατηγορίες οι οποίες έχουν σαν φόντο ένα γκρι χρώμα το οποίο είναι το αρχειο title.gif του φακέλου includes. Επίσης χρησιμοποιούνται και τα εικονίδια collapsed.gif και expanded.gif για να αναδεικνύεται πιο όμορφα μια ανεπτυγμένη κατηγορία του κύριου μενού.

Για την σωστή λειτουργία του κύριου μενού χρησιμοποιείται και το sdmenu.js. Είναι ένα javascript αρχείο το οποίο είναι υπεύθυνο για την κίνηση του

μενού και την εμφάνιση και απόκρυψη των στοιχειών κάθε κατηγορίας. Το Μενού δεν σχεδιάστηκε από μας, βρέθηκε στο dynamic Drive. Τροποποιήθηκε όμως από εμάς έτσι ώστε να το κάνουμε πιο ευέλικτο και ταιριαστό στην εφαρμογή της διαχείρισης ιστορικού ασθενών.

Ένα άλλο αρχείο χρήσιμο είναι το view.css. Αυτό είναι υπεύθυνο σχετικά με τη μορφοποίηση της φόρμας. Όλες οι φόρμες χρησιμοποιούν αυτό το css. Βάζει ετικέτες πάνω από κάθε πεδίο και περιγραφή από κάτω. Όταν ο χρήστης περάσει με το ποντίκι του, πάνω από κάποιο πεδίο τότε εμφανίζεται ένα σχετικό μήνυμα από δεξιά. Αυτό μπορεί να εμφανίζει σχετικές πληροφορίες με το πεδίο της φόρμας το οποίο διερχόμαστε. Σχετικά με τις φόρμες υπάρχει αναλυτική περιγραφή στη σχετική ενότητα.

Για την μορφοποίηση του layout της εφαρμογής χρησιμοποιήθηκε το view_main.css. Το αρχείο αυτό θέτει το φόντο, το μέγεθος του πίνακα, το χρώμα γεμίσματος κάθε κελιού και τις μορφοποιήσεις κάθε επικεφαλίδας και κάθε παραγράφου. Άλλο ένα αρχείο μορφοποίησης είναι το view_results.css το οποίο χρησιμοποιήθηκε για τη μορφοποίηση κάθε σελίδας η όποια εμφανίζεται έπειτα από κάποια αναζήτηση. Τροποποιεί κάθε κελί στα χρώματα που χρησιμοποιούνται αλλά και στη γραμματοσειρά. Τέλος ακόμη ένα αρχείο είναι το view_confirm.css

Θα πρέπει να επισημανθεί ότι σελίδες στο διαδίκτυο μπορεί να αναπαρίστανται διαφορετικά από ένα φυλλομετρητή σε κάποιον άλλο. Η εφαρμογή έχει σχεδιαστεί να δουλεύει όσο το δυνατόν καλύτερα σε Mozilla Firefox. Δεν προορίζεται για χρήστες διαδικτύου οπότε μπορούμε να ελέγξουμε ότι οι χρήστες θα έχουν ως πρόγραμμα περιήγησης τον Firefox για την αναπαράσταση της εφαρμογής. Έτσι δεν μας ενοχλεί να προτείνουμε σαν προτεινόμενο φυλλομετρητη τον Firefox.

4.3 Τεκμηρίωση PHP-HTML

4.3.1 HTML-Διάταξη αρχικής σελίδας

Μετά από μια επιτυχή σύνδεση του χρήστη είτε αυτός είναι διαχειριστής είτε ιατρός στον περιηγητή εμφανίζεται η αρχική σελίδα. Η αρχική σελίδα είναι ένας πίνακας. Αποτελείται από τρεις γραμμές και δύο στήλες. Η πρώτη και η τρίτη γραμμή καταλαμβάνουν δύο στήλες.

Στη πρώτη γραμμή είναι η επικεφαλίδα (header) της εφαρμογής. Εκεί περιέχεται ο τίτλος της εφαρμογής «Διαχείριση Ιστορικού Ασθενών» και στην πάνω δεξιά γωνία υπάρχουν δύο κουμπιά. Το ένα επιστρέφει τον χρήστη στην αρχική σελίδα και το δεύτερο κάνει αποσύνδεση του χρήστη από το σύστημα.

Στην δεύτερη γραμμή και πρώτη στήλη υπάρχει το κύριο μενού της εφαρμογής. Ανάλογα με τον τύπο χρήστη της εφαρμογής οι επιλογές είναι και διαφορετικές. Οι δυνατότητες του κάθε χρήστη περιγράφονται στο κεφάλαιο 1. Επισημάνεται και πάλι ότι το μενού είναι από το dynamic drive και μπορεί να το κατεβάσει οποιοσδήποτε. Επεξεργάστηκε έτσι ώστε να είναι πιο ταιριαστό στην εφαρμογή μας.

Στη δεύτερη γραμμή και δεύτερη στήλη υπάρχει ένα iframe, το οποίο καταλαμβάνει όλο το κελί. Όλες οι λειτουργίες της εφαρμογής οι οποίες καλούνται από το κύριο μενού φορτώνονται στο iframe. Όταν γίνεται φόρτωση της ιστοσελίδας στο iframe φορτώνεται το welcome.php. Αυτό το αρχείο παρουσιάζει το ονοματεπώνυμο και τον τύπο του και περιέχει και ένα ρολόι με ημερομηνία το οποίο το παίρνει από την ώρα του υπολογιστή. Τέλος στη τρίτη γραμμή (footer) υπάρχουν τα στοιχεία των δημιουργών της εφαρμογής. Το iframe αν είναι τόσο μικρό ώστε να μην φαίνεται ολόκληρο το περιεχόμενο εμφανίζει ένα scroll bar για την καλύτερη απεικόνιση. Το όνομα του iframe είναι main και όλα τα link της εφαρμογής έχουν σαν target το frame με όνομα main.

Η διάσταση του πίνακα καλύπτει το 95% του παραθύρου σε πλάτος και 600 pixel σε ύψος. Από αυτό το μέγεθος το 10% του ύψους και το 100% του πλάτους

το καλύπτει η πρώτη γραμμή, το 80% του ύψους και 20% του πλάτους είναι για το κυρίο μενού. Το 80% του ύψους και 80% του πλάτους έιναι για το iframe ενώ τέλος το 10% του ύψους και 100% του πλάτους για το footer.

4.3.2 HTML - Φόρμες

Στην συγκεκριμένη ενότητα αναλύονται και παρουσιάζονται τμήματα κώδικα που αφορούν της φόρμες εισαγωγής στοιχείων. Είναι μια γενική περιγραφή της φόρμας που χρησιμοποιείται. Η ανάλυση αφορά τα στοιχεία από τα οποία αποτελείται μια φόρμα και το πώς αυτή λειτουργεί. Οι λειτουργίες της εφαρμογής αναλύονται ξεχωριστά.

Παντού μέσα στην εφαρμογή υπάρχουν html φόρμες. Φόρμα είναι η σελίδα στην οποία κάνουμε σύνδεση, η δημιουργία κάποιας από τις οντότητες και γενικότερα όπου υπάρχει κάποιο πεδίο εισαγωγής κάποιας πληροφορίας προς το σύστημα. Το πεδίο εισαγωγής στην εφαρμογή μας μπορεί να είναι πλαίσιο, radio button ή drop down list και textarea.

Κάθε μία φόρμα ξεκινά με την χρήση της ετικέτας <form> και κλείνει αντίστοιχα με </form>. Για την δημιουργία κάποιου ιατρού η ετικέτα της φόρμας είναι η ακόλουθη:

```
<form
        name="myform"
        id="form_20073"
        class="appnitro"
        method="post"
        action="create_doctor.php"
        onsubmit="return verify_doctor();"
>
```

Για λόγους ευκολίας όλες οι φόρμες έχουν όνομα myform και ίδιο id. Το ονομα και το id χρησιμοποιούνται για το HTML DOM και για τις συναρτήσεις javascript. Η οδηγία class=”appnitro” καθορίζει το css που θα χρησιμοποιηθεί για την μορφοποίηση της φόρμας. Η οδηγία method=”post” καθορίζει τον τρόπο με τον οποίο θα αποσταλούν τα δεδομένα και θα χειριστούν από την php. Το

action="create_doctor.php" καθορίζει το αρχείο το οποίο θα κληθεί να δεχτεί τα δεδομένα της φόρμας όταν γίνει η υποβολή της. Με την υποβολή της φόρμας το iframe φορτώνει το create_doctor.php και γίνεται καταχώρηση των στοιχείων. Τέλος κατά την υποβολή της φόρμας καλείται η συνάρτηση verify_doctor(), η οποία υπάρχει στο αρχείο verify_form. Η συνάρτηση αυτή ελέγχει την φόρμα για το αν υπάρχουν άδεια υποχρεωτικά πεδία. Σχετικά με τις συναρτήσεις του verify_form υπάρχει σχετική ενότητα στο κομμάτι με τα javascript.

Για την μορφοποίηση της φόρμας χρησιμοποιείται το στοιχείο και . Το στοιχείο div κάνει ομαδοποίηση των στοιχείων που υπάρχουν ανάμεσα και προκαλεί αλλαγή γραμμής εκεί που κλείνει η ετικέτα. Το στοιχείο span κάνει και αυτό ομαδοποίηση του περιεχομένου του αλλά δεν προκαλεί αλλαγή γραμμής. Τα div και span είναι πια ο πιο σύγχρονος ευέλικτος και αξιόπιστος τρόπος για την μορφοποίηση μιας σελίδας με τη βοήθεια των cascading style sheets

Όλα τα στοιχεία της φόρμας βρίσκονται μέσα σε μια λίστα. Κάθε ένα στοιχείο της φόρμας αποτελεί και ξεχωριστό στοιχείο της λίστας. Μια html λίστα ξεκινά με την ετικέτα <ul> και τελειώνει με το , κάθε στοιχείο της λίστας ξεκινά με την ετικέτα <li> και τελειωνει με την ετικέτα . Κάθε ένα στοιχείο της λίστας έχει και το δικό του id, δηλαδή <li id="li_1" >.

Παρακάτω παρουσιάζεται τμήμα κώδικα με τις ετικέτες που περιγράφηκαν μέχρι τώρα. Αποτελεί το πεδίο συμπλήρωσης του πατρώνυμου.

```
<li id="li_2" >
  <label class="description" for="element_2">(*) Πατρώνυμο </label>
  <div>
    <input
        id="element_2"
        name="element_2"
        class="element text medium"
        type="text"
        maxlength="255"
        value=""
        onBlur="validateCharacter(this)"/>
  </div>
  <p class="guidelines" id="guide_2">
        <small>Εισάγετε το πατρώνυμο.</small>
  </p>
</li>
```

Κάθε στοιχείο της φόρμας όπως αναφέρθηκε και στοιχείο μιας λίστας. Με την ετικέτα <label> και το κατάλληλο css έχουμε ως αποτέλεσμα να εμφανίζεται πάνω από το πεδίο μια επικεφαλίδα με την κατάλληλη περιγραφή. Το input πεδίο βρίσκεται μέσα σε div ετικέτες. Με τις ετικέτες <p> και το κατάλληλο css class="guidelines" έχουμε σαν αποτέλεσμα όταν κάνουμε rollover πάνω από κάποιο πεδίο να εμφανίζεται κάποιο σχετικό μήνυμα βοήθειας σχετικά με το συγκεκριμένο πεδίο. Ακολουθεί το αποτέλεσμα του παραπάνω κώδικα.

Σε περίπτωση που το πεδίο της λίστας αποτελείται από παραπάνω τους ενός στοιχείου τότε υπάρχει και μία περιγραφή κάτω από το κάθε πεδίο. Αυτό επιτυγχάνεται με την οδηγία <label for="element_1_1">Όνομα</label>. Παρακάτω παρουσιάζεται το αποτέλεσμα αυτής της διαδικασίας σχετικα με το ονοματεπώνυμο το οποίο αποτελείται από δύο ξεχωριστά πεδία το όνομα και το επώνυμο.

Κάθε ένα στοιχείο της φόρμας έχει ένα name. Το όνομα αυτό είναι μοναδικό για τη φόρμα και χρησιμοποιείται έτσι ώστε να πάρουμε το πεδίο και με τη μέθοδο post της php. Έστω για παράδειγμα ότι έχουμε το πεδίο με name=”element_1” για να πάρουμε αυτό το πεδίο στην php και να το εκχωρήσουμε σε μια μεταβλητή θα πρέπει να γράψουμε $var=$_POST[element_1]. Έχοντας διαφορετικά ονόματα στα πεδία μπορούμε να τα προσδιορίσουμε μοναδικά και να τα χειριστούμε κατάλληλα στο php κομμάτι της εφαρμογής.

Όπως αναφέρθηκε και παραπάνω μπορούμε να περάσουμε πληροφορίες με χρήση πεδίου εισαγωγής χαρακτήρων, radio buttons, drop down list και textarea.

Για να εισάγουμε χαρακτήρες στη φόρμα μπορούμε κάνοντας χρήση του ακόλουθου κώδικα:

```
<input
        id="element_1"
        name="element_1"
        class="element text medium"
        type="text"
        maxlength="25"
        value=""
        onBlur="validateCharacter(this)"
/>
```

Έτσι έχουμε ένα πεδίο όπως αυτό στη φόρμα με το όνομα. Το όνομα και το id του πεδίου είναι μοναδικά για κάθε στοιχείο. Ο τύπος του πεδίου καθορίζεται με την οδηγία type="text". Ο χρήστης μπορεί να εισάγει ως και 25 χαρακτηρες, το πεδίο δεν έχει κάποια τιμή συμπληρωμένη αφού η οδηγία value είναι κενή. Τέλος για το πεδίο καλείται η συνάρτηση validateCharacter() έτσι ώστε να μην μπορει ο χρήστης να βάλει αριθμους. Για τις συγκεκριμένες συναρτήσεις έγινε αναφορά στο κομμάτι της javascript.

Στην εφαρμογή χρησιμοποιούνται radio buttons για την υποβολή του φύλου. Για τη δημιουργία radio buttons ο κώδικας που χρειάζεται είναι ο παρακάτω:

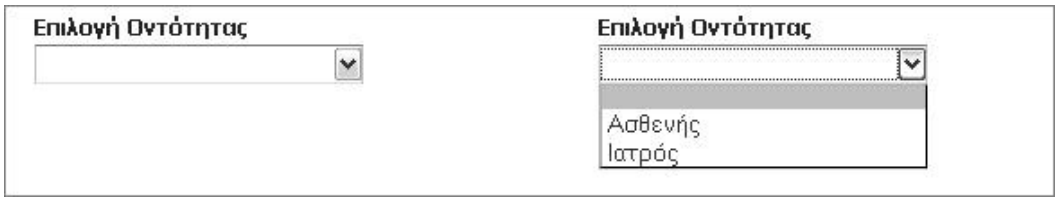
```
<input
        id="element_13_1"
        name="element_13"
        class="element radio"
        type="radio"
        value="Άντρας"
/>
```

Οι οδηγίες σχετικά με το input είναι ίδιες με το προηγούμενο πεδίο οπότε δεν υπάρχει λόγος να επισημανθεί κάτι. Το μόνο που πρέπει να προσέξουμε είναι ότι το type="radio" και το value="Άντρας". Έτσι δημιουργείται ένα radio box που δίπλα του γράφει την λέξη Άντρας. Ακολουθεί ένα παράδειγμα με το αποτέλεσμα της φόρμας.

Υπάρχουν αρκετά drop down list σε διαφορες φόρμες. Για να δημιουργήσουμε ένα drop down list πρέπει ο κώδικας μας να είναι ο παρακάτω.

```
<Select name="type" class="element select medium">
        <Option VALUE="" selected="selected"></option>
        <Option VALUE="patient">Ασθενής</option>
        <Option VALUE="doctor">Ιατρός</option>
</Select>
```

Με την ετικέτα select δηλώνουμε ένα drop down list. Κάθε select έχει ένα μοναδικό όνομα και το css με το οποίο συσχετίζεται για την μορφοποίηση του. Το συγκεκριμένο select είναι από την φόρμα της απλής αναζήτησης. Οι επιλογές σε ένα drop down list ορίζονται με την ετικέτα <option>, στην οδηγία value="" βάζουμε την τιμή που θέλουμε να περάσει στη βάση και ανάμεσα από τις ετικέτες βάζουμε αυτό που θέλουμε ο χρήστης να διαβάζει. Αν θέλουμε κάποια συγκεκριμένη επιλογή να είναι προεπιλεγμένη πρέπει να προσθέσουμε σαν οδηγια και το selected="selected". Ακολουθεί το αποτέλεσμα που παίρνουμε από το παραπάνω τμήμα κώδικα.



Τελευταίος τρόπος εισαγωγής στοιχείων στην εφαρμογή είναι με χρήση textarea. Με το textarea μπορούμε να εισάγουμε κειμενο με την διαφορά όμως ότι μπορουμε να βλέπουμε όλο το κειμενο που δίνουμε. Αυτός ο τρόπος χρησιμοποιείται για την καταχώρηση διάγνωσης και αντίστοιχης θεραπείας. Παρουσιάζεται ο κώδικας που απαιτείται για την δημιουργία ενός textarea.

```
<textarea id="element_2"name="element_2" class="element textarea medium">

</textarea>
```

Και πάλι οι οδηγίες που βρίσκονται μέσα στην ετικέτα δεν έχουν καμία διαφορά με τις οδηγίες που περιγράψαμε παραπανω. Ακολουθεί το αποτέλεσμα του παραπάνω κώδικα

Σε όλες τις φόρμες που απαιτούν χρήση ημερομηνίας χρησιμοποιείται ένα javascript ημερολόγιο το οποίο είναι ελεύθερο και μπορεί να το βρει κανείς στο διαδίκτυο. Για τη λειτουργία του είναι απαραίτητη η οδηγία στο head τμήμα του html κώδικα:

```
<script type="text/javascript" src="../../includes/calendar.js"></script>
```

Επίσης είναι απαραίτητο το εικονίδιο calendar.gif από τον φάκελο includes δίνοντας τον κώδικα:

```
<img id="cal_img_4" class="datepicker" src="../../includes/calendar.gif"
```

Το ημερολόγιο κάνει χρήση ενός ακόμη script και ενός css. Η εξήγηση του τρόπου με τον οποίο δουλεύει το ημερολόγιο είναι πέρα από το σκοπό της τεκμηρίωσης της εφαρμογής.

Σχεδόν κάθε φόρμα έχει δυο κουμπιά, ένα για την υποβολή και ένα για να γίνει reset η φόρμα. Για να δημιουργήσουμε αυτά τα κουμπιά πρέπει να εισάγουμε τον παρακάτω κώδικα.

```
<input id="saveForm" class="button_text" type="submit"
name="submit"  value="Υποβολή"/>

<input id="saveForm" class="button_text" type="reset"
name="Reset" value="Reset"/>
```

Το αποτέλεσμα του παραπάνω τμήματος είναι η εικόνα που φαίνεται παρακάτω:

Τέλος για την σκιά πάνω και κάτω από την φόρμα χρησιμοποιούνται δύο εικόνες, top.png και bottom.png. Η μαύρη γραμμή πάνω σε όλες τις φόρμες επιτυγχάνεται με το <h1><a>Δημιουργία Ιατρού</h1> και τη χρήση του κατάλληλου css δηλαδή του view.css.

4.3.3 Αρχείο db_connect.php και db_disconnect.php

Στο τμήμα κώδικα που φαίνεται παρακάτω για να γίνει η σύνδεση με την βάση δεδομένων χρησιμοποιείται η συνάρτηση mysql_connect(). Η συνάρτηση αυτή παίρνει 3 ορίσματα. Το πρώτο όρισμα προσδιορίζει τον mySql Server, ο οποίος είναι στη θέση localhost. Έπειτα ακολουθεί το όνομα χρήστη το οποίο είναι το root και ο κωδικός πρόσβασης ο οποίος είναι το κενό. Τα στοιχεία αυτά σχετικά με τον λογαριασμό είναι τα προκαθορισμένα από το wamp. Σε περίπτωση που δεν καταφέρει να συνδεθεί στη βάση δεδομένων εμφανίζεται το μηνυμα “Could not connect: “ και ακολουθεί το συγκεκριμένο σφάλμα από τον mysql server.

Έπειτα ακολουθεί η συνάρτηση mysql_select_db(). Παίρνει δύο ορίσματα το πρώτο είναι η βάση δεδομένων στην οποία συνδεόμαστε και το δεύτερο όρισμα

είναι η σύνδεση για τον mysql server. Αφού πραγματοποιηθεί η παραπάνω διαδικασία έχουμε μια επιτυχή σύνδεση στη βάση δεδομένων της εφαρμογής.

```
<?php
        $con = mysql_connect("localhost","root","");

        if (!$con) {
                die('Could not connect: ' .
mysql_error());
         }

        mysql_select_db("hospital", $con);
```

Σωστό όμως είναι αφού κάνουμε σύνδεση στον Mysql server και στη βάση δεδομένων να κάνουμε και αποσύνδεση στο τέλος χρήσης της βάσης. Αυτό γίνεται με την συνάρτηση mysql_close() και σαν όρισμα παίρνει την μεταβλητή με τα στοιχεία του mysql server. Παρακάτω φαίνεται το πώς κάνουμε αποσύνδεση.

```
<?php

        mysql_close($con);

?>
```

4.3.4 Δημιουργία Γιατρού

Για να γίνει η δημιουργία ενός γιατρού ο χρήστης (Διαχειριστής) θα πρέπει να πατήσει το σχετικό link από το κύριο μενού. Στην συνέχεια το σύστημα ανταποκρίνεται εμφανίζοντας του την φόρμα εισαγωγής στοιχείων του γιατρού.

Ο χρήστης στην συνέχεια καλείτε να συμπληρώσει αυτή την φόρμα με τα στοιχειά τα οποία χαρακτηρίζουν τον κάθε γιατρό, στην συνέχεια καλείτε να πατήσει το πλήκτρο υποβολή ώστε να καταχωρηθούν τα δεδομένα στην βάση δεδομένων, σε περίπτωση που ο χρήστης δεν έχει συμπλήρωση όλα τα απαραίτητα στοιχειά εμφανίζεται μήνυμα το οποίο τον πληροφορεί για το πια

στοιχεία είναι απαραίτητα να συμπληρωθούν ώστε να μπορέσει να γίνει η υποβολή της φόρμας.

Για τη διαδικασία που περιγράφεται παραπάνω χρησιμοποιούνται τα ακόλουθα αρχεία με τη σειρά που εκτελούνται: form_create_doctor.php και create_doctor.php.

Η φόρμα εισαγωγής περιέχει διάφορα στοιχεία τα οποία αφορούν τον γιατρό, εδώ αναφέρουμε τα απαραίτητα στοιχεία τα οποία πρέπει να συμπληρωθούν για την υποβολή της φόρμας τα οποία είναι τα εξής: Όνομα, Επώνυμο, Πατρώνυμο, Αριθμός δελτίου ταυτότητας, Ειδικότητα, όνομα χρήστη και κωδικό πρόσβασης.

Με το πάτημα του πλήκτρου υποβολή, στέλνονται στο αρχείο create_doctor.php όλα τα δεδομένα της φόρμας τα οποία αφού φιλτραριστούν από συναρτήσεις (trim(), md5() και άλλες) που έχουμε αναφέρει χωριστά σε άλλο κομμάτι της τεκμηρίωσης χρησιμοποιούνται για να εισάγουμε τα δεδομένα στην βάση δεδομένων αφού έχουν περαστεί πρώτα σε μεταβλητές.

Παρακάτω αναφέρονται μερικά από τα στοιχεία τα οποία δέχεται το αρχείο create_doctor.php όπως φαίνονται στο κομμάτι κώδικα το οποίο περιέχει αυτό το αρχείο.

```
$username= mysql_real_escape_string( $_POST['element_11'] );
$eid_id= mysql_real_escape_string( $_POST['element_14'] );
$onoma= mysql_real_escape_string( $_POST['element_1_1'] );
$epwnimo= mysql_real_escape_string( $_POST['element_1_2'];
```

```
$patrwnimo= mysql_real_escape_string( $_POST['element_2'] );
$adt= mysql_real_escape_string( $_POST['element_3'] );
$password = mysql_real_escape_string( $_POST['element_12'] );
$password = md5($password);
```

Οι μεταβλητές αυτές είναι απαραίτητες για την συμπλήρωση του πίνακα ο οποίος περιέχει τα στοιχεία των γιατρών και των χρηστών. Μάλιστα εδώ θα πρέπει να αναφέρουμε ότι κατά την δημιουργία ενός γιατρού δημιουργείτε και ένας χρήστης, τα στοιχεία του χρηστή καταχωρούνται στον πίνακα user ο οποίος συσχετίζεται με τον πίνακα γιατρός, έτσι κάθε γιατρός έχει ένα μοναδικό όνομα χρήστη ώστε να μπορεί να συνδέεται στο σύστημα και να κάνει τις κατάλληλες εργασίες για τις οποίες έχει δικαίωμα . Παρακάτω παρουσιάζεται το τμήμα του sql κώδικα που κάνει εισαγωγή των στοιχείων στη βάση δεδομένων για τον πίνακα user.

```
$sql_1="INSERT INTO user (username, password, level )
VALUES
        ('$username','$password','1')";
```

Στον πίνακα user παρατηρούμε το χαρακτηριστικό level το οποίο εισάγουμε στην βάση, αυτό χαρακτηρίζει τα δικαιώματα τα οποία θα έχει ο κάθε χρήστης στο σύστημα μετά την σύνδεση του, τα επίπεδα είναι 0, 1 και 2 , το επίπεδο 0 χαρακτηρίζει έναν γιατρό ο οποίος έχει δημιουργηθεί και είναι ανενεργός στο σύστημα (δεν μπορεί να συνδεθεί αλλά παραμένει γιατί έχει πραγματοποιήσει στο παρελθόν κάποιες εξετάσεις-διαγνώσεις ώστε να διατηρείτε ακέραιο το ιστορικό), το επίπεδο 1 χαρακτηρίζει έναν γιατρό ο οποίος μπορεί να εκτελέσει όλες τις ενέργειες τις οποίες έχει δικαίωμα να εκτελέσει ένας γιατρός και το επίπεδο 2 είναι το επίπεδο διαχειριστή συστήματος ο οποίος έχει πλήρη δικαιώματα στο σύστημα.

Παρακάτω παρουσιάζεται το τμήμα του sql κώδικα που κάνει εισαγωγή των στοιχείων στη βάση δεδομένων για τον πίνακα doctor.

```
$sql_2="INSERT INTO doctor
(d_id, username, eid_id, onoma, epwnimo, patrwnimo, adt, filo,
imer_gen, dieuthinsi, poli, xwra, tk, til1, til2, kinito, email)
VALUES (
NULL, '$username', '$eid_id', '$onoma', '$epwnimo', '$patrwnimo',
'$adt', '$filo', '$imer_gen', '$dieuthinsi', '$poli', '$xwra', '$tk', '$til1', '$til2',
'$kinito', '$email')";
```

Αφού γίνει η εισαγωγή των στοιχείων στην βάση και μετά την καταχώρηση τους εμφανίζεται μήνυμα το οποίο πληροφορεί τον χρηστή ότι έχει γίνει η δημιουργία επιτυχώς και ότι ο γιατρός έχει καταχωρηθεί στο σύστημα.

4.3.5 Δημιουργία Ασθενούς

Ο τρόπος με τον οποίο γίνεται η δημιουργία ενός ασθενή είναι ίδιος με αυτόν της δημιουργίας ενός γιατρού ο οποίος αναφέρθηκε παραπάνω.

Ο χρήστης (Διαχειριστής η γιατρός) θα πρέπει να πατήσει το σχετικό link από το κύριο μενού. Στην συνέχεια το σύστημα ανταποκρίνεται εμφανίζοντας του την φόρμα εισαγωγής στοιχείων του ασθενή.

Ο χρήστης στην συνέχεια καλείτε να συμπληρώσει αυτή την φόρμα με τα στοιχειά τα οποία χαρακτηρίζουν τον κάθε ασθενή, στην συνέχεια καλείτε να πατήσει το πλήκτρο υποβολή ώστε να καταχωρηθούν τα δεδομένα στην βάση δεδομένων, σε περίπτωση που ο χρήστης δεν έχει συμπλήρωση όλα τα απαραίτητα στοιχειά εμφανίζεται μήνυμα το οποίο τον πληροφορεί για το πια στοιχεία είναι απαραίτητα να συμπληρωθούν ώστε να μπορέσει να γίνει η υποβολή της φόρμας , όπως ακριβώς γίνεται και στην δημιουργία ενός γιατρού.

Για τη διαδικασία που περιγράφεται παραπάνω χρησιμοποιούνται τα ακόλουθα αρχεία με τη σειρά που εκτελούνται: form_create_patient.php και create_patient.php.

Η φόρμα εισαγωγής περιέχει διάφορα στοιχεία τα οποία αφορούν τον ασθενή, εδώ αναφέρουμε τα απαραίτητα στοιχεία τα οποία πρέπει να συμπληρωθούν για την υποβολή της φόρμας τα οποία είναι τα εξής: Όνομα, Επώνυμο, Πατρώνυμο, Αριθμός δελτίου ταυτότητας και Ταμείο.

Παρακάτω αναφέρονται μερικά από τα στοιχεία τα οποία δέχεται το αρχείο create_patient.php μετα την υποβολη της φορμας, όπως φαίνονται στο κομμάτι κώδικα το οποίο περιέχει αυτό το αρχείο.

```
$onoma= mysql_real_escape_string( $_POST['element_1_1'] );
$epwnimo= mysql_real_escape_string( $_POST['element_1_2'] );
$patrwnimo= mysql_real_escape_string( $_POST['element_2'] );
$adt= mysql_real_escape_string( $_POST['element_3'] );
```

Παρακάτω παρουσιάζεται το τμήμα του sql κώδικα που κάνει εισαγωγή των στοιχείων στη βάση δεδομένων για τον πίνακα patient.

```
$sql="INSERT INTO patient
(p_id, onoma, epwnimo, patrwnimo, adt, filo, imer_gen, dieuthinsi, poli, xwra, tk,
tam_id, imer_egg, til1, til2, kinito, email)
VALUES
( NULL, '$onoma', '$epwnimo', '$patrwnimo', '$adt', '$filo', '$imer_gen', '$dieuthinsi',
'$poli', '$xwra', '$tk', '$tam_id', '$imer_egg', '$til1', '$til2',          '$kinito', '$email')";
```

Αφού γίνει η εισαγωγή των στοιχείων στην βάση και μετά την καταχώρηση τους εμφανίζεται μήνυμα το οποίο πληροφορεί τον χρηστή ότι έχει γίνει η δημιουργία επιτυχώς και ότι ο ασθενής έχει καταχωρηθεί στο σύστημα.

4.3.6 Δημιουργία Ειδικότητας

Για την δημιουργία μιας ειδικότητας ο χρήστης (Διαχειριστής) θα πρέπει να πατήσει το σχετικό link από το κύριο μενού. Στην συνέχεια το σύστημα ανταποκρίνεται εμφανίζοντας του την φόρμα εισαγωγής νέας Ιατρικής ειδικότητας.

Ο χρήστης στην συνέχεια καλείτε να συμπληρώσει αυτή την φόρμα με μοναδικό στοιχειό το όνομα της ειδικότητας και να πατήσει το πλήκτρο υποβολή.

Για τη διαδικασία που περιγράφεται παραπάνω χρησιμοποιούνται τα ακόλουθα αρχεία με τη σειρά που εκτελούνται: form_create_eidikotita.php και create_eidikotita.php.

Το αρχείο create_eidikotita.php δέχεται στην συνέχεια από την φόρμα το όνομα τις ειδικότητας και αφού κάνει τις κατάλληλες ενέργειες το εισάγει στην βάση δεδομένων όπως φαίνεται στο παρακάτω κομμάτι κώδικα:

```
$kat=mysql_real_escape_string( $_POST['element_1'] );
```

Παρακάτω παρουσιάζεται το τμήμα του sql κώδικα που κάνει εισαγωγή του στοιχείου στη βάση δεδομένων:

```
$sql="INSERT INTO eidikotita (e_id, kat)
VALUES (NULL, '$kat')";
```

Αφού γίνει η εισαγωγή του στοιχείου στην βάση και μετά την καταχώρηση του εμφανίζεται μήνυμα το οποίο πληροφορεί τον χρηστή ότι έχει γίνει η δημιουργία επιτυχώς και ότι η ειδικότητα έχει καταχωρηθεί στο σύστημα.

4.3.7 Δημιουργία Ταμείου

Για την δημιουργία ενός ταμείου ο χρήστης (Διαχειριστής η γιατρός) θα πρέπει να πατήσει το σχετικό link από το κύριο μενού. Στην συνέχεια το σύστημα ανταποκρίνεται εμφανίζοντας του την φόρμα εισαγωγής νέου Ταμείου ασθενούς.

Ο χρήστης στην συνέχεια καλείτε να συμπληρώσει αυτή την φόρμα με μοναδικό στοιχειό το όνομα του ταμείου και να πατήσει το πλήκτρο υποβολή.

Για τη διαδικασία που περιγράφεται παραπάνω χρησιμοποιούνται τα ακόλουθα αρχεία με τη σειρά που εκτελούνται: form_create_tameio.php και create_tameio.php.

Το αρχείο create_tameio.php δέχεται στην συνέχεια από την φόρμα το το όνομα του ταμείου και αφού κάνει τις κατάλληλες ενέργειες το εισάγει στην βάση δεδομένων όπως φαίνεται στο παρακάτω κομμάτι κώδικα:

```
$onoma=mysql_real_escape_string( $_POST['element_1'] );
```

Παρακάτω παρουσιάζεται το τμήμα του sql κώδικα που κάνει εισαγωγή του στοιχείου στη βάση δεδομένων:

```
$sql="INSERT INTO tameio (t_id, onoma)
VALUES (NULL, '$onoma')";
```

Αφού γίνει η εισαγωγή του στοιχείου στην βάση και μετά την καταχώρηση του εμφανίζεται μήνυμα το οποίο πληροφορεί τον χρηστή ότι έχει γίνει η δημιουργία επιτυχώς και ότι το ταμείο έχει καταχωρηθεί στο σύστημα.

4.3.8 Απλή αναζήτηση Ιατρού και Ασθενούς

Η απλή αναζήτηση γίνεται από το κύριο μενού πατώντας την αντίστοιχη σύνδεση. Το σύστημα ανταποκρίνεται εμφανίζοντας τρία πεδία σχετικά με την αναζήτηση τα οποία είναι η επιλογή της οντότητας, η λέξι κλειδί, και το πεδίο αναζήτησης. Ο χρήστης καλείται να διαλέξει ανάμεσα από τις οντότητες γιατρός και ασθενείς, να πληκτρολογήσει κάποια λέξη κλειδί και να επιλέξει το πεδίο στο οποίο αντιστοιχεί η λέξη κλειδί. Για παράδειγμα μπορεί να διαλέξει ιατρός με λέξη κλειδί Νταγιαμάς και πεδίο αναζήτησης Επώνυμο. Έπειτα ο χρήστης πρέπει να πατήσει το κουμπί της αναζήτησης.

Το σύστημα ανταποκρίνεται με τα αποτελέσματα της οντότητας έχοντας διάφορες επιλογές δίπλα από κάθε εγγραφή. Ο χρήστης μπορεί να κάνει κάποια ενέργεια στην εγγραφή που θέλει. Για την απλή αναζήτηση τα αρχεία που απαιτούνται είναι δύο και παρουσιάζονται με τη σειρά που εκτελούνται: apli_anazitisi.php, apli_results.php

Το αρχείο apli_anazitisi.php είναι η φόρμα που εμφανίζεται για την απλή αναζήτηση. Υπάρχουν τρία στοιχεία στην φόρμα. Το πρώτο (επιλογή οντότητας) έχει όνομα type, name="type". Το δεύτερο στοιχείο (λέξη κλειδί) έχει όνομα find, name="find" και το τρίτο στοιχείο έχει όνομα field, name="field". Τα δεδομένα που πληκτρολογεί ο χρήστης περνάνε στο σύστημα με τη βοήθεια της μεθόδου post. Αυτά θα μας βοηθήσουν στην κατανόηση του τρόπου λειτουργίας του δεύτερου αρχείου.

Το δεύτερο αρχείο apli_results.php λαμβάνει τα πεδία που αναφέρονται πιο πάνω τα περνάει από φιλτράρισμα και τα εκχωρεί σε μεταβλητές. Έπειτα εκτελεί το sql ερώτημα για την αναζήτηση. Ακολουθεί τμήμα κώδικα που δείχνει τις καταχωρήσεις, το φιλτράρισμα και ερώτημα sql.

```
$find = mysql_real_escape_string( $_POST['find'] );
$field= mysql_real_escape_string( $_POST['field'] );
$tipos= mysql_real_escape_string( $_POST['type'] );

$find = strtoupper($find);
$find = strip_tags($find);
$find = trim ($find);

$data = mysql_query("SELECT * FROM $tipos WHERE upper($field)
LIKE'$find%'");
```

Αν ο χρήστης διαλέξει να αναζητήσει γιατρό τότε θα ισχύσει η συνθήκη tipos=="doctor" και θα δημιουργήσει ένα πίνακα με όλους τους γιατρούς που ικανοποιούν το sql ερώτημα. Δίπλα στο κάθε αποτέλεσμα υπάρχουν όλες τις δυνατές ενέργειες τις οποίες μπορεί να κάνει ο χρήστης στην εγγραφή που τον ενδιαφέρει.

Σε περίπτωση που δεν βρέθηκε κάποια εγγραφή με βάση τα κριτήρια που δόθηκαν τότε το σύστημα επιστρέφει σχετικό μήνυμα. Για να γίνει αυτό χρησιμοποιείται η συνάρτηση mysql_num_rows() η οποία μετρά τις εγγραφές για το sql ερώτημα που παρουσιάστηκε παραπάνω. Το πώς γίνεται αυτό φαίνεται στον ακόλουθο κώδικα.

```
$anymatches=mysql_num_rows($data);

if ($anymatches == 0){
        echo "<br />Δεν βρέθηκε κάποιος γιατρός που να ταιριαζει με
το στοιχείο αναζήτησης<br />";
}
```

Επίσης υπάρχει κώδικας που υπενθυμίζει το στοιχείο αναζήτησης εφόσον υπάρχει. Αυτός παρουσιάζεται παρακάτω

```
if($find!=""){
        echo "<b>Στοιχείο αναζήτησης:</b> " .$find;
}
```

4.3.9 Προηγμένη αναζήτηση Ιατρού και Ασθενούς

Η προηγμένη αναζήτηση γίνεται από το κύριο μενού πατώντας την αντίστοιχη σύνδεση. Το σύστημα ανταποκρίνεται εμφανίζοντας μια φόρμα η οποία περιέχει τέσσερα πεδία σχετικά με την αναζήτηση τα οποία είναι η εισαγωγή ονόματος, επωνύμου, πατρωνύμου και επιλογή ειδικότητας.

Ο χρήστης καλείται να πληκτρολογήσει κάποια λέξη κλειδί και σε οποιοδήποτε από τα πεδία στα οποία επιθυμεί η ακόμα και σε κανένα από αυτά . Έπειτα ο χρήστης πρέπει να πατήσει το κουμπί της αναζήτησης. Το σύστημα ανταποκρίνεται με τα αποτελέσματα έχοντας διάφορες επιλογές δίπλα από κάθε εγγραφή.

Ο χρήστης μπορεί να κάνει κάποια ενέργεια στην εγγραφή που θέλει. Για την πολλαπλή αναζήτηση τα αρχεία που απαιτούνται είναι δύο και παρουσιάζονται με τη σειρά που εκτελούνται: form_MultipleSearchDoctor.php, Multiple SearchDoctor.php. Το αρχείο form_MultipleSearchDoctor.php είναι η φόρμα που εμφανίζεται για την απλή αναζήτηση. Υπάρχουν τέσσερα στοιχεία στην φόρμα. Το τρία πρώτα έχουν όνομα element_1, element_2 και element_3 για το όνομα, επώνυμο και πατρώνυμο αντίστοιχα. Το τέταρτο στοιχείο (ειδικότητα) έχει όνομα element_4 και είναι ένα drop down list το οποίο φορτώνεται δυναμικά από την βάση όπως φαίνεται στο παρακάτω κομμάτι κώδικα.

```
<?php
mysql_connect("localhost", "root", "") or die(mysql_error());
mysql_select_db("hospital") or die(mysql_error());
mysql_query("set names 'greek'");
$sql = "SELECT * FROM eidikotita";
```

Τα δεδομένα που πληκτρολογεί ο χρήστης περνάνε στο σύστημα με τη βοήθεια της μεθόδου post. Αυτά θα μας βοηθήσουν στην κατανόηση του τρόπου λειτουργίας του δεύτερου αρχείου.

```
echo '<select class="element select medium" id="element_4"
name="element_4">
<option value="" selected="selected"></option>';
$result = mysql_query($sql);
while ($row = mysql_fetch_assoc($result)) {
echo '<option value="'.$row["e_id"].'">'
.$row["kat"].'</option>';
}echo '</SELECT>';
?>
```

Το δεύτερο MultipleSearchDoctor.php λαμβάνει τα πεδία που αναφέρονται πιο πάνω τα περνάει από φιλτράρισμα και τα εκχωρεί σε μεταβλητές. Έπειτα εκτελεί το sql ερώτημα για την αναζήτηση. Ακολουθεί τμήμα κώδικα που δείχνει τις καταχωρήσεις, το φιλτράρισμα και το ερώτημα sql.

```
$element_1 = mysql_real_escape_string( $_POST['element_1'] );
$element_1 = strtoupper($element_1);
$element_1 = strip_tags($element_1);
$element_1 = trim ($element_1);
$data = mysql_query("
```

```
SELECT *
FROM doctor
WHERE onoma LIKE'$element_1%'
AND epwnimo LIKE'$element_2%'
AND patrwnimo LIKE '$element_3%'
AND eid_id LIKE '$element_4%'
");
```

Δίπλα στο κάθε αποτέλεσμα υπάρχουν όλες τις δυνατές ενέργειες τις οποίες μπορεί να κάνει ο χρήστης στην εγγραφή που τον ενδιαφέρει.

Σε περίπτωση που δεν βρέθηκε κάποια εγγραφή με βάση τα κριτήρια που δόθηκαν τότε το σύστημα επιστρέφει σχετικό μήνυμα. Για να γίνει αυτό χρησιμοποιείται η συνάρτηση mysql_num_rows() η οποία μετρά τις εγγραφές για το sql ερώτημα που παρουσιάστηκε παραπάνω. Το πώς γίνεται αυτό φαίνεται στον ακόλουθο κώδικα.

```
$anymatches=mysql_num_rows($data);
if ($anymatches == 0){
        echo "<br />Δεν βρέθηκε κάποιος γιατρός που να ταιριαζει με
το στοιχείο αναζήτησης<br />";
}
```

Επίσης υπάρχει κώδικας που υπενθυμίζει το στοιχείο αναζήτησης εφόσον υπάρχει. Αυτός παρουσιάζεται παρακάτω:

```
echo "<br /><b>Στοιχεία προς αναζήτηση:</b> <br />" ."Όνομα:
".$element_1.'<br/>'."Επώνυμο: ".$element_2.'<br/>'."Πατρώνυμο:
".$element_3.'<br/>'."Ειδικότητα: ".$a.'<br/>';
```

Με ακριβώς τον ίδιο τρόπο γίνεται και η αναζήτηση του ασθενούς, τα αρχεια τα οποια χρησιμοποιουντε είναι: form_MultipleSearchPatient.php και MultipleSearchPatient.php η μονή διάφορα είναι ότι το πεδίο element_4 είναι ένα drop down list το οποίο φορτώνει από την βάση τα ταμεία στα οποία μπορεί να ανήκει ένας ασθενής παρακάτω φαίνεται και ο κώδικας με τον οποίο φορτώνονται τα ταμεία από την βάση.

```
if ($element_4==!""){
$data2 = mysql_query("SELECT * FROM tameio where
t_id='$element_4'");
while($result = mysql_fetch_array( $data2 )){
$a=$result['onoma'];
        }
}
```

Άλλη μια διαφορά είναι το ερώτημα SELECT το οποίο κάνουμε για να βγάλουμε τα αποτελέσματα, ο κώδικας φαίνεται παρακάτω:

```
$data = mysql_query("
SELECT *
FROM patient
WHERE onoma LIKE'$element_1%'
AND epwnimo LIKE'$element_2%'
AND patrwnimo LIKE '$element_3%'
AND tam_id LIKE '$element_4%'");
```

4.3.10 Αλλαγή στοιχείων ιατρού

Υπάρχει το ενδεχόμενο κατά την υποβολή των στοιχείων ενός ιατρού να περάσουν κάποιες πληροφορίες λανθασμένα. Ο διαχειριστής με αυτή τη λειτουργία μπορεί να τροποποιήσει τα στοιχεία του ιατρού, εκτός από το username.

Για να κάνει ο διαχειριστής του συστήματος αλλαγή κάποιων στοιχείων ενός γιατρού θα πρέπει να γίνει κάποια αναζήτηση πρώτα. Αλλαγή στοιχείων μπορεί να κάνει είτε στα αποτελέσματα της απλής αναζήτησης, είτε στα αποτελέσματα της προηγμένης αναζήτησης είτε πατώντας αλλαγή στοιχείων. Στην ανάλυση που ακολουθεί θα μελετήσουμε το κουμπί αλλαγή στοιχείων γιατρού, αφού τις αναζητήσεις θα τις δούμε αναλυτικά στη συνέχεια.

Πατώντας αλλαγή στοιχείων γιατρού ο διαχειριστής πρέπει να αναζητήσει τον ιατρό που επιθυμεί. Έπειτα στα αποτελέσματα να επιλέξει αυτόν που θέλει να τροποποιήσει. Το σύστημα ανταποκρίνεται εμφανίζοντας μια φόρμα σαν αυτή της δημιουργίας ιατρού με τη μόνη διαφορά ότι όλα τα πεδία περιέχουν τις τιμές που είχαν καταχωρηθεί στη βάση δεδομένων. Ο διαχειριστής κάνει τις αλλαγές που επιθυμεί και πατά το κουμπί υποβολής. Έπειτα καλείται η php για να κάνει τις απαραίτητες αλλαγές στη βάση.

Για την διαδικασία που περιγράφεται παραπάνω χρησιμοποιούνται 4 αρχεία. Αυτά είναι με λογική σειρά κλήσης τα εξής: search_doctor_to_update.php, doctor_results_update.php, form_update_doctor.php, update_doctor_entry.php.

Τα δύο πρώτα αρχεία δεν θα μας απασχολήσουν σε αυτή την ενότητα αφού είναι παρόμοια με την απλή αναζήτηση η οποία αναλύεται σε επόμενη ενότητα. Το μόνο που χρειάζεται να γνωρίζετε για την συγκεκριμένη ενότητα είναι ότι το doctor_results_update.php στέλνει το d_id του γιατρού στην φόρμα αλλαγής στοιχείων.

Η φόρμα αλλαγής στοιχείων form_update_doctor.php, εκτελεί ένα ερώτημα sql στο πάνω μέρος της με κριτήριο το d_id. Με βάση το d_id που δέχεται καταχωρεί σε μεταβλητές όλα τα χαρακτηριστικά του γιατρού. Με τον τρόπο αυτό βάζουμε στην οδηγία value="" τη μεταβλητή που θέλουμε και εμφανίζεται η παλιά κατάχώρηση της βάσης δεδομένων στο αντίστοιχο πεδίο της φόρμας.

Μία δυσκολία υπήρξε στην ημερομηνία. Η ημερομηνία καταχωρείται από τρία πεδία σε ένα χαρακτηριστικό στη βάση δεδομένων. Έτσι το πρόβλημα που υπήρξε είναι να τοποθετηθεί η ημερομηνία από ένα αλφαριθμητικό σε τρία κομμάτια. Η μορφή της ημερομηνίας στη βάση είναι 2009-4-20. έτσι χρησιμοποιήθηκε η συνάρτηση strtok(), η οποία δέχεται μια μεταβλητή και ένα χαρακτήρα τον οποία όταν βρει θα χωρίσει το αλφαριθμητικό σε κομμάτια. Για τον επιτυχή διαχωρισμό χρησιμοποιήθηκε ένας πίνακας στον οποίο καταχωρήθηκε η μέρα, ο μήνας και το έτος. Με τον τρόπο αυτό τοποθετούνται οι τιμές του πίνακα στις αντίστοιχες θέσεις της ημερομηνίας. Ακολουθεί το κομμάτι κώδικα με την επιλογή των στοιχείων του γιατρού βάση το d_id και την καταχώρηση των τιμών στις μεταβλητές.

```
$d_id=$_POST['d_id'];

$sql = "SELECT * FROM doctor WHERE doctor.d_id=$d_id";

$result = mysql_query($sql);

while ($row = mysql_fetch_assoc($result)) {
        $username=$row["username"];
        $eid_id=$row["eid_id"];
        $onoma= $row["onoma"];
        $epwnimo=$row["epwnimo"];
        $patrwnimo=$row["patrwnimo"];
        $adt=$row["adt"];
        $filo=$row["filo"];
        $imer_gen=$row["imer_gen"];
        $dieuthinsi=$row["dieuthinsi"];
        $poli=$row["poli"];
        $xwra=$row["xwra"];
        $tk=$row["tk"];
        $til1=$row["til1"];
        $til2=$row["til2"];
        $kinito=$row["kinito"];
        $email=$row["email"];
};

$token = strtok($imer_gen, "-");
$i=0;
while ($token != false){
        $i+=1;
        $pin[$i] = "$token";
        $token = strtok("-");
}
$year=$pin[1];
$month=$pin[2];
$day=$pin[3];
```

Έχοντας όλα τα χαρακτηριστικά σε μεταβλητές μπορούμε εύκολα να βάλουμε την τιμή τους να φαίνεται σε κάποιο πεδίο κειμένου. Αυτό μπορει να επιτευχθεί βάζοντας στο value=”” του πλαισίου κειμένου την μεταβλητη. Επειδή εχουμε ένωση αλφαριθμητικών η μεταβλητή θα μπει με τη μορφή ‘.$var.’ Αντίστοιχα συμπληρώνονται και όλα τα πεδία κειμένου. Ακριβώς με τον ίδιο τρόπο γίνεται και για το textarea Ακολουθεί παράδειγμα που παρουσιάζει ακριβώς αυτή τη διαδικασία.

```
<input type="text" maxlength="255" value="'.$patrwnimo.'" />
```

Για την επιλογή της καταχωρημένης τιμής σε κάποιο drop down list θα πρέπει να πρέπει η καταχωρημενη επιλογή να πάρει την οδηγία selected=”selected”. Για να πετύχουμε αυτό θα πρέπει να κάνουμε κάποιους ελεγχους. Παρακάτω φαίνεται ο κώδικας για την ειδικότητα του γιατρού. Οι ειδικότητες φορτώνονται δυναμικά από την βάση δεδομένων και τον πίνακα ειδικότητα.

```
$sql = "SELECT * FROM eidikotita";

echo '
<select class="element select medium" id="element_14"
name="element_14">

<option value="" ></option>';

$result = mysql_query($sql);

while ($row = mysql_fetch_assoc($result)) {
        if($eid_id==$row["e_id"])
                echo '<option value="'.$row["e_id"].'" selected="selected"
                >'.$row["kat"].'</option>';
        else
                echo '<option
                value="'.$row["e_id"].'">'.$row["kat"].'</option>';
}

echo '</SELECT>';
```

Με αυτόν τον τρόπο που φαίνεται παραπάνω για τον συγκεκριμένο γιατρό φορτώνονται και προστίθενται όλες οι ιατρικές ειδικότητες που υπάρχουν στη βάση δεδομένων. Από αυτές τις ειδικότητες γίνεται προεπιλογή η ειδικότητα στην οποία ανήκει ο ιατρός δίνοντας την οδηγία selected="selected".

Τελευταία περίπτωση είναι να έχουμε κάποιο radio button. Αυτό γίνεται στην επιλογή του φύλου. Για να επιλεχτεί αυτόματα το radio button στο οποίο

ανήκει ο ιατρός πρέπει να δώσουμε την οδηγία checked. Ακολουθεί το τμήμα κώδικα που είναι υπεύθυνο για την επιλογή του καταχωρημένου φύλου.

```
if($filo=="Άντρας"){
echo '
<input id="element_13_1" name="element_13" class="element radio"
type="radio" value="Άντρας" checked/>
<label class="choice" for="element_13_1">Άντρας</label>
<input id="element_13_2" name="element_13" class="element radio"
type="radio" value="Γυναίκα" />
<label class="choice" for="element_13_2">Γυναίκα</label>';
}
elseif($filo=="Γυναίκα"){
echo '
<input id="element_13_1" name="element_13" class="element radio"
type="radio" value="Άντρας" />
<label class="choice" for="element_13_1">Άντρας</label>
<input id="element_13_2" name="element_13" class="element radio"
type="radio" value="Γυναίκα" checked />
<label class="choice" for="element_13_2">Γυναίκα</label>';
}
else{
echo '
<input id="element_13_1" name="element_13" class="element radio"
type="radio" value="Άντρας" />
<label class="choice" for="element_13_1">Άντρας</label>
<input id="element_13_2" name="element_13" class="element radio"
type="radio" value="Γυναίκα" />
<label class="choice" for="element_13_2">Γυναίκα</label>';
}
```

Το παράπανω τμήμα κώδικα επιλέγει τον άντρα αν το καταχωρημένο φύλο είναι άντρας, γυναίκα αν το καταχωρημένο φύλο είναι γυναίκα και δεν επιλέγει τίποτα αν δεν υπάρχει καταχωρημένο στοιχείο για το φύλο.

Όταν ο χρήστης πατήσει το κουμπί της υποβολής καλείται το update_doctor_entry.php για να κάνει υποβολή και ανανέωση των στοιχείων της εγγραφής. Κατά την υποβολή καλούνται όλες οι συναρτήσεις όπως και στην δημιουργία γιατρού. Στο update_doctor_entry.php στέλνεται και το d_id του γιατρού για να γίνει η αναβάθμιση.

Τα δεδομένα τα οποία μπαίνουν στη φόρμα φιλτράρονται μέσω php injection και γίνεται και validation των δεδομένων για λόγους ασφάλειας. Αφού καταχωρηθεί κάθε πεδίο σε μεταβλητή εκτελείται το ερώτημα που παρουσιάζεται παρακάτω.

```
$sql="UPDATE doctor
SET d_id='$d_id', eid_id='$eid_id', onoma='$onoma',
epwnimo='$epwnimo',
patrwnimo='$patrwnimo', adt='$adt', filo='$filo', imer_gen='$imer_gen',
dieuthinsi='$dieuthinsi', poli='$poli', xwra='$xwra',        tk='$tk', til1='$til1',
til2='$til2', kinito='$kinito', email='$email'
WHERE d_id='$d_id'";

if (!mysql_query($sql,$con)){
        die('Error: ' . mysql_error());
}else{
        echo "<h3>Ο ιατρός <b> ".$epwnimo." ".$onoma." </b>
τροποποιήθηκε επιτυχώς</h3> <br />";
}
```

Σημαντική παρατήρηση κατά την αναβάθμιση είναι ότι το d_id παραμένει το ίδιο. Σε περίπτωση που δεν εκτλεστεί σωστά το ερώτημα εμφανίζεται η λέξη σφάλμα με το αντίστοιχο σφάλμα.

4.3.11 Αλλαγή στοιχείων ασθενούς

Η αλλαγή των στοιχείων ενός ασθενή λειτουργεί ακριβώς κατά τον ίδιο τρόπο με την αλλαγή στοιχείων ενός ιατρού. Δεν έχει κάποια ιδιαιτερότητα σε σχέση με την προηγούμενη περιγραφή.

Μπορούμε να κάνουμε αλλαγή των στοιχείων κάποιου ασθενή έπειτα από μια απλή ή προηγμένη αναζήτηση πατώντας το μολυβάκι στα αποτελεσματα είτε από το κουμπί αλλαγή στοιχείων ασθενούς στο κύριο μενού.

Σε περίπτωση που πατήσουμε από το κύριο μενού τα αρχεία που χρειαζόμαστε είναι τα: search_patient_to_update.php, patient_results_update.php,

form_update_patient.php, update_patient.php. Παρακάτω παρουσιάζεται το ερώτημα για την αναβάθμιση της εγγραφής του πίνακα patient για το συγκεκριμένο p_id. Ο τρόπος είναι ίδιος με αυτόν του ιατρού έτσι παρουσιάζεται μόνο το ερώτημα.

```
$sql="UPDATE patient
        SET p_id='$p_id', onoma='$onoma', epwnimo='$epwnimo',
        patrwnimo='$patrwnimo', adt='$adt', filo='$filo',
        imer_gen='$imer_gen', dieuthinsi='$dieuthinsi', poli='$poli',
        xwra='$xwra', tk='$tk', tam_id='$tam_id', imer_egg='$imer_egg',
        til1='$til1', til2='$til2', kinito='$kinito', email='$email'
        WHERE p_id='$p_id'";
```

Σημαντική παρατήρηση κατά την αναβάθμιση είναι ότι το p_id παραμένει το ίδιο.

4.3.12 Αλλαγή ονόματος ταμείου ασφάλισης

Η αλλαγή του ονόματος του ταμείου ασφάλισης μπορεί να γίνει επιλέγοντας αλλαγή ταμείου ασφάλισης στο κύριο μενού της εφαρμογής. Τα αρχεία που απαιτούνται είναι τέσσερα και εκτελούνται με τη σειρά που φαίνονται: search_tameio_to_update.php, tameio_results_update.php, form_update tameio.php, update_tameio.php. Η διαδικασία για την αλλαγή είναι παρόμοια με αυτή της αλλαγής του γιατρού που περιγράφεται παραπάνω. Στο τελευταίο αρχείο στέλνεται το t_id και το νέο όνομα. Είναι σημαντικό να τονιστεί ότι το t_id παραμένει το ίδιο. Το ερώτημα για το update φαίνεται παρακάτω.

```
$sql="UPDATE tameio SET t_id='$t_id', onoma='$onoma' WHERE
t_id='$t_id'";
```

4.3.13 Αλλαγή ονόματος ιατρικής ειδικότητας

Η αλλαγή του ονόματος ειδικότητας μπορεί να γίνει επιλέγοντας αλλαγή ονόματος ειδικότητας στο κύριο μενού της εφαρμογής. Τα αρχεία που απαιτούνται είναι τέσσερα και εκτελούνται με τη σειρά που φαίνονται: search_eidikotita _to_update.php, eidikotita_results_update.php, form_update_eidikotita.php, update_eidikotita.php. Η διαδικασία για την αλλαγή είναι παρόμοια με αυτή του γιατρού που περιγράφεται παραπάνω. Στο τελευταίο αρχείο στέλνεται το e_id και το νέο όνομα. Είναι σημαντικό να τονιστεί ότι το e_id παραμένει το ίδιο. Το ερώτημα για το update φαίνεται παρακάτω.

```
$sql="UPDATE eidikotita SET e_id='$e_id', kat='$kat' WHERE e_id='$e_id'";
```

4.3.14 Προβολή στοιχείων ιατρού

Η προβολή στοιχείων ιατρού μπορεί να γίνει στα αποτελέσματα της απλής ή προηγμένης αναζήτησης ιατρού ή πατώντας προβολή στοιχείων ιατρού από το κύριο μενού.

Σε περίπτωση που ο χρήστης πατήσει από το μενού τα αρχεία που εκτελούνται είναι τα: search_doctor_to_view.php, doctor_results_view.php, form_view_doctor.php.

Με όποιο τρόπο και να διαλέξει ο χρήστης να κάνει την προβολή των στοιχείων το αποτέλεσμα είναι να ανοίξει το form_view_doctor.php το οποίο δέχεται ένα d_id και βρίσκει όλα τα στοιχεία σχετικά με τον ιατρό. Η διαδικασία που ακολουθείται είναι ακριβώς η ίδια με την αλλαγή.

Στην προβολή στοιχείων όμως η φόρμα δεν έχει μέθοδο αποστολής των στοιχείων ούτε κάποιο action δηλωμένο. Επίσης όλα τα πεδία υποβολής στοιχείων έχουν για value τις αντίστοιχες τιμές που έχουν καταχωρηθεί στη βάση δεδομένων αλλά έχουν την οδηγία disabled. Με την οδηγία αυτή ο χρήστης δεν μπορεί να

επηρεάσει κάποια από τις πληροφορίες που βλέπει για τον ιατρό. Τέλος η φόρμα δεν έχει κουμπιά υποβολής αλλά το μόνο που μπορεί να κάνει είναι να επιστρέψει στην αρχική σελίδα.

4.3.15 Προβολή στοιχείων ασθενούς

Ο τρόπος της προβολής των στοιχείων του ασθενούς είναι ίδιος με αυτή του ιατρού. Ο διαχειριστής και ο γιατρός μπορεί να κάνει προβολή των στοιχείων μέσω της απλής αναζήτησης, της προηγμένης αναζήτησης ή από την προβολή των στοιχείων του ασθενούς στο κύριο μενού.

Σε περίπτωση που γίνει επιλογή από το μενού έχουμε χρήση των αρχείων search_patient_to_view.php, patient_results_view.php, form_view_patient.php. Όποιο τρόπο και αν ακολουθήσει ο χρήστης για την διαδικασία στέλνεται το p_id του ασθενούς στο form_view_patient.php.

4.3.16 Διαγραφή γιατρού

Η διαγραφή ενός γιατρού μπορεί να γίνει μόνο από τον διαχειριστή του συστήματος κάνοντας κλικ στο αντίστοιχο link του μενού της εφαρμογής. Για λόγους ασφαλείας και αξιοπιστίας του ιστορικού τέτοια διαγραφή δεν μπορεί να κάνει ο γιατρός.

Επιλέγοντας αυτή τη λειτουργία ο διαχειριστής καλείται να επιλέξει τον γιατρό τον οποίο θέλει να διαγράψει. Η μέθοδος έχει αναλυθεί ήδη και είναι αυτή της απλής αναζήτησης. Το σύστημα ανταποκρίνεται με μια λίστα από αποτελέσματα γιατρών με βάση το κριτήριο αναζήτησης. Έπειτα ο χρήστης πατάει το κουμπί της διαγραφής δίπλα από τον γιατρό που θέλει να σβήσει. Το σύστημα ανταποκρίνεται με μήνυμα επιβεβαίωσης της ενέργειας και μετά από θετική απάντηση του χρήστη σβήνεται ο αντίστοιχος γιατρός.

Για την επίτευξη της παραπάνω ενέργειας χρησιμοποιούνται τρία αρχεία. search_doctor_to_delete.php, doctor_results_delete.php, delete_doctor.php. παρακάτω παρουσιάζεται το ερώτημα της διαγραφής.

```
$result = mysql_query("SELECT * FROM doctor WHERE d_id ='$_POST[d_id]'");
while($row = mysql_fetch_array($result)){
        $user = $row['username'];
}
$d_id=$_POST['d_id'];
$sql1="DELETE FROM doctor WHERE d_id =$d_id";
```

Στο αρχείο αυτό στέλνονται όλα τα απαραίτητα δεδομένα για την διαγραφή συγκεκριμένου γιατρού από τον πίνακα doctors.Το δεδομένο το οποίο χαρακτηρίζει μοναδικά την κάθε εγγραφή του πίνακα doctor είναι το d_id. Με την επιτυχή διαγραφή του γιατρού εμφανίζεται αντίστοιχο μήνυμα.

Σε αυτό το σημείο θα πρέπει να αναφέρουμε ότι πριν την διαγραφή γίνεται ένας έλεγχος με ένα ερώτημα στην βάση το οποίο «κοιτάζει» αν ο συγκεκριμένος γιατρός έχει καταχωρήσει στο σύστημα κάποια διάγνωση για κάποιον ασθενή, στην περίπτωση που βρεθούν στοιχειά τα οποία συσχετίζονται με αυτόν τον γιατρό, τότε δεν είναι δυνατή η διαγραφή του γιατρού για λογούς διατήρησης της ακεραιότητας του ιστορικού των ασθενών τα οποία είναι ευαίσθητα προσωπικά δεδομένα. Παρακάτω φαίνεται το κομμάτι κώδικα με το οποίο γίνεται αυτός ο έλεγχος.

```
$data = mysql_query("SELECT * FROM examines WHERE
d_id=$d_id");
$anymatches=mysql_num_rows($data);
if($anymatches == 0){
}
else{
}
```

Στο παραπάνω κομμάτι ελέγχουμε στην ουσία αν υπάρχει κάποια εγγραφή στον πίνακα examines που να αφορά τον συγκεκριμένο γιατρό και αναλόγως το αποτέλεσμα (έλεγχος if) κάνουμε τις κατάλληλες ενέργειες (εμφάνιση μηνύματος ότι ο γιατρός δεν μπορεί να διαγράφει ή διαγραφή του γιατρού).

4.3.17 Διαγραφή ασθενούς

Ο τρόπος με τον οποίο γίνεται ι η διαγραφή ενός ασθενή είναι ίδιος με αυτόν της διαγραφής ενός γιατρού ο οποίος αναφέρθηκε παραπάνω.

Η διαγραφή ενός ασθενή μπορεί να γίνει από τον διαχειριστή και από έναν γιατρό του συστήματος κάνοντας κλικ στο αντίστοιχο link του μενού της εφαρμογής.

Επιλέγοντας αυτή τη λειτουργία ο χρήστης καλείται να επιλέξει τον ασθενή τον οποίο θέλει να διαγράψει. Η μέθοδος έχει αναλυθεί ήδη και είναι αυτή της απλής αναζήτησης. Το σύστημα ανταποκρίνεται με μια λίστα από αποτελέσματα ασθενών με βάση το κριτήριο αναζήτησης. Έπειτα ο χρήστης πατάει το κουμπί της διαγραφής δίπλα από τον ασθενή που θέλει να σβήσει. Το σύστημα ανταποκρίνεται με μήνυμα επιβεβαίωσης της ενέργειας και μετά από θετική απάντηση του χρήστη σβήνεται ο αντίστοιχος ασθενής.

```
$p_id=$_POST['p_id'];
$sql="DELETE FROM patient WHERE p_id =$p_id";
```

Για την επίτευξη της παραπάνω ενέργειας χρησιμοποιούνται τρία αρχεία. search_patient_to_delete.php, patient_results_delete.php, delete_patient.php. παρακάτω παρουσιάζεται το SQL ερώτημα της διαγραφής.

Στο αρχείο αυτό στέλνονται όλα τα απαραίτητα δεδομένα για την διαγραφή συγκεκριμένου ασθενή από τον πίνακα patient.Το δεδομένο το οποίο χαρακτηρίζει

μοναδικά την κάθε εγγραφή του πίνακα patient είναι το p_id.Με την επιτυχή διαγραφή του ασθενή εμφανίζεται αντίστοιχο μήνυμα.

Σε αυτό το σημείο θα πρέπει να αναφέρουμε ότι πριν την διαγραφή γίνεται ένας έλεγχος με ένα ερώτημα στην βάση το οποίο «κοιτάζει» αν για τον συγκεκριμένο ασθενή έχει καταχωρήθει στο σύστημα κάποια διάγνωση η εξέταση, στην περίπτωση που βρεθούν στοιχειά τα οποία συσχετίζονται με αυτόν τον ασθενή, τότε δεν είναι δυνατή η διαγραφή του ασθενή για λογούς διατήρησης της ακεραιότητας του ιστορικού των ασθενών τα οποία είναι ευαίσθητα προσωπικά δεδομένα. Παρακάτω φαίνεται το κομμάτι κώδικα με το οποίο γίνεται αυτός ο έλεγχος.

```
$data = mysql_query("SELECT * FROM implements WHERE
p_id=$p_id");
$anymatches=mysql_num_rows($data);
$data2 = mysql_query("SELECT * FROM examines WHERE
p_id=$p_id");
$anymatches2=mysql_num_rows($data2);
if($anymatches == 0 && $anymatches2 == 0){}
elseif {}
 ……
```

Στο παραπάνω κομμάτι ελέγχουμε στην ουσία αν υπάρχει κάποια εγγραφή στον πίνακα examines και στον πίνακα implements που να αφορά τον συγκεκριμένο ασθενή και αναλόγως το αποτέλεσμα (έλεγχος if) κάνουμε τις κατάλληλες ενέργειες (εμφάνιση μηνύματος ότι ο ασθενής δεν μπορεί να διαγράφει ή διαγραφή του ασθενή).

4.3.18 Διαγραφή ειδικότητας

Η διαγραφή μιας ειδικότητας μπορεί να γίνει από τον διαχειριστή και του συστήματος κάνοντας κλικ στο αντίστοιχο link του μενού της εφαρμογής.

Επιλέγοντας αυτή τη λειτουργία ο χρήστης καλείται να επιλέξει ειδικότητα την οποία θέλει να διαγράψει. Η μέθοδος έχει αναλυθεί ήδη και είναι αυτή της απλής αναζήτησης. Το σύστημα ανταποκρίνεται με μια λίστα από αποτελέσματα με όλες τις ειδικότητες. Έπειτα ο χρήστης πατάει το κουμπί της διαγραφής δίπλα από μια ειδικότητα που θέλει να σβήσει. Το σύστημα ανταποκρίνεται με μήνυμα επιβεβαίωσης της ενέργειας και μετά από θετική απάντηση του χρήστη σβήνεται η αντίστοιχη ειδικότητα.

```
$_POST['e_id'];
DELETE FROM eidikotita WHERE e_id=$e_id";
```

Για την επίτευξη της παραπάνω ενέργειας χρησιμοποιούνται τρία αρχεία. search_eidikotita_to_delete.php, eidikotita_results_delete.php, delete_eidiko tita.php. παρακάτω παρουσιάζεται το SQL ερώτημα της διαγραφής.

Στο αρχείο αυτό στέλνονται όλα τα απαραίτητα δεδομένα για την διαγραφή συγκεκριμένης ειδικότητας από τον πίνακα eidikotita.Το δεδομένο το οποίο χαρακτηρίζει μοναδικά την κάθε εγγραφή του πίνακα eidikotita είναι το e_id.Με την επιτυχή διαγραφή της ειδικότητας εμφανίζεται αντίστοιχο μήνυμα.

4.3.19 Διαγραφή Ταμείου

Η διαγραφή ενός ταμείου μπορεί να γίνει από τον διαχειριστή και του συστήματος κάνοντας κλικ στο αντίστοιχο link του μενού της εφαρμογής.

Επιλέγοντας αυτή τη λειτουργία ο χρήστης καλείται να επιλέξει το ταμείο το οποίο θέλει να διαγράψει. Η μέθοδος έχει αναλυθεί ήδη και είναι αυτή της απλής αναζήτησης. Το σύστημα ανταποκρίνεται με μια λίστα από αποτελέσματα με όλα τα ταμεία. Έπειτα ο χρήστης πατάει το κουμπί της διαγραφής δίπλα από το ταμείο που θέλει να σβήσει. Το σύστημα ανταποκρίνεται με μήνυμα επιβεβαίωσης της ενέργειας και μετά από θετική απάντηση του χρήστη σβήνεται το αντίστοιχο ταμείο.

```
$t_id=$_POST['t_id'];
        $sql="DELETE FROM tameio WHERE t_id ='$t_id'";
```

Για την επίτευξη της παραπάνω ενέργειας χρησιμοποιούνται τρία αρχεία. search_tameio_to_delete.php, tameio_results_delete.php, delete_tameio.php. παρακάτω παρουσιάζεται το SQL ερώτημα της διαγραφής.

Στο αρχείο αυτό στέλνονται όλα τα απαραίτητα δεδομένα για την διαγραφή του συγκεκριμένου ταμείου από τον πίνακα tameio.Το δεδομένο το οποίο χαρακτηρίζει μοναδικά την κάθε εγγραφή του πίνακα tameio είναι το t_id.Με την επιτυχή διαγραφή του ταμείου εμφανίζεται αντίστοιχο μήνυμα.

4.3.20 Καταχώρηση Διάγνωσης

Για να γίνει κάποια καταχώρηση διάγνωσης ο χρήστης θα πρέπει να πατήσει το σχετικό link από το κύριο μενού. Επιλέγοντας καταχώρηση διάγνωσης το σύστημα ανταποκρίνεται παραπέμποντας τον χρήστη να αναζητήσει τον ασθενή στον οποίο θέλει να κάνει διάγνωση. Η αναζήτηση αυτή είναι μια παρόμοια με την απλή αναζήτηση που περιγράφεται παραπάνω. Από την αναζήτηση προκύπτουν τα αποτελέσματα από τα οποία πρέπει ο χρήστης να επιλέξει τον ασθενή που επιθυμεί. Το σύστημα εμφανίζει τη φόρμα εισαγωγής διάγνωσης με τα βασικά στοιχεία του ασθενούς και δύο μεγάλα πεδία συμπλήρωσης κειμένου σχετικά με την διάγνωση και θεραπεία. Ο χρήστης καλείται να συμπληρώσει αυτά τα πεδία και να πατήσει υποβολή, αλλά δεν μπορεί να αφήσει την διάγνωση κενή. Πατώντας υποβολή καταχωρούνται τα στοιχεία της διάγνωσης στη βάση δεδομένων.

Για τη διαδικασία που περιγράφεται παραπάνω χρησιμοποιούνται τα ακόλουθα αρχεία με τη σειρά που εκτελούνται: search_patient_to_diagnose.php, patient_results_diagnose.php, form_diagnose_patient.php, diagnose_patient.php. Το βασικό αρχείο το οποίο θα εξετάσουμε είναι το diagnose_patient.php.

Το αρχείο αυτό δέχεται τέσσερα στοιχεία για τη λειτουργία του. Αυτά είναι το d_id, το p_id, η diagnwsi και η therapeia. Τα στοιχεία αυτά καταχωρούνται σε μεταβλητές όπως φαίνεται παρακάτω.

```
$d_id=$_GET['d_id'];
$p_id=$_POST['p_id'];
$diagnwsi=mysql_real_escape_string($_POST['element_2'
]);
```

Οι μεταβλητές αυτές είναι απαραίτητες για την συμπλήρωση του πίνακα με τις διαγνώσεις. Παρακάτω παρουσιάζεται το τμήμα του sql κώδικα που κάνει εισαγωγή των στοιχείων στη βάση δεδομένων.

```
$sql = "INSERT INTO examines (d_id, p_id, diagnwsi, therapeia, date_time)
        VALUES ( '$d_id', '$p_id', '$diagnwsi', '$therapeia',
        CURRENT_TIMESTAMP);";
```

Όπως φαίνεται κατά την εισαγωγή της διάγνωσης εισάγεται και ένα timestamp από την ώρα του συστήματος που τρέχει στον wamp server. Με αυτό τον τρόπο προσδιορίζεται μοναδικά και η διάγνωση συγκεκριμένου γιατρού προς συγκεκριμένο ασθενή. Τέλος το σύστημα αναφέρει σχετικό μήνυμα για την επιτυχή εισαγωγή της διάγνωσης στο σύστημα.

4.3.21 Προβολή διαγνώσεων ασθενούς

Πατώντας το σχετικό link στο κύριο μενού της εφαρμογής ο χρήστης καλείται να αναζητήσει τον ασθενή του οποίου θέλει να δει τις διαγνώσεις. Το σύστημα ανταποκρίνεται εμφανίζοντας όλους τους ασθενείς για αποτελέσματα σύμφωνα με τα κριτήρια που δόθηκαν. Επιλέγοντας τον ασθενή εμφανίζεται μια λίστα με όλες τις διαγνώσεις που είναι καταχωρημένες στη βάση δεδομένων για τον συγκεκριμένο ασθενή. Ο χρήστης καλείται τώρα να επιλέξει τη διάγνωση που επιθυμεί για προβολή. Το σύστημα ανταποκρίνεται και πάλι εμφανίζοντας μια

φόρμα ίδια με τη φόρμα της εισαγωγής αλλά με τα πεδία κλειδωμένα και χωρίς κουμπιά υποβολής.

Η προβολή της διάγνωσης είναι παρόμοια με την προβολή του γιατρού και υπάρχει λόγος να αναλυθεί και πάλι. Για την λειτουργία της προβολής διάγνωσης ασθενούς απαραίτητα αρχεία είναι τέσσερα και εμφανίζονται με τη σειρά εκτέλεσης τους: search_patient_to_view_diagnosis.php, view_patient_diagnosis.php, patient_diagnosis_results_view.php, form_view_ patient_diagnosis.php.

Για την προβολή των διαγνώσεων του ασθενούς δηλαδή το αρχείο view_patient_diagnosis.php το μόνο που θα πρέπει να δείξουμε είναι την επιλογή όλων ων διαγνώσεων του ασθενούς που γίνεται με το ερώτημα που ακολουθεί.

```
$data = mysql_query("SELECT * FROM examines WHERE
p_id='$_POST[p_id]'");
```

4.3.22 Διαγραφή διάγνωσης ασθενούς

Η διαγραφή μιας διάγνωσης ενός ασθενούς μπορεί να γίνει μόνο από τον διαχειριστή του συστήματος κάνοντας κλικ στο αντίστοιχο link του μενού της εφαρμογής. Για λόγους ασφαλείας και αξιοπιστίας των διαγνώσεων τέτοια διαγραφή δεν μπορεί να κάνει ο γιατρός.

Επιλέγοντας αυτή τη λειτουργία ο διαχειριστής καλείται να επιλέξει τον ασθενή για τον οποίο θέλει να διαγράψει την διάγνωση. Η μέθοδος έχει αναλυθεί ήδη και είναι αυτή της απλής αναζήτησης. Το σύστημα ανταποκρίνεται με μια λίστα από αποτελέσματα ασθενών με βάση το κριτήριο αναζήτησης. Ο χρήστης επιλέγει τον ασθενή που θέλει και η εφαρμογή εμφανίζει όλες τις διαγνώσεις του συγκεκριμένου ασθενούς. Έπειτα ο χρήστης πατάει το κουμπί της διαγραφής δίπλα από τη διάγνωση που θέλει να σβήσει. Το σύστημα ανταποκρίνεται με μήνυμα επιβεβαίωσης της ενέργειας και μετά από θετική απάντηση του χρήστη σβήνεται η διάγνωση.

Για την επιτευξη της παραπάνω ενέργειας χρησιμοποιούνται τέσσερα αρχεία. search_patient_to_delete_diagnosis.php, patient_results_view.php, patient_diagnosis_to_delete.php, delete_patient_diagnosis.php. παρακάτω παρουσιάζεται το ερώτημα της διαγραφής.

```
$sql = "DELETE FROM examines WHERE d_id = '$_POST[d_id]' and p_id =
'$_POST[p_id]' and date_time = '$_POST[date_time]'";
```

Στο αρχείο αυτό στέλνονται όλα τα απαραίτητα δεδομένα για την διαγραφή συγκεκριμένης διάγνωσης από τον πίνακα examines.Τα δεδομένα αυτά τα οποία χαρακτηρίζουν μοναδικά την κάθε εγγραφή του πίνακα examines είναι τα d_id, p_id, date_time.Με την επιτυχή διαγραφή της διάγνωσης εμφανίζεται αντίστοιχο μήνυμα.

4.3.23 Καταχώρηση εξετάσεων

Για να γίνει κάποια καταχώρηση εξέτασης ο χρήστης θα πρέπει να πατήσει το σχετικό link από το κύριο μενού. Επιλέγοντας καταχώρηση εξέτασης το σύστημα ανταποκρίνεται παραπέμποντας τον χρήστη να αναζητήσει τον ασθενή στον οποίο θέλει να κάνει διάγνωση καθώς και να επιλέξει τι είδους εξέταση θα κάνει καταχώρηση. Η αναζήτηση αυτή είναι μια παρόμοια με την απλή αναζήτηση που περιγράφεται παραπάνω. Από την αναζήτηση προκύπτουν τα αποτελέσματα από τα οποία πρέπει ο χρήστης να επιλέξει τον ασθενή που επιθυμεί. Το σύστημα εμφανίζει τη φόρμα εισαγωγής εξέτασης (ανάλογα με το τα τι έχει επιλέξει ο χρήστης) με τα στοιχεία της εξέτασης. Ο χρήστης καλείται να συμπληρώσει αυτά τα πεδία και να πατήσει υποβολή. Πατώντας υποβολή καταχωρούνται τα στοιχεία της διάγνωσης στη βάση δεδομένων.

Για τη διαδικασία που περιγράφεται παραπάνω χρησιμοποιούνται τα ακόλουθα αρχεία με τη σειρά που εκτελούνται: search_patient_to_ assign_exam.php, patient_results_assign_exam.php.Στην συνεχεία και αφού ο χρήστης επιλέξει τον ασθενή στον οποίο θέλει να εισάγει την εξέταση εμφανίζεται

η φόρμα της εξέτασης που θέλει να καταχωρήσει ανάλογα με το τι έχει επιλέξει στο αρχείο search_patient_to_assign_exam.php.

Παρακάτω αναφέρονται όλοι οι τύποι των εξετάσεων καθώς και τα SQL Query που χρησιμοποιούμε για να εισάγουμε τα στοιχεία στην βάση δεδομένων.

4.3.23.1 Αιματολογική εξέταση

Τα αρχεία που χρησιμοποιούνται για την εισαγωγή αυτής της εξέτασης είναι τα έξης: form_aimatologiki.php, create_aimatologiki.php.

Παρακάτω φαίνεται ο κώδικας του αρχείου create_aimatologiki.php ο όποιος δέχεται από την φόρμα form_aimatologiki.php τα δεδομένα, κάνει τις κατάλληλες εργασίες και τα καταχωρεί στην βάση.

```
$leuka_aim= mysql_real_escape_string( $_POST['element_1'] );
$lemfokyttara= mysql_real_escape_string( $_POST['element_2'] );
$erythra_aim= mysql_real_escape_string( $_POST['element_3'] );
$aimatokritis= mysql_real_escape_string( $_POST['element_4'] );
$aimosferini= mysql_real_escape_string( $_POST['element_5'] );
$aimopetalia= mysql_real_escape_string( $_POST['element_6'] );
```

Ο παρακάτω κώδικας δείχνει τα απαραίτητα SQL Insert για τον πίνακα aimatologikh, exam και implements.

```
$sql="INSERT INTO exam
(ex_id, tipos) VALUES
(NULL, 'Αιματολογική')";
$sql2="INSERT INTO aimatologiki
(ex_id, leuka_aim, lemfokyttara, erythra_aim, aimatokritis, aimosferini,
aimopetalia)
VALUES (
LAST_INSERT_ID(),
'$leuka_aim',
'$lemfokyttara',
'$erythra_aim',
'$aimatokritis',
'$aimosferini',
'$aimopetalia')";
$sql3="INSERT INTO implements
(p_id, ex_id, date_time)
VALUES ('$p_id', LAST_INSERT_ID(), CURRENT_TIMESTAMP) ";
```

4.3.23.2 Βιοχημικός έλεγχος

Τα αρχεία που χρησιμοποιούνται για την εισαγωγή αυτής της εξέτασης είναι τα έξης: form_ bioximikos.php, create_ bioximikos.php.

Παρακάτω φαίνεται ο κώδικας του αρχείου create_ bioximikos.php ο όποιος δέχεται από την φόρμα form_ bioximikos.php τα δεδομένα, κάνει τις κατάλληλες εργασίες και τα καταχωρεί στην βάση.

```
mysql_real_escape_string( $_POST['element_1'] );
$sgpt= mysql_real_escape_string( $_POST['element_2'] );
$sakxaro= mysql_real_escape_string( $_POST['element_3'] );
$ouria= mysql_real_escape_string( $_POST['element_4'] );
$kreatinini= mysql_real_escape_string( $_POST['element_5'] );
$ouriko_oksi= mysql_real_escape_string( $_POST['element_6'] );
$ca= mysql_real_escape_string( $_POST['element_7'] );
$p= mysql_real_escape_string( $_POST['element_8'] );
$k= mysql_real_escape_string( $_POST['element_9'] );
$na= mysql_real_escape_string( $_POST['element_10'] );
$leukwmata_olika= mysql_real_escape_string( $_POST['element_11'] );
$leukwmata_leukwmatines= mysql_real_escape_string( $_POST['element_12'] );
$hdl= mysql_real_escape_string( $_POST['element_13'] );
$ldl= mysql_real_escape_string( $_POST['element_14'] );
$triglikeridia= mysql_real_escape_string( $_POST['element_15'] );
$fe= mysql_real_escape_string( $_POST['element_16'] );
$feritinh= mysql_real_escape_string( $_POST['element_17'] );
```

Ο παρακάτω κώδικας δείχνει τα απαραίτητα SQL Insert για τον πίνακα bioximikos, exam και implements.

```
$sql="INSERT INTO exam
(ex_id, tipos)
VALUES
(NULL, 'Βιοχημικός')";

$sql2="INSERT INTO bioximikos (ex_id, sgot, sgpt, sakxaro, ouria, kreatinini,
ouriko_oksi, ca, p, k, na, leukwmata_olika, leukwmata_leukwmatines, hdl, ldl,
triglikeridia, fe, feritinh)
VALUES (
LAST_INSERT_ID(),'$sgot','$sgpt','$sakxaro','$ouria','$kreatinini','$ouriko_oksi'
,'$ca','$p','$k','$na','$leukwmata_olika','$leukwmata_leukwmatines','$hdl',
'$ldl','$triglikeridia','$fe','$feritinh')";

$sql3="INSERT INTO implements
(p_id, ex_id, date_time)
VALUES
('$p_id', LAST_INSERT_ID(), CURRENT_TIMESTAMP) ";
```

4.3.23.3 Ορμονικός έλεγχος

Τα αρχεία που χρησιμοποιούνται για την εισαγωγή αυτής της εξέτασης είναι τα έξης: form_ ormonikos.php, create_ ormonikos.php.

Παρακάτω φαίνεται ο κώδικας του αρχείου create_ ormonikos.php ο όποιος δέχεται από την φόρμα form_ ormonikos.php τα δεδομένα, κάνει τις κατάλληλες εργασίες και τα καταχωρεί στην βάση.

```
$p_id=$_POST['p_id'];
$TSH= mysql_real_escape_string( $_POST['element_1'] );
$T4= mysql_real_escape_string( $_POST['element_2'] );
$T3= mysql_real_escape_string( $_POST['element_3'] );
```

Ο παρακάτω κώδικας δείχνει τα απαραίτητα SQL Insert για τον πίνακα ormonikos, exam και implements.

```
$sql="INSERT INTO exam (ex_id, tipos) VALUES (NULL, 'Ορμονικός')";
$sql2="INSERT INTO ormonikos (ex_id, TSH, T4, T3)
VALUES (
LAST_INSERT_ID(),
'$TSH',
'$T4',
'$T3')";
$sql3="INSERT INTO implements
(p_id, ex_id, date_time)
VALUES
('$p_id', LAST_INSERT_ID(), CURRENT_TIMESTAMP) ";
```

4.3.23.4 Ορολογικός έλεγχος

Τα αρχεία που χρησιμοποιούνται για την εισαγωγή αυτής της εξέτασης είναι τα έξης: form_ orologikos.php, create_ orologikos.php.

Παρακάτω φαίνεται ο κώδικας του αρχείου create_ orologikos.php ο όποιος δέχεται από την φόρμα form_ orologikos.php τα δεδομένα, κάνει τις κατάλληλες εργασίες και τα καταχωρεί στην βάση.

```
$p_id=$_POST['p_id'];
$widal= mysql_real_escape_string( $_POST['element_1'] );
$wright= mysql_real_escape_string( $_POST['element_2'] );
$MonoSpot= mysql_real_escape_string( $_POST['element_3'] );
$RPR= mysql_real_escape_string( $_POST['element_4'] );
$CMV= mysql_real_escape_string( $_POST['element_5'] );
$toksoplasma= mysql_real_escape_string( $_POST['element_6'] );
$erythra= mysql_real_escape_string( $_POST['element_7'] );
$HSV= mysql_real_escape_string( $_POST['element_8'] );
$EBV= mysql_real_escape_string( $_POST['element_9'] );
```

Ο παρακάτω κώδικας δείχνει τα απαραίτητα SQL Insert για τον πίνακα orologikos, exam και implements.

```
$sql="INSERT INTO exam
(ex_id, tipos)
VALUES
(NULL, 'Ορολογικός')";
$sql2="INSERT INTO
orologikos
(ex_id, widal, wright, MonoSpot, RPR, CMV, toksoplasma, erythra, HSV, EBV )
VALUES
(LAST_INSERT_ID(),
'$widal',
'$wright',
'$MonoSpot',
'$RPR',
'$CMV',
'$toksoplasma',
'$erythra',
'$HSV',
'$EBV')";
$sql3="INSERT INTO implements
 (p_id, ex_id, date_time)
VALUES
('$p_id', LAST_INSERT_ID(), CURRENT_TIMESTAMP) ";
```

4.3.23.5 Ουρολογικός έλεγχος

Τα αρχεία που χρησιμοποιούνται για την εισαγωγή αυτής της εξέτασης είναι τα έξης: form_ ourologikos.php, create_ ourologikos.php.

Παρακάτω φαίνεται ο κώδικας του αρχείου create_ ourologikos.php ο όποιος δέχεται από την φόρμα form_ ourologikos.php τα δεδομένα, κάνει τις κατάλληλες εργασίες και τα καταχωρεί στην βάση.

```
$p_id=$_POST['p_id'];
$xria= mysql_real_escape_string( $_POST['element_1'] );
$opsi= mysql_real_escape_string( $_POST['element_2'] );
$eidiko_baros= mysql_real_escape_string( $_POST['element_3']
);
$puosferia= mysql_real_escape_string( $_POST['element_4'] );
$eruthra= mysql_real_escape_string( $_POST['element_5'] );
```

Ο παρακάτω κώδικας δείχνει τα απαραίτητα SQL Insert για τον πίνακα ourologikos, exam και implements.

```
$sql="INSERT INTO exam
(ex_id, tipos)
VALUES (NULL, 'Ουρολογική')";

$sql2="INSERT INTO ourologiki (ex_id, xria, opsi, eidiko_baros, puosferia, eruthra)
VALUES (
LAST_INSERT_ID(),'$xria','$opsi','$eidiko_baros','$puosferia','$eruthra')";

$sql3="INSERT INTO implements
(p_id, ex_id, date_time)
VALUES
 ('$p_id', LAST_INSERT_ID(), CURRENT_TIMESTAMP) ";
```

4.3.23.6 Πηκτικός έλεγχος

Τα αρχεία που χρησιμοποιούνται για την εισαγωγή αυτής της εξέτασης είναι τα έξης: form_ piktikos.php, create_ piktikos.php.

Παρακάτω φαίνεται ο κώδικας του αρχείου create_ piktikos.php ο όποιος δέχεται από την φόρμα form_ piktikos.php τα δεδομένα, κάνει τις κατάλληλες εργασίες και τα καταχωρεί στην βάση.

```
$p_id=$_POST['p_id'];
$xronos_rohs= mysql_real_escape_string( $_POST['element_1'] );
$PT= mysql_real_escape_string( $_POST['element_2'] );
$PTT= mysql_real_escape_string( $_POST['element_3'] );
$id_plasmatos= mysql_real_escape_string( $_POST['element_4'] );
```

Ο παρακάτω κώδικας δείχνει τα απαραίτητα SQL Insert για τον πίνακα piktikos, exam και implements.

```
$sql="INSERT INTO exam
(ex_id, tipos)
VALUES
(NULL, 'Πηκτικός')";

$sql2="INSERT INTO piktikos (ex_id, xronos_rohs, PT, PTT, id_plasmatos)
VALUES
(LAST_INSERT_ID(),'$xronos_rohs','$PT','$PTT','$id_plasmatos')";

$sql3="INSERT INTO implements
 (p_id, ex_id, date_time)
VALUES
('$p_id', LAST_INSERT_ID(), CURRENT_TIMESTAMP) ";
```

4.3.23.7 Καταχώρηση Απεικονιστικής Εξέτασης

Η διαδικασία καταχώρησης μια απεικονιστικής εξέτασης διαφέρει λίγο από τις άλλες. Σε αυτού του είδους την εξέταση ο χρήστης δεν έχει να συμπληρώσει απλός μια φόρμα με στοιχειά, αλλά πρέπει να εισάγει και μια εικόνα η οποία θα μπορεί να είναι για παράδειγμα μια ακτινογραφία. Στην συνέχεια θα πρέπει να παραθέσει κάποια σχόλια σε ένα Text Field στο οποίο εμφανίζεται στην φόρμα εισαγωγής και να επιλέξει από ένα Drop Down List τον τύπο της εξέτασης.

Η εικόνα αυτή θα πρέπει με κάποιο τρόπο να διατηρείτε και να εμφανίζεται όταν την χρειαζόμαστε, η λύση η οποία επιλέξαμε είναι αυτή της αντιγραφής τις εικόνας σε κάποιο συγκεκριμένο Directory με την μέθοδο File Upload και στην συνέχεια καταχωρούμε τα στοιχεία της εικόνας καθώς τον τύπο και την περιγραφή της στην βάση δεδομένων.

Τα αρχεία τα οποία χρησιμοποιούνται είναι τα έξης: UploadForm.php και upload_file.php. Παρακάτω παραθέτουμε τον κώδικα από το αρχείο upload_file.php το οποίο δέχεται τα δεδομένα από το αρχείο UploadForm.php όπως αναφέραμε παραπάνω.

```
$name=$_FILES["file"]["name"];
$typos=$_POST["element_3"];
if ((($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] ==
"image/pjpeg"))
&& ($_FILES["file"]["size"] < 600000))
```

Στο παραπάνω κομμάτι ελέγχουμε τον τύπο της εικόνας ο όποιος θα πρέπει να είναι gif η jpeg (ο τύπος pjpeg είναι για να γίνεται ο έλεγχος και σε Internet explorer).

```
move_uploaded_file($_FILES["file"]["tmp_name"],
  "upload/" . $_FILES["file"]["name"]);
```

Στο παραπάνω κομμάτι γίνεται η αντιγραφή (Upload) της εικόνας στο Directory το οποίο έχουμε καθορίσει.

```
$sql="INSERT INTO exam (ex_id, tipos)
VALUES
(NULL, 'Απεικονιστική')";

$sql2="INSERT INTO apeikonistiki
 (ex_id, typos, perigrafi, location)
VALUES
(LAST_INSERT_ID(), '$typos' ,'$_POST[element_2]', '$name')";

$sql3="INSERT INTO implements
(p_id, ex_id, date_time)
VALUES
('$_POST[p_id]', LAST_INSERT_ID(), CURRENT_TIMESTAMP) ";
```

Στο παραπάνω κομμάτι βλέπουμε τα απαραίτητα Insert στην βάση ώστε να κρατήσουμε πληροφορίες για το όνομα του αρχείου τον τύπο του και την περιγραφή του.

4.3.24 Διαγραφή εξέτασης

Η διαγραφή μιας εξέτασης ενός ασθενούς μπορεί να γίνει μόνο από τον διαχειριστή του συστήματος κάνοντας κλικ στο αντίστοιχο link του μενού της εφαρμογής. Για λόγους ασφαλείας και αξιοπιστίας των εξέτασεων τέτοια διαγραφή δεν μπορεί να κάνει ο γιατρός.

Επιλέγοντας αυτή τη λειτουργία ο διαχειριστής καλείται να επιλέξει τον ασθενή για τον οποίο θέλει να διαγράψει την εξέτασης. Η μέθοδος έχει αναλυθεί ήδη και είναι αυτή της απλής αναζήτησης. Το σύστημα ανταποκρίνεται με μια λίστα από αποτελέσματα ασθενών με βάση το κριτήριο αναζήτησης. Ο χρήστης επιλέγει τον ασθενή που θέλει και η εφαρμογή εμφανίζει όλες τις εξέτασεις του συγκεκριμένου ασθενούς. Έπειτα ο χρήστης πατάει το κουμπί της διαγραφής δίπλα από τη εξέταση που θέλει να σβήσει. Το σύστημα ανταποκρίνεται με μήνυμα επιβεβαίωσης της ενέργειας και μετά από θετική απάντηση του χρήστη σβήνεται η εξέταση.

Για την επιτευξη της παραπάνω ενέργειας χρησιμοποιούνται τέσσερα αρχεία. search_patient_to_delete_exam.php, patient_results_delete_exam.php, patient_exams.php, delete_exam.php. παρακάτω παρουσιάζεται το ερώτημα της διαγραφής, μάλιστα σε αυτήν την περίπτωση υπάρχουν και διαφορετικά SQL ερωτήματα τα οποία αντιστοιχούν σε κάθε εξέταση, έτσι ανάλογα με τον τύπο της εξέτασης που διαγράφει ο χρήστης εκτελείτε και διαφορετικό SQL Query, ο τύπος καθορίζεται από την μεταβλητή $tipos η οποία ελέγχεται μέσα σε μια if συνθήκη.

```
$ex_id=$_POST['ex_id'];
$p_id=$_POST['p_id'];
$date_time=$_POST['date_time'];
$tipos=$_POST['tipos'];
$sql="DELETE FROM implements WHERE ex_id =$ex_id and p_id=$p_id
and date_time='$date_time'";
if($tipos=='Αιματολογική')
$sql2="DELETE FROM aimatologiki WHERE ex_id =$ex_id";
elseif($tipos=='Απεικονιστική')
$sql2="DELETE FROM apeikonistiki WHERE ex_id =$ex_id";
elseif($tipos=='Βιοχημικός')
$sql2="DELETE FROM bioximikos WHERE ex_id =$ex_id";
elseif($tipos=='Ορμονικός')
$sql2="DELETE FROM ormonikos WHERE ex_id =$ex_id";
elseif($tipos=='Ορολογικός')
$sql2="DELETE FROM orologikos WHERE ex_id =$ex_id";
elseif($tipos=='Ουρολογική')
$sql2="DELETE FROM ourologiki WHERE ex_id =$ex_id";
elseif($tipos=='Πηκτικός')
$sql2="DELETE FROM piktikos WHERE ex_id =$ex_id";
$sql3="DELETE FROM exam WHERE ex_id =$ex_id";
```

Στην συνεχεία εκτελούνται τα sql2 και sql3 ώστε να γίνουν οι απαραίτητες εργασίες στην βάση. Με την επιτυχή διαγραφή της εξέτασης εμφανίζεται αντίστοιχο μήνυμα.

4.3.25 Σύνδεση στο σύστημα και ασφάλεια (Log-In & Security)

Όπως έχουμε ήδη αναφέρει, στο σύστημα μας έχουμε δυο ειδών χρηστές με διαφορετικά δικαιώματα ο καθένας.

Οι χρηστές αυτοί είναι: ο Διαχειριστής του συστήματος ο όποιος έχει πλήρη δικαιώματα και ο γιατρός ο όποιος έχει περιορισμένα δικαιώματα. κάθε χρήστης έχει ένα username και ένα password ώστε να μπορεί να συνδεθεί στο σύστημα και να μπορέσει να πραγματοποιήσει κάποιες εργασίες.

Ο τρόπος με τον οποίο διαχωρίζονται οι χρηστές στην βάση δεδομένων έχει να κάνει με ένα πεδίο το οποίο αναφέρεται ως access level και βρίσκεται στον πινάκα user. Αυτό το πεδίο καθορίζει αν ένας χρήστης είναι διαχειριστής, γιατρός η

γιατρός ο όποιος δεν μπορεί να κάνει log-in. Οι τιμές του access level μπορούν να πάρουν τρεις τιμές οι οποίες είναι οι έξης: για τον διαχειριστή η τιμή 2 για τον γιατρό η τιμή 1 και για τον ανενεργό γιατρό η τιμή 0.

Τα αρχεία τα οποία χρησιμοποιούνται για να συνδεθεί κάποιος στο σύστημα είναι τα έξης: main_login.php και checklogin.php.

Το main_login.php. είναι μια φόρμα στην οποία εισάγει ο χρήστης τα στοιχειά του και στην συνέχεια με το πάτημα του κουμπιού σύνδεση στέλνονται στο αρχείο check_login.php.

Παρακάτω αναφέρονται κομμάτια κώδικα του check_login.php και μια μικρή περιγραφή για το πώς αυτά λειτουργούν.

```
$myusername = stripslashes($myusername);
$mypassword = stripslashes($mypassword);
$myusername = mysql_real_escape_string($myusername);
$mypassword = mysql_real_escape_string($mypassword);
$mypassword = md5($mypassword);
```

Εδώ βλέπουμε τις μεταβλητές myusername και mypassword να περνάνε από κατάλληλες συναρτήσεις όπως έχουμε αναφέρει σε άλλο κεφάλαιο.

```
$sql2=mysql_query("SELECT * FROM $tbl_name WHERE
username='$myusername' and password='$mypassword'");
$sql="SELECT * FROM $tbl_name WHERE username='$myusername'
and password='$mypassword'";
$sql3=mysql_query("SELECT * FROM doctor WHERE
username='$myusername'");
```

Στο παραπάνω κομμάτι κώδικα δημιουργούμε τα απαραίτητα SQL ερωτήματα ώστε να ελέγξουμε αν υπάρχει κάποιος χρήστης στον πινάκα user και αναλόγως να προβούμε στις κατάλληλες ενέργειες.

```
session_register("myusername");
session_register("mypassword");
$_SESSION['level'] =$level;
$_SESSION['onoma'] =$onoma;
$_SESSION['epwnimo'] =$epwnimo;
$_SESSION['d_id'] =$d_id;
```

Αν το username και το password περάσουν τον έλεγχο τότε δημιουργούμε μια καινούργια Session ώστε κάθε ενεργεία του χρηστή να χαρακτηρίζεται από αυτόν. Με αυτόν τον τρόπο μπορούμε να ορίσουμε μεταβλητές οι οποίες είναι Global εμβέλειας και να έχουμε πρόσβαση στα στοιχειά του χρηστή η ότι άλλο μπορεί να χαρακτηρίζει αυτή την Session.

```
if($level==2){
header("location:../main_admin/main_admin.php");
}
elseif($level==1){
header("location:../main_doctor/main_doctor.php");
}
```

Παραπάνω φαίνεται ο τρόπος με τον οποίο ο κάθε χρήστης πηγαίνει ανάλογα με το access level του στην σελίδα στην οποία έχει δικαίωμα και επιτρέπεται να έχει πρόσβαση.

Εδώ θα πρέπει να αναφέρουμε ότι σε κάθε σελίδα την οποία χρησιμοποιεί ένας χρήστης έχουμε προσθέσει ένα κομμάτι κώδικα το οποίο ελέγχει για λογούς ασφάλειας το access level του χρηστή. Σε περίπτωση που δεν έχει δικαίωμα πρόσβασης εμφανίζεται μήνυμα λανθασμένης πρόσβασης στην συγκεκριμένη σελίδα. Με αυτόν τον τρόπο μπορούμε να αποτρέψουμε έναν γιατρό να εκτελέσει εργασίες administrator ακόμα και αν επιχειρήσει να πληκτρολογήσει στον browser απευθείας το link κάποιας φόρμας του administrator. Ο κώδικας αυτός αναφέρεται παρακάτω.

```
session_start();
if(!session_is_registered('myusername') || $_SESSION['level']!=2)
header("location:../includes/noaccess.php");
```

4.3.26 Αλλαγή κωδικού πρόσβασης ιατρού από διαχειριστή

Με την λειτουργία αυτή ο διαχειριστής έχει τη δυνατότητα να αντικαταστήσει τον κωδικό πρόσβασης του ιατρού χωρίς να χρειάζεται να γνωρίζει τον παλιό κωδικό πρόσβασης. Έτσι ο ιατρός μπορεί να αποκαταστήσει την σύνδεση του στην εφαρμογή σε περίπτωση απώλειας του κωδικού πρόσβασης.

Για να εκτελεστεί η αλλαγή του κωδικού πρόσβασης του ιατρού πρέπει ο διαχειριστής να επιλέξει από το κύριο μενού το σχετικό κουμπί. Το σύστημα ανταποκρίνεται παραπέμποντας τον διαχειριστή να κάνει αναζητήσει τον ιατρό για τον οποίο θέλει να κάνει αλλαγή του κωδικού πρόσβασης. Έπειτα ακολουθούν τα αποτελέσματα και ο διαχειριστής επιλέγει το ιατρό που επιθυμεί. Εμφανίζεται μια φόρμα με το όνομα του γιατρού και το όνομα χρήστη για το σύστημα. Ο διαχειριστής συμπληρώνει τον επιθυμητό κωδικό πρόσβασης και πατά το κουμπί της αλλαγής κωδικού. Το σύστημα έπειτα ανταποκρίνεται με το κατάλληλο μήνυμα.

Για την διαδικασία που περιγράφεται παραπάνω χρησιμοποιούνται τέσσερα αρχεία. Η σειρά που εκτελούνται είναι και η σειρά με την οποία αναφέρονται. Τα αρχεία αυτά είναι: search_doctor_change_pass.php, doctor_results_change_pass.php, form_change_pass_doctor.php, change_doctor_password.php. Τα δύο πρώτα αρχεία είναι παρόμοια με την απλή αναζήτηση. Το form_change_pass_doctor.php χρησιμοποιείται για την εισαγωγή του νέου κωδικού πρόσβασης. Το change_doctor_password.php χρησιμοποιείται για την εισαγωγή του νέου κωδικού πρόσβασης στη βάση δεδομένων και είναι αυτό που θα αναλύσουμε σε αυτή την ενότητα.

```
$sql="UPDATE user SET password='$password'
WHERE username='$username'";
```

Η εντολή που παρουσιάζεται παραπάνω είναι η βασική εντολή αυτού του αρχείου. Ο νέος κωδικός πρόσβασης καταχωρείται στη μεταβλητή $password

αφού γίνει πρώτα έλεγχος για php injection. Γνωστό είναι και το username για το οποίο γίνεται η αλλαγή αφού αποστέλλεται μαζί με τους νέους κωδικούς πρόσβασης από την φόρμα.

Για την εκτέλεση του παραπάνω ερωτήματος χρησιμοποιείται ο κώδικας που ακολουθεί. Αν είναι επιτυχής παίρνουμε ένα σχετικό μήνυμα ενώ αν αποτύχει παίρνουμε το σφάλμα που προέκυψε.

```
if (!mysql_query($sql,$con)){
        die('Error: ' . mysql_error());
}
else{
        echo "<h3>Η αλλαγή κωδικού πρόσβασης ήταν επιτυχής</h3> ";
}
```

4.3.27 Αλλαγή κωδικού πρόσβασης χρήστη

Για λόγους ασφαλείας ο χρήστης είτε είναι ιατρός είτε είναι διαχειριστής θα πρέπει να αλλάζει τακτικά τον κωδικό πρόσβασης του στην εφαρμογή. Αυτό μπορεί να το κάνει επιλέγοντας αλλαγή κωδικού πρόσβασης από το κύριο μενού της εφαρμογής.

Για να εκτελεστεί αυτή η λειτουργία η εφαρμογή εμφανίζει μία φόρμα στην οποία παρουσιάζονται τα βασικά στοιχεία του χρήστη και τρία πεδία σχετικά με τους κωδικούς πρόσβασης. Το πρώτο πεδίο ζητά τον παλιό κωδικό πρόσβασης το δεύτερο ζητά τον νέο κωδικό πρόσβασης και το τρίτο ζητά και πάλι το νέο κωδικό πρόσβασης. Αν οι νέοι κωδικοί είναι ίδιοι προχωρά η διαδικασία αλλιώς γίνεται πάλι προσπάθεια.

Για την διαδικασία που περιγράφεται παραπάνω απαιτούνται δυο αρχεία τα οποία αναφέρονται με τη σειρά που εκτελούνται: form_change_pass_by_ doctor.php, update_pass_by_dr.php. Σε αυτή την ενότητα θα δείξουμε τον τρόπο λειτουργίας του update_pass_by_dr.php που εκτελεί το update ερώτημα που παρουσιάζεται παρακάτω.

Για την λειτουργία χρησιμοποιούνται τρεις μεταβλητές οι οποίες έιναι και τα κύρια χαρακτηριστικά της λειτουργίας. Αυτές είναι:

```
$d_id=$_GET['d_id'];
$given_old= mysql_real_escape_string( $_POST['element_2'] );
$new= mysql_real_escape_string( $_POST['element_3'] );
```

Το d_id είναι το κύριο κλειδί του γιατρού, το given_old είναι ο παλιός κωδικός πρόσβασης και το new είναι ο νέος κωδικός πρόσβασης. Βρίσκεται ο παλιός κωδικός πρόσβασης και το όνομα χρήστη με τη βοήθεια sql ερωτημάτων στην όπως φαίνεται παρακάτω.

```
$result = mysql_query("SELECT * FROM doctor WHERE d_id=$d_id");

while($row = mysql_fetch_array($result)){
        $username=$row['username'];
 }

$result2 = mysql_query("SELECT * FROM user WHERE
username='$username'");

while($row2 = mysql_fetch_array($result2)){
        $old=$row2['password'];
 }
```

Τα αποτελέσματα των ερωτημάτων καταχωρούνται σε μεταβλητές. Και εκτελείται ο κώδικας που φαίνεται παρακάτω

```
if($old==$given_old){

echo "<h3>Ο παλιός κωδικός πρόσβασης που δόθηκε είναι σωστός</h3>
<br />";

mysql_query("UPDATE user SET password = '$new' WHERE username =
'$username' AND password = '$old'");

echo "<h3>Η αλλαγή του password ήταν επιτυχής</h3> <br />";

}
else{
echo "<h3>Ο παλιός κωδικός πρόσβασης που δόθηκε είναι λάθος</h3> <br
/>";
echo '<a align="center" href="form_change_pass_by_doctor.php?&d_id=
'.$d_id.'"><input type="button" value="Προσπαθήστε ξανά"></a>';
}
```

Το παραπάνω τμήμα κώδικα ελέγχει τον κωδικό πρόσβασης που έδωσε ο χρήστης σαν παλιό με τον παλιό κωδικό πρόσβασης από τη βάση δεδομένων. Αν είναι ίδιοι προχωρά στην αλλαγή αν όχι παραπέμπει τον χρήστη να δοκιμάσει ξανά. Το ερώτημα σχετικά με την αλλαγή είναι αυτό που φαίνεται μέσα στη συνάρτηση mysql_query().

4.3.28 Στατιστικά

Η προβολή των στατιστικών μπορεί να γίνει από το διαχειριστή του συστήματος από το κεντρικό μενού πατώντας το σχετικό κουμπί στο κεντρικό μενού της εφαρμογής. Το σύστημα ανταποκρίνεται ζητώντας από το χρήστη να εισάγει την ημερομηνία για την οποίο θέλει να δει τις καταμετρήσεις σχετικά με τις εξετάσεις και διαγνώσεις. Έπειτα επιστρέφεται ένα πίνακας με τον αριθμό των διαγνώσεων των εξετάσεων γενικά και των εξετάσεων ανά κατηγορία.

Για να γίνει αυτό η εφαρμογή μετρά όλα τα ex_id του πίνακα implements για την συγκεκριμένη ημερομηνία. Από αυτά τα ex_id μετρά κάθε εξέταση ανά τύπο με τη βοήθεια μετρητών. Για την επιτευξη των παραπάνω χρησιμοποιούνται τα αρχεία: form_statistics.php και view_statistics.php.

4.3.29 Ελληνικοι χαρακτηρες στην PHP & MYSQL

Για να μπορέσει να υποστηρίξει το σύστημα μας ελληνικούς χαρακτήρες, τόσο στην βάσει όσο και στην PHP, έχουν γίνει κάποιες ρυθμίσεις οι οποίες αναφέρονται παρακάτω:

- PHP-HTML

  Σε κάθε αρχείο PHP έχουμε εισάγει το παρακάτω κομμάτι κώδικα μετά από την σύνδεση στην βάσει δεδομένων

```
mysql_query("set names 'greek'");
```

  Με αυτόν τον τρόπο υποστηρίζονται τα ελληνικά σε κάθε ερώτημα το οποίο εκτελείτε μέσω της PHP.

  Ακόμη σε κάθε φόρμα HTML έχουμε τον έξης κώδικα ώστε να υποστηρίζονται τα ελληνικά σε όλες τις φόρμες:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

- MYSQL

  Κατά την δημιουργία της βάσης χρησιμοποιήσαμε collation greek_general_ci , όπως επίσης και το MYSQL connection collation είναι και αυτό greek_general_ci.

  Ακόμη εχουν γίνει κάποιες επιπρόσθετες αλλαγές στα αρχεία του wamp server

- Στο αρχείο ρυθμίσεων της MYSQL έχουμε προσθέσει την εντολή:

default-character-set=greek

- Στο αρχείο php.ini έχουμε προσθέσει: default_charset=”iso-8859-7”
- Στο αρχείο httpd.ini του apache server: Φέραμε πρώτα στη γραμμή language priority το el και προσθέσαμε την γραμμή AddDefaultCharset iso-8859-7

4.4 SQL Injections

Με τον όρο SQL Injections αναφερόμαστε στην περίπτωση που ένας χρήστης χρησιμοποιεί στα πεδία μιας φόρμας χαρακτήρες αποφυγής (Escape Characters) οι οποίοι στην συνέχεια χρησιμοποιούνται από κάποιο SQL ερώτημα (SQL Statement).

Με αυτόν τον τρόπο ο τελικός χρήστης μιας εφαρμογής μπορεί να επιφέρει ανεπιθύμητα αποτελέσματα στην βάση δεδομένων της εφαρμογής , παρακάτω αναφέρονται μερικές περιπτώσεις SQL Injection και πως μπορούμε να τις αποφύγουμε.

Υπάρχουν διάφορες κατηγορίες οι SQL Injection οποίες αναφέρονται παρακάτω:

- Λανθασμένος έλεγχος χαρακτήρων διαφυγής (Incorectly Filtered Escape Character)
- Λανθασμένη πληκτρολογηση (Incorrect type handling)
- Χρηση Magic String (Login Page)

Παράδειγμα:

```
<?php
// Query database to check if there are any matching users
$query = "SELECT * FROM users WHERE user='{$_POST['username']}'
 AND password='{$_POST['password']}'";
mysql_query($query);

// We didn't check $_POST['password'], it could be anything the user wan
ted! For example:
$_POST['username'] = 'aidan';
$_POST['password'] = "' OR ''='";

// This means the query sent to MySQL would be:
echo $query;
?>
```

Στο παραπάνω παράδειγμα το ερώτημα το οποίο θα σταλεί τελικά στην SQL θα είναι το εξής (το οποίο είναι και αποτέλεσμα τις echo $query):

```
SELECT *
FROM users
WHERE user='aidan' AND password='' OR ''=''
```

Με αυτόν τον τρόπο μπορεί οποιοσδήποτε να κάνει σύνδεση (Log-In) στο σύστημα χωρίς να έχει κωδικό πρόσβασης (password).

Επεξήγηση μεθόδων της PHP που χρησιμοποιούνται στην Εφαρμογή

4.4.1 Χρήση μεθόδου για αποφυγή SQL Injections στην PHP

Στην εφαρμογή μας πριν κάνουμε οποιαδήποτε διεργασία που αφορά την βάση δεδομένων η οποία περιέχει στοιχειά που έχουν εισαχθεί από τον χρήστη χρησιμοποιούμε την μέθοδο mysql_real_escape_string() ώστε να αποφύγουμε τα SQL Injections.

Η Περιγραφή της μεθόδου mysql_real_escape_string() αναφερετε παρακάτω:

```
mysql_real_escape_string ( string $unescaped_string [, resource $link_identifier ] )
```

Τι είναι η mysql_real_escape_string(); Αυτή η έκφραση καλεί την μέθοδο mysql_real_escape_string από την βιβλιοθήκη της MySQL η οποία αποτρέπει τον χαρακτήρα \ (backslash) πριν από τους χαρακτήρες όπως φαίνεται παρακάτω:*x00*, *n*, *r*, \, *'*, *"* and *x1a.*

Αυτή η συνάρτηση χρησιμοποιείτε ώστε να εισάγουμε ασφαλή δεδομένα πριν την εκτέλεση ενός MySQL ερωτήματος.

Ένα απλό παράδειγμα για το πώς χρησιμοποιείτε φαίνεται στο παρακάτω κομμάτι κώδικα :

```
<?php
// Connect
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')
   OR die(mysql_error());

// Query
$query = sprintf("SELECT * FROM users WHERE user='%s' AND password='%
s'",
         mysql_real_escape_string($user),
         mysql_real_escape_string($password));
?>
```

4.4.2 Η μέθοδος trim()

Σύνταξη:

```
trim ( string $str [, string $charlist ] )
```

Με την χρήση αυτής της συνάρτησης, μπορούμε να εισάγουμε ένα String και να μας επιστρέψει το String (στην περίπτωση που δεν έχουμε ορίσει το πεδίο string $charlist) χωρίς να έχει μέσα επαναλαμβανόμενους τους παρακάτω χαρακτήρες:

- " " (ASCII 32 (0x20)), ο χαρακτήρας κενό (an ordinary space).
- "\t" (ASCII 9 (0x09)), a tab.
- "\n" (ASCII 10 (0x0A)), a new line (line feed).
- "\r" (ASCII 13 (0x0D)), a carriage return.
- "\0" (ASCII 0 (0x00)), Κενο Byte (the NUL-byte).
- "\x0B" (ASCII 11 (0x0B)), a vertical tab.

Ας δούμε ένα παράδειγμα παρακάτω:

```
<?php

$text   = "\t\tThese are a few words :) ...  ";
$binary = "\x09Example string\x0A";
$hello  = "Hello World";
var_dump($text, $binary, $hello);

print "\n";

$trimmed = trim($text);
var_dump($trimmed);

$trimmed = trim($text, " \t.");
var_dump($trimmed);

$trimmed = trim($hello, "Hdle");
var_dump($trimmed);

// trim the ASCII control characters at the beginning and end of $binary
// (from 0 to 31 inclusive)
$clean = trim($binary, "\x00..\x1F");
var_dump($clean);

?>
```

Η έξοδος της εκτέλεσης του παραπάνω παραδείγματος φαίνεται παρακάτω, μπορούμε να παρατηρήσουμε ότι τα Strings τα οποία έχουν «περάσει» από την συνάρτηση trim() δεν εμφανίζουν επανάληψη των χαρακτήρων που αναφέραμε, με αυτόν τον τρόπο μπορούμε και αποφεύγουμε τις περιττές επαναλήψεις.

Έξοδος:

```
string(32) "        These are a few words :) ...  "
string(16) "    Example string
"
string(11) "Hello World"

string(28) "These are a few words :) ..."
string(24) "These are a few words :)"
string(5) "o Wor"
string(14) "Example string"
```

4.4.3 Η μέθοδος strtoupper()

```
strtoupper ( string $string )
```

Με την χρήση αυτής της συνάρτησης, μπορούμε να εισάγουμε ένα String και να μας επιστρέψει το String μετατρεποντας ολους τους χαρακτηρες σε κεφαλαιους.

Ας δούμε ένα παράδειγμα παρακάτω:

```
<?php
 $str = "Mary Had A Little Lamb and She LOVED It So";
 $str = strtoupper($str);
 echo $str;
 ?>
```

Μετά από την εκτέλεση του παραπάνω PHP κώδικα θα έχουμε την έξοδο που έχουμε παρακάτω , μπορούμε να παρατηρήσουμε ότι όλοι οι χαρακτήρες του String έχουν μετατραπεί από μικρούς σε κεφαλαίους.

```
MARY HAD A LITTLE LAMB AND SHE LOVED IT SO
```

4.4.4 Η μέθοδος strip_tags()

```
strip_tags ( string $str [, string $allowable_tags ] )
```

Με την χρήση αυτής της συνάρτησης, μπορούμε να εισάγουμε ένα String και να μας επιστρέψει το String αποφεύγοντας όλα τα HTML και PHP tags, σε περίπτωση που θέλουμε να επιτρέπεται η εμφάνιση κάποιου tag τότε το προσθέτουμε στο πεδίο $allowable_tags.

Ας δούμε ένα παράδειγμα παρακάτω:

```
<?php
$text = '<p>Test paragraph.</p><!-- Comment --
> <a href="#fragment">Other text</a>';
echo strip_tags($text);
echo "\n";

// Allow <p> and <a>
echo strip_tags($text, '<p><a>');
?>
```

Μετά από την εκτέλεση του παραπάνω PHP κώδικα θα έχουμε την έξοδο που έχουμε παρακάτω , μπορούμε να παρατηρήσουμε ότι όλοι οι χαρακτήρες του String οι οποίοι είναι PHP η HTML tag αγνοούνται, εκτός από την δεύτερη περίπτωση που επιτρέπουμε κάποιους από αυτούς, ας δούμε την έξοδο παρακάτω:

```
Test paragraph. Other text

<p>Test paragraph.</p> <a href="#fragment">Other text</a>
```

4.4.5 Η μέθοδος md5()

```
string md5 ( string $str [, bool $raw_output= false ] )
```

Με την χρήση αυτής της συνάρτησης, μπορούμε να εισάγουμε ένα String και να μας επιστρέψει το String υπολογίζοντας το MD5 Hash χρησιμοποιώντας τον αλγόριθμο RSA Data Security, Inc. MD5 Message-Digest Algorithm .

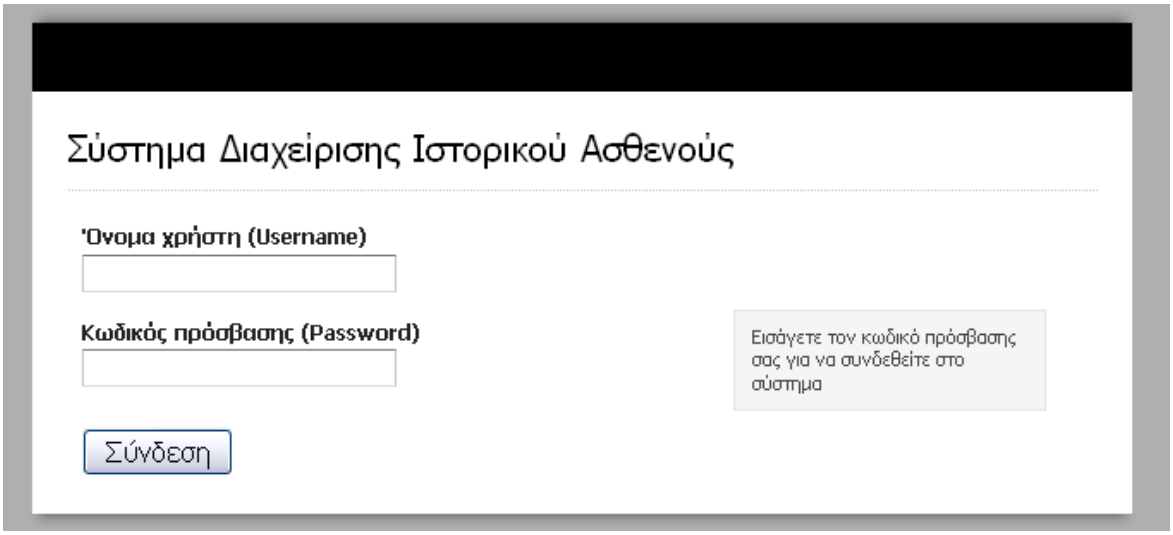
Ας δούμε ένα παράδειγμα παρακάτω:

```
<?php
$str = 'apple';

if (md5($str) === '1f3870be274f6c49b3e31a0c6728957f') {
    echo "Would you like a green or red apple?";
    exit;
}
?>
```

Μετά από την εκτέλεση του παραπάνω PHP κώδικα θα έχουμε την εμφάνιση του κειμένου: *Would you like a green or red apple?* Στην περίπτωση που ο υπολογισμός του md5 hash βγάλει το αποτέλεσμα το οποίο περιέχεται μέσα στην if.

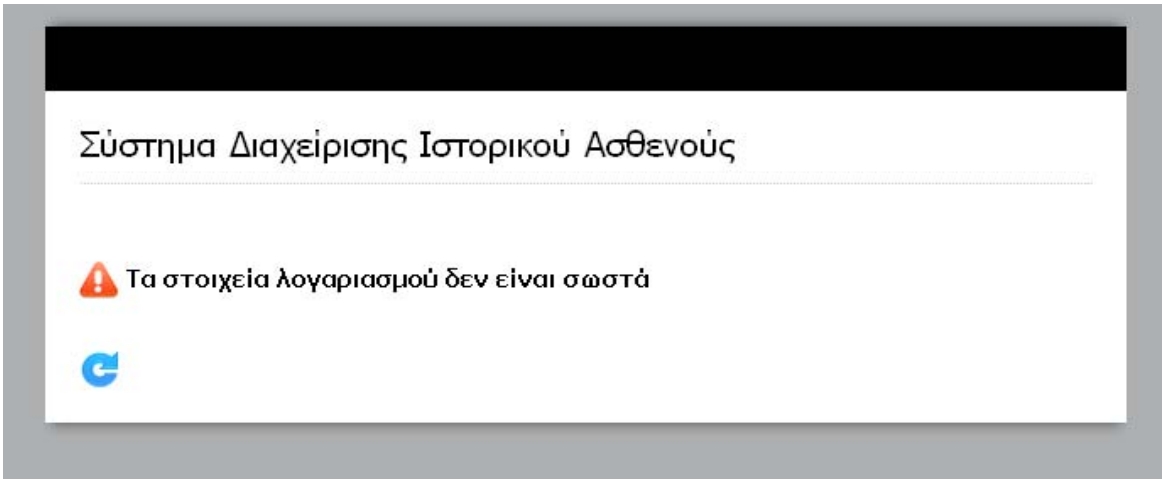
Κεφάλαιο 5: Εγχειρίδιο Χρήσης

5.1 Σύνδεση στο σύστημα



5.1 Φόρμα Σύνδεσης (log-in).

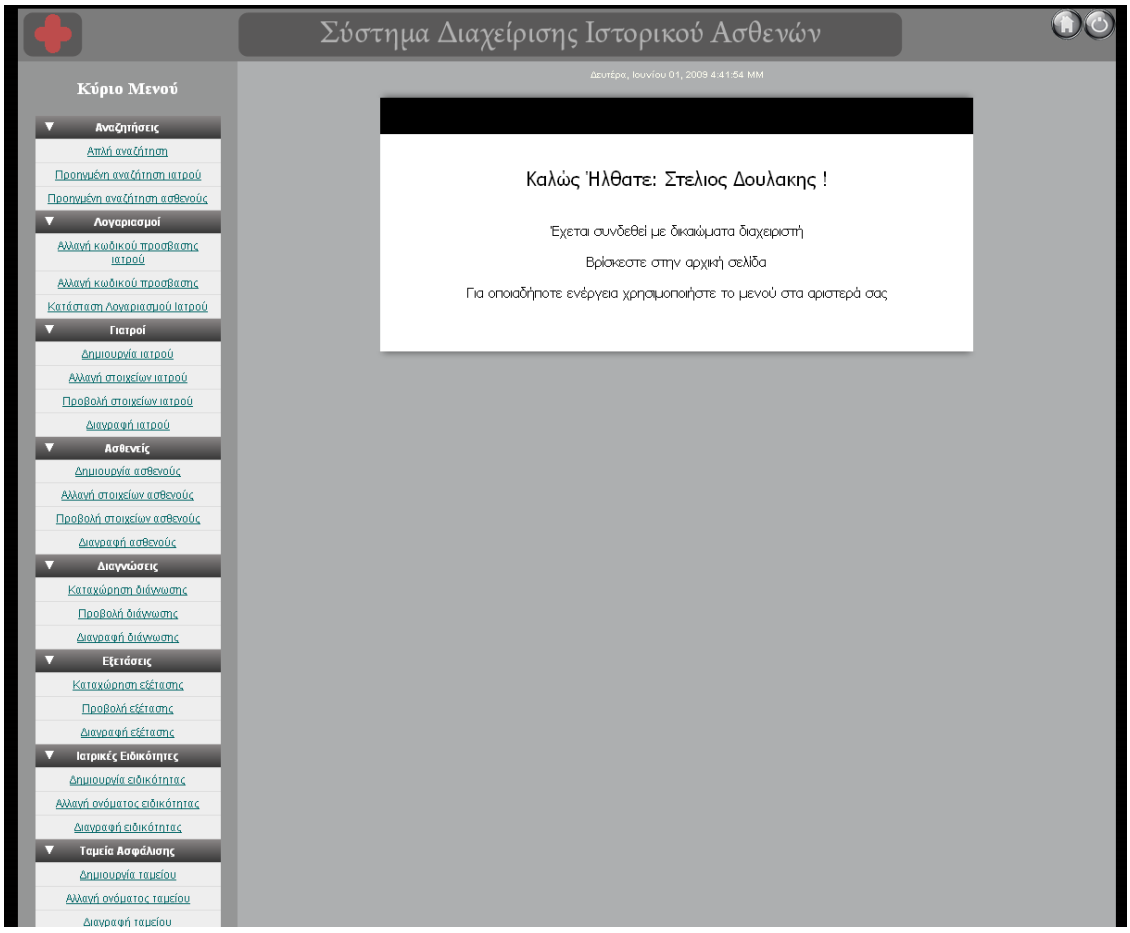
Στην εικόνα 5.1 βλέπουμε την αρχική σελίδα στην οποία ο χρήστης πρέπει να πληκτρολογήσει ένα όνομα χρήστη (username)και έναν κωδικό πρόσβασης (password) ώστε να μπορέσει να κάνει τις κατάλληλες εργασίες το σύστημα.



5.2 Μήνυμα σφάλματος μετά από αποτυχία σύνδεσης.

Σε περίπτωση που ο χρήστης πληκτρολογήσει εσφαλμένο όνομα χρήστη η κωδικό πρόσβασης τότε εμφανίζεται το παραπάνω μήνυμα και με το μπλε κουμπί επιστρέφει στην βασική σελίδα σύνδεσης (εικόνα 5.1).

Σε περίπτωση που γίνει επαλήθευση του κωδικού και του ονόματος χρήστη από το σύστημα τότε ο χρήστης έχει την δυνατότητα να δει την αρχική σελίδα και να εκτελέσει τις ενέργειες που επιθυμεί, παρακάτω βλέπουμε την αρχική σελίδα του διαχειριστή του συστήματος ο όποιος έχει πρόσβαση σε όλες τις επιλογές καθώς έχει πλήρες μενού, παρακάτω βλέπουμε την αρχική σελίδα του Διαχειριστή.

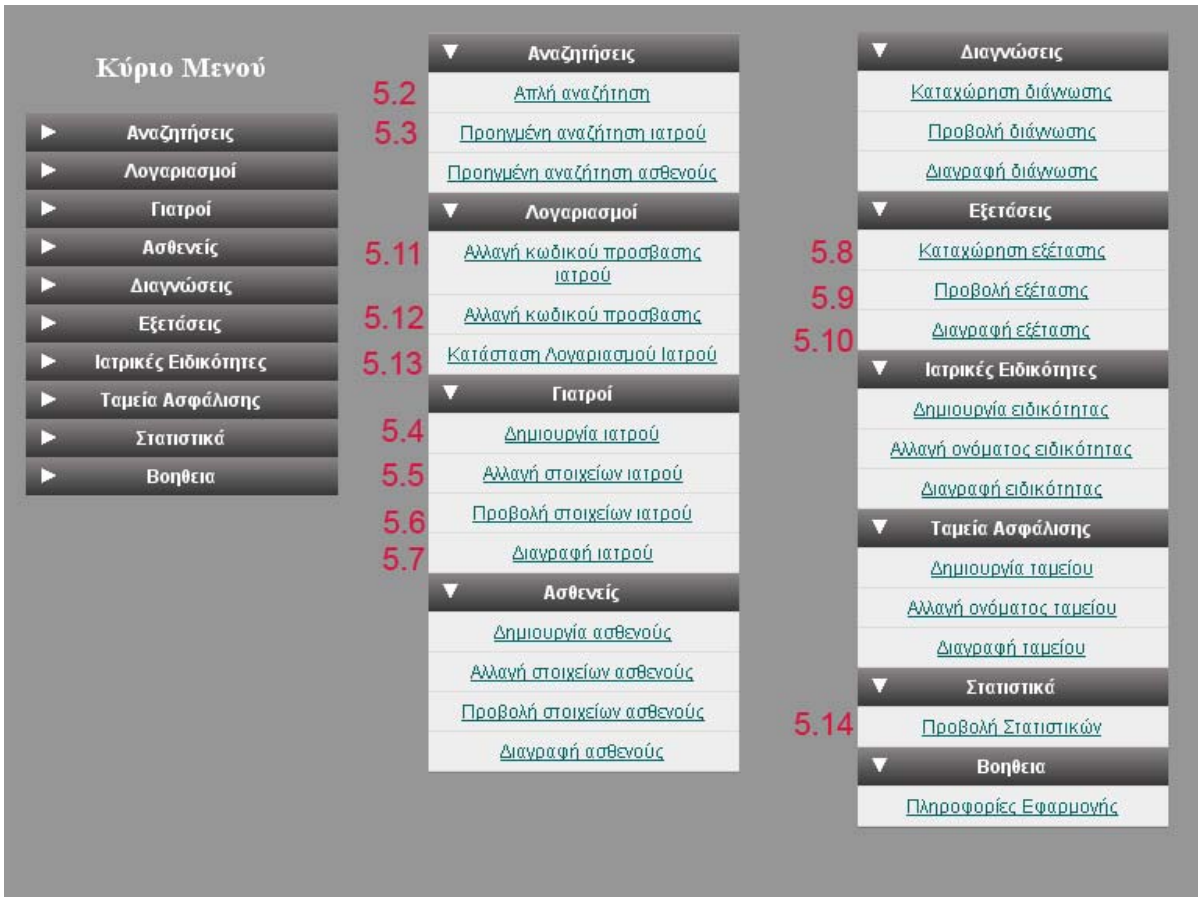


5.3 Αρχική σελίδα Διαχειριστή συστήματος.

Με την επιτυχή σύνδεση στο σύστημα ο χρήστης βλέπει την αρχική σελίδα (εικόνα 5.3)η οποία καλωσορίζει τον χρήστη εμφανίζοντας το άνομα του. Σε αυτήν την σελίδα παρατηρούμε στο αριστερό μέρος το κυρίως μενού από το οποίο ο χρήστης εκτελεί όλες τις απαραίτητες ενέργειες.

Στην πάνω αριστερά πλευρά παρατηρούμε δυο κουμπιά. Το δεξί κουμπί είναι για να αποσυνδέει τον χρήστη από το σύστημα και το αριστερό για να τον επιστρέφει στην αρχική του σελίδα.

Παρακάτω παραθέτουμε το βασικό μενού του συστήματος από το οποίο, όπως έχουμε αναφέρει, ο χρήστης επιλεγεί την εργασία την οποία θέλει να κάνει. Το μενού στην παρακάτω εικόνα (εικόνα 5.4) είναι αριθμημένο και κάθε αριθμός αντιστοιχεί σε μια λειτουργιά του μενού η οποία παραπέμπει στην αντίστοιχη ενότητα αυτού του κεφαλαίου, όσες εργασίες δεν είναι αριθμημένες είναι ακριβώς ίδιες με κάποιες από τις οποίες θα εξηγήσουμε παρακάτω.

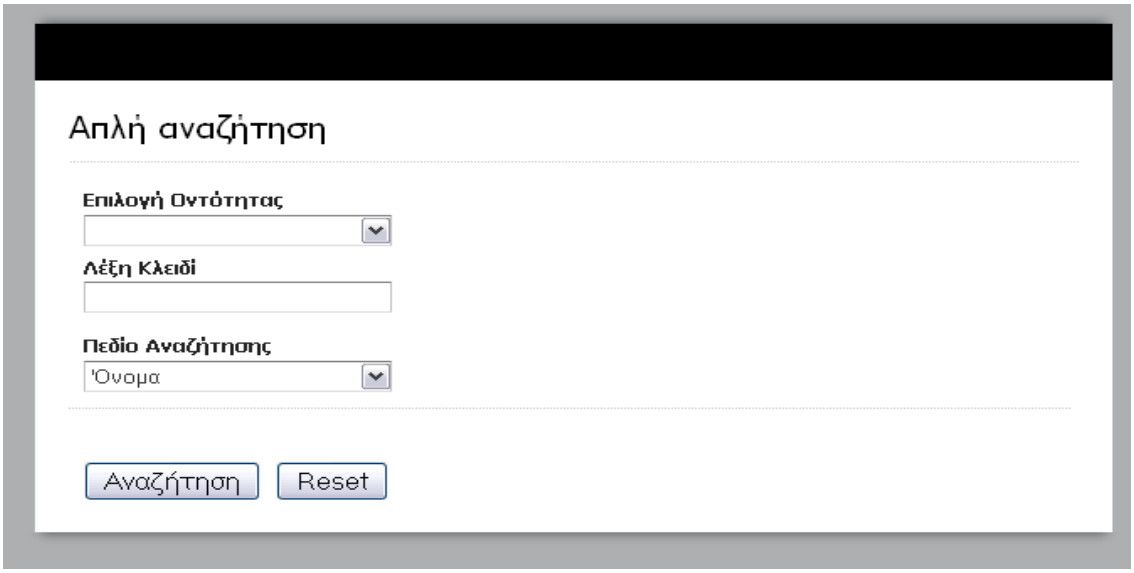


5.4 Βασικό μενού.

5.2 Απλή αναζήτηση

Μια από τις βασικές εργασίες τις οποίες εκτελούνται στο σύστημα είναι η αναζήτηση. Ο χρήστης όταν επιλέξει από του μενού την απλή αναζήτηση του

εμφανίζεται μια φόρμα στην οποία θα ρέπει να επιλέξει αν θέλει να αναζητήσει κάποιον γιατρό η κάποιον ασθενή όπως φαίνεται στην παρακάτω εικόνα στο πεδίο επιλογή οντότητας (εικόνα 5.5).



5.5 Απλή Αναζήτηση.

Στην περίπτωση που αναζητήσουμε ασθενή, εμφανίζονται τα παρακάτω αποτελέσματα όπως φαίνονται στην εικόνα 5.6, σε αυτήν την περίπτωση έχουμε 4 επιλογές (προβολή, αλλαγή-τροποποίηση, διαγραφή και προβολή ιστορικού), οι οποίες αναλύονται παρακάτω.

5.6 Αποτελέσματα απλής αναζήτησης ασθενή.

Οι τρεις πρώτες επιλογές έχουν να κάνουν με την προβολή, τροποποίηση και διαγραφή ενός ασθενή(οι οποίες αναλύονται στης ενότητες 5.5, 5.6 και 5.7 αντίστοιχα), η 4^{η} επιλογή προβάλει όλες τις εξετάσεις και τις διαγνώσεις οι οποίες

υπάρχουν καταχωρημένες στο σύστημα και φαίνονται στην παρακάτω εικόνα (εικόνα 5.7).

5.7 Προβολή ιστορικού ασθενούς.

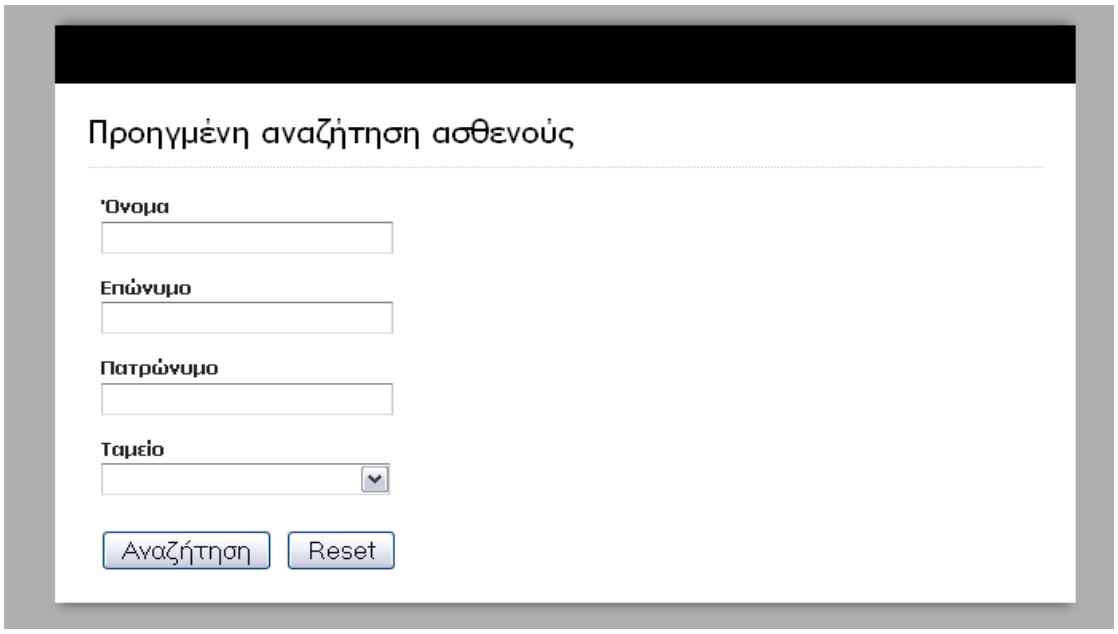
Στην παραπάνω εικόνα με το πάτημα του κουμπιού προβολή διπλά από κάθε εξέταση η διάγνωση μπορούμε να δούμε αναλυτικές πληροφορίες για την κάθε εγγραφή.

Στην περίπτωση που κάνουμε απλή αναζήτηση ιατρού, η διαδικασία είναι ακριβώς ιδία με αυτήν που περιγράφεται παραπάνω για τον ασθενή με την μόνη διάφορα ότι στην εμφάνιση των αποτελεσμάτων εμφανίζονται μόνο οι επιλογές: προβολή, τροποποίηση και διαγραφή.

5.3 Προηγμένη αναζήτηση ασθενή-ιατρού

Παρακάτω θα περιγράψουμε την προηγμένη αναζήτηση ασθενή η οποία είναι ακριβώς ίδια με αυτή του ιατρού τα αποτελέσματα της αναζήτησης είναι όπως ακριβώς αναφέρονται στην παράγραφο 5.2 και φαίνονται στην εικόνα 5.6.

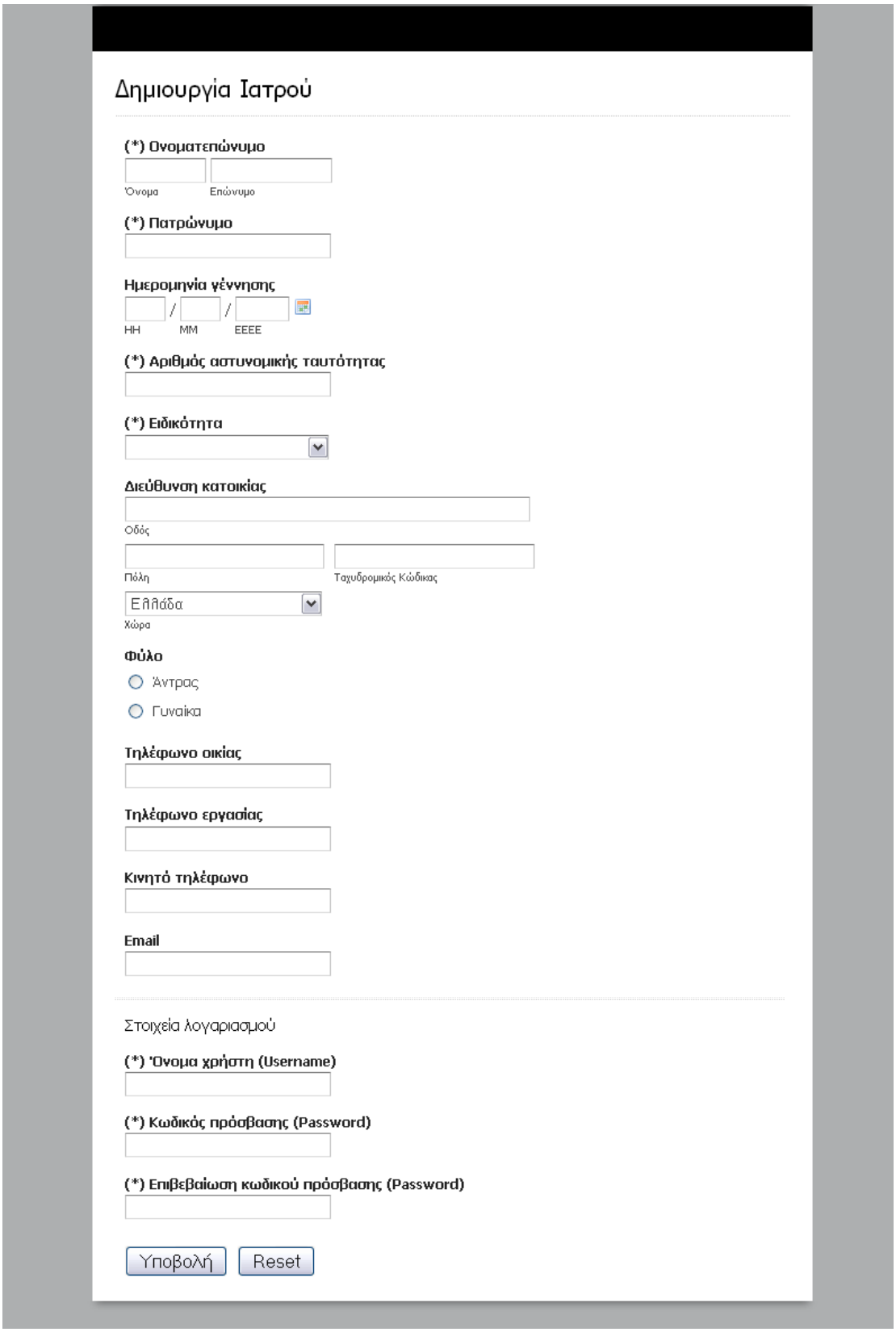
Ο χρήστης κάνοντας κλικ στην επιλογή 5.3 (προηγμένη αναζήτηση) του εμφανίζεται η ακόλουθη φόρμα (εικόνα 5.8), σε αυτήν καλείτε να συμπληρώσει οποία από τα πεδία θέλει η και κανένα από αυτά και να πατήσει το πλήκτρο αναζήτηση το οποίο θα του εμφανίσει τα αποτελέσματα.



5.8 Προηγμένη αναζήτηση.

5.4 Δημιουργία Ιατρού

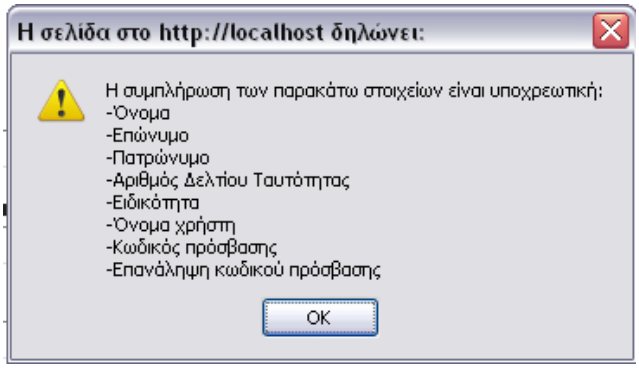
Ο τρόπος δημιουργίας οποιασδήποτε οντότητας στο σύστημα μας είναι ακριβώς ίδιος. Παρακάτω εξηγούμε την δημιουργία ιατρού.



5.9 φόρμα δημιουργίας γιατρού.

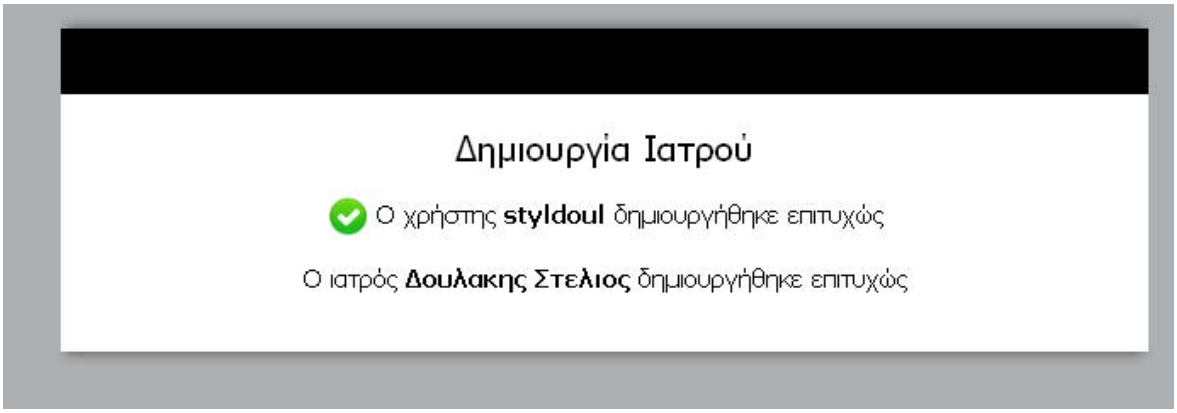
Στην παραπάνω εικόνα φαίνεται η φόρμα δημιουργίας ιατρού την οποία καλείτε να συμπληρώσει ο χρήστης, τα πεδία τα οποία φαίνονται με αστεράκι είναι

υποχρεωτικά και σε περίπτωση μη συμπλήρωσης τους εμφανίζεται μήνυμα όπως φαίνεται στην παρακάτω εικόνα 5.10.



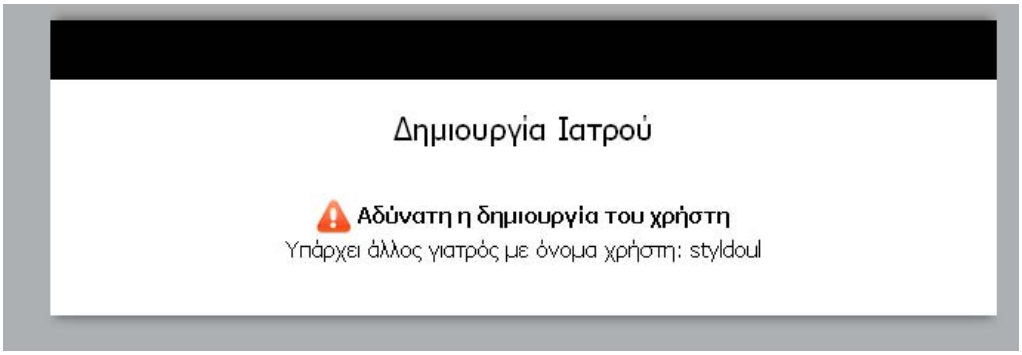
5.10 μήνυμα συμπλήρωσης απαραιτήτων στοιχείων.

Στην περίπτωση επιτυχημένης εγγραφής ιατρού στο σύστημα εμφανίζεται μήνυμα το οποίο φαίνεται παρακάτω και πληροφορεί τον χρήστη ότι έχει δημιουργηθεί επιτυχώς ο γιατρός καθώς και ο λογαριασμός του στην εικόνα 5.11.



5.11 μήνυμα επιτυχημένης εγγραφής ιατρού.

Στην περίπτωση μη επιτυχημένης εγγραφής ιατρού στο σύστημα εμφανίζεται μήνυμα το οποίο φαίνεται παρακάτω στην εικόνα 5.12.

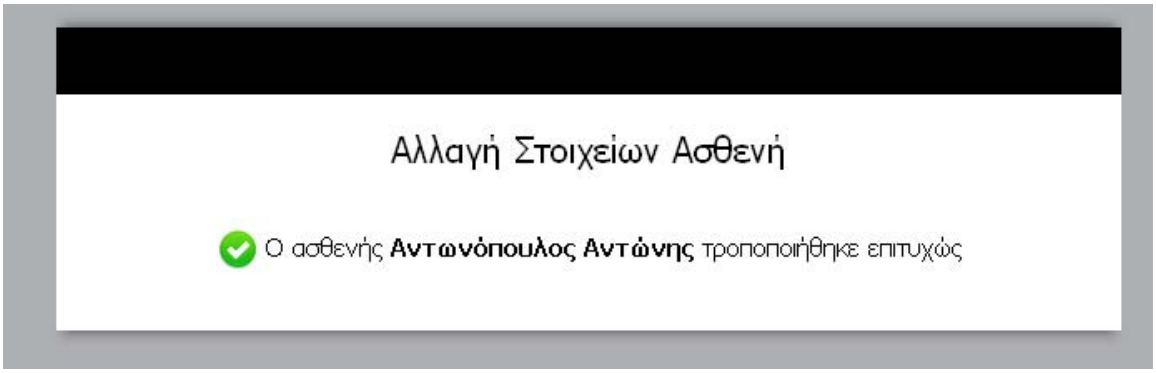


5.12 μήνυμα μη επιτυχημένης εγγραφής ιατρού.

5.5 Αλλαγή στοιχείων Ιατρού

Η τροποποίηση οποιασδήποτε οντότητας στο σύστημα μας γίνεται ακριβώς με τον ίδιο τρόπο όπως αναφέρουμε παρακάτω. Κατά την αλλαγή στοιχείων γιατρού ο χρήστης πρέπει να αναζητήσει τον γιατρό τον οποίο θέλει να τροποποιήσει, η διαδικασία αναζήτησης είναι ακριβώς ίδια με αυτήν της απλής αναζήτησης που αναφέραμε παραπάνω.

Στην περίπτωση αλλαγής γιατρού εμφανίζεται η φόρμα όπως φαίνεται στην εικόνα 5.9 συμπληρωμένη με τα ήδη υπάρχοντα στοιχειά του γιατρού, ο χρήστης μπορεί να αλλάξει τα στοιχειά και να πατήσει το πλήκτρο υποβολή για οποιαδήποτε επιλογή επιθυμεί, μετά την αλλαγή των στοιχείων εμφανίζεται μήνυμα που τον πληροφορεί για την επιτυχία της τροποποίησης όπως φαίνεται στην παρακάτω εικόνα.

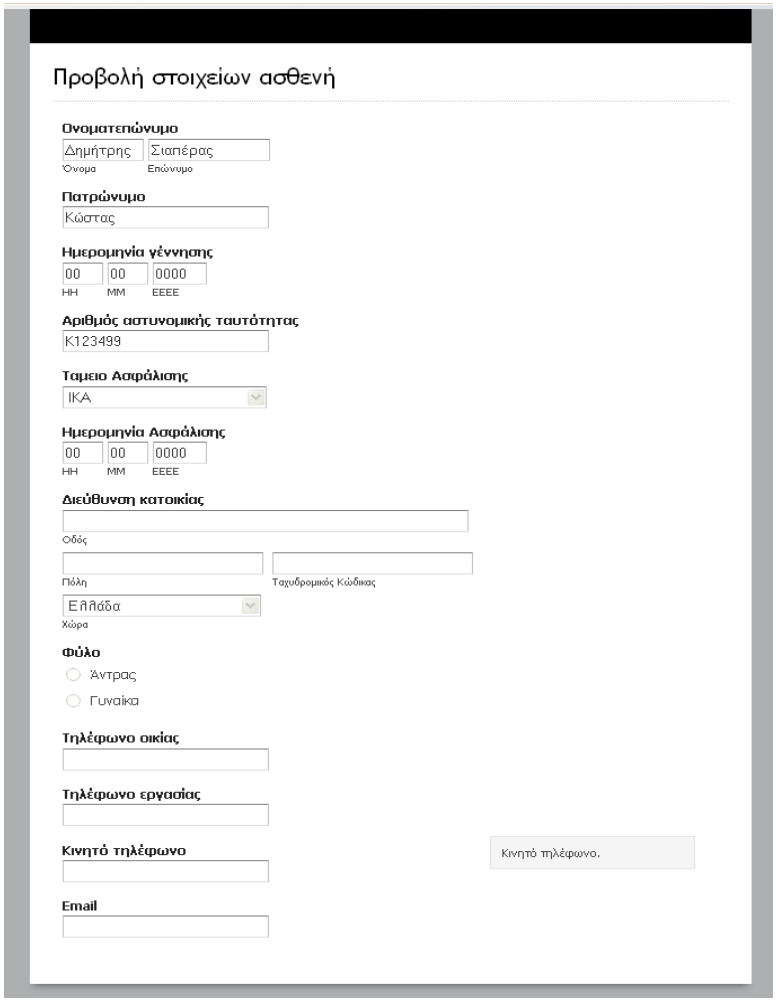


5.13 μήνυμα επιτυχημένης τροποποίησης ιατρού.

5.6 Προβολή στοιχείων Ιατρού

Η προβολή οποιασδήποτε οντότητας στο σύστημα μας γίνεται ακριβώς με τον ίδιο τρόπο όπως αναφέρουμε παρακάτω. Κατά την προβολή στοιχείων γιατρού ο χρήστης πρέπει να αναζητήσει τον γιατρό τον οποίο θέλει να δει, η διαδικασία αναζήτησης είναι ακριβώς ίδια με αυτήν της απλής αναζήτησης που αναφέραμε παραπάνω.

Στην περίπτωση αλλαγής γιατρού εμφανίζεται η φόρμα όπως φαίνεται στην εικόνα 5.9 συμπληρωμένη με τα ήδη υπάρχοντα στοιχειά του γιατρού μονό που ο χρήστης δεν μπορεί να πειράξει κανένα από τα πεδία καθώς παραλείπεται και το πλήκτρο υποβολή όπως φαίνεται παρακάτω.

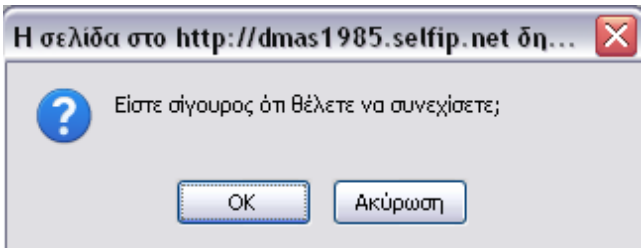


5.14 Φόρμα προβολής στοιχείων.

5.7 Διαγραφή Ιατρού

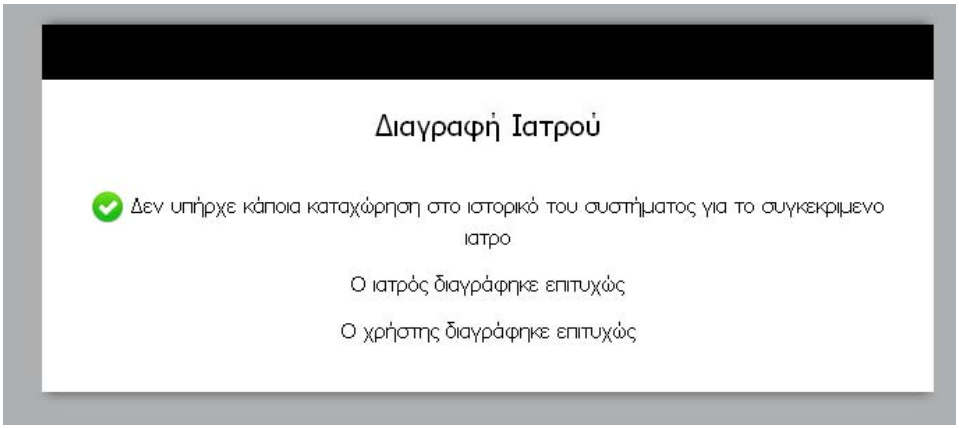
Η διαγραφή οποιασδήποτε οντότητας στο σύστημα μας γίνεται ακριβώς με τον ίδιο τρόπο όπως αναφέρουμε παρακάτω. Κατά την διαγραφή στοιχείων γιατρού ο χρήστης πρέπει να αναζητήσει τον γιατρό τον οποίο θέλει να διαγράψει, η διαδικασία αναζήτησης είναι ακριβώς ίδια με αυτήν της απλής αναζήτησης που αναφέραμε παραπάνω.

Πριν την διαγραφή του γιατρού εμφανίζεται μήνυμα επιβεβαίωσης της διαγραφής το οποίο ρωτάει τον χρήστη αν είναι σίγουρος και αν θέλει να συνεχίσει σε αυτήν την ενέργεια, αυτό το μήνυμα φαίνεται παρακάτω στην εικόνα 5.15.



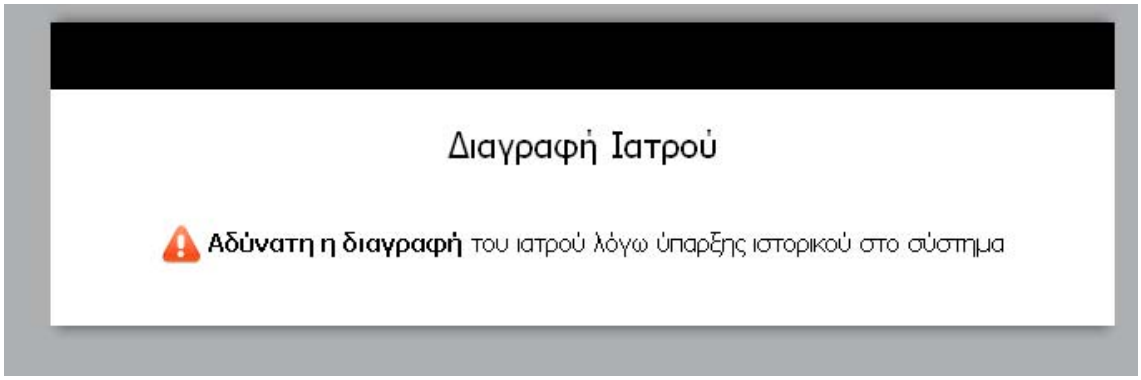
5.15 μήνυμα επιβεβαίωσης.

Σε περίπτωση που είναι δυνατή η διαγράφη του ιατρού εμφανίζεται μήνυμα το οποίο πληροφορεί τον χρήστη όπως φαίνεται στην παρακάτω εικόνα.



5.16 Μήνυμα επιβεβαίωσης διαγραφής.

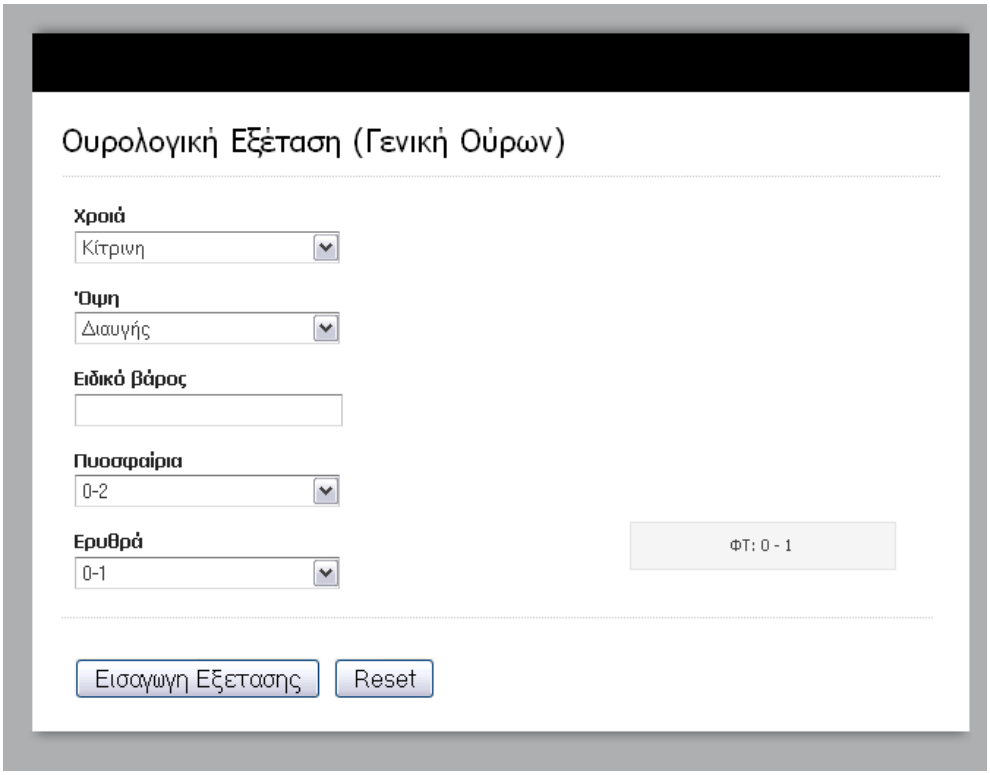
Σε περίπτωση που δεν είναι δυνατή η διαγράφη του ιατρού εμφανίζεται μήνυμα το οποίο πληροφορεί τον χρήστη όπως φαίνεται στην παρακάτω εικόνα.



5.17 Μήνυμα μη δυνατής διαγραφής.

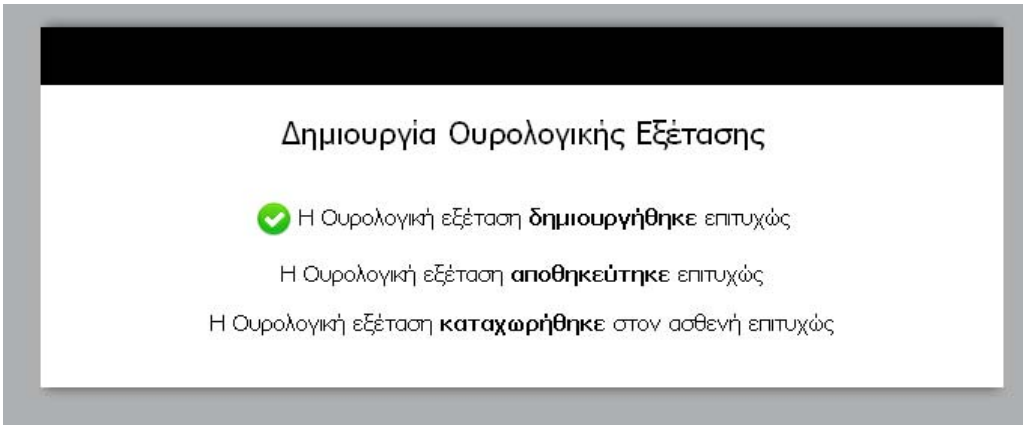
5.8 Καταχώρηση εξέτασης

Η καταχώρηση εξέτασης (όπως και της διάγνωσης) στο σύστημα μας γίνεται αφού ο χρήστης αναζητήσει τον ασθενή για τον οποίο θέλει να καταχωρήσει την εξέταση και αφού έχει επιλέξει τον τύπο της εξέτασης τον οποίο επιθυμεί, η διαδικασία αναζήτησης είναι ακριβώς ίδια με αυτήν της απλής αναζήτησης που αναφέραμε παραπάνω.



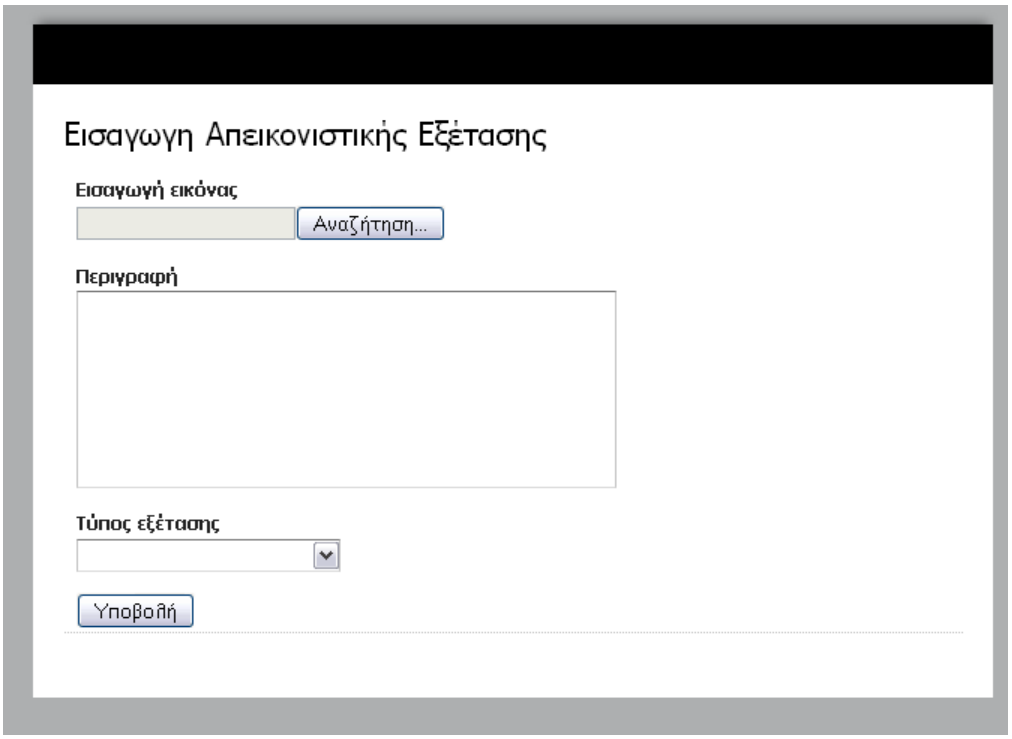
5.18 Φόρμα ουρολογικής εξέτασης.

Στην περίπτωση καταχώρησης οποιασδήποτε εξέτασης, εκτός από την απεικονιστική την οποία εξηγούμε παρακάτω, εμφανίζεται η φόρμα της κάθε εξέτασης όπως φαίνεται στην παραπάνω εικόνα 5.18. Ο χρήστης καλείτε να συμπληρώσει τα στοιχειά και να πατήσει το πλήκτρο υποβολή, μετά την συμπλήρωση των στοιχείων εμφανίζεται μήνυμα που τον πληροφορεί για την επιτυχία της καταχώρησης όπως φαίνεται στην παρακάτω εικόνα 5.19.

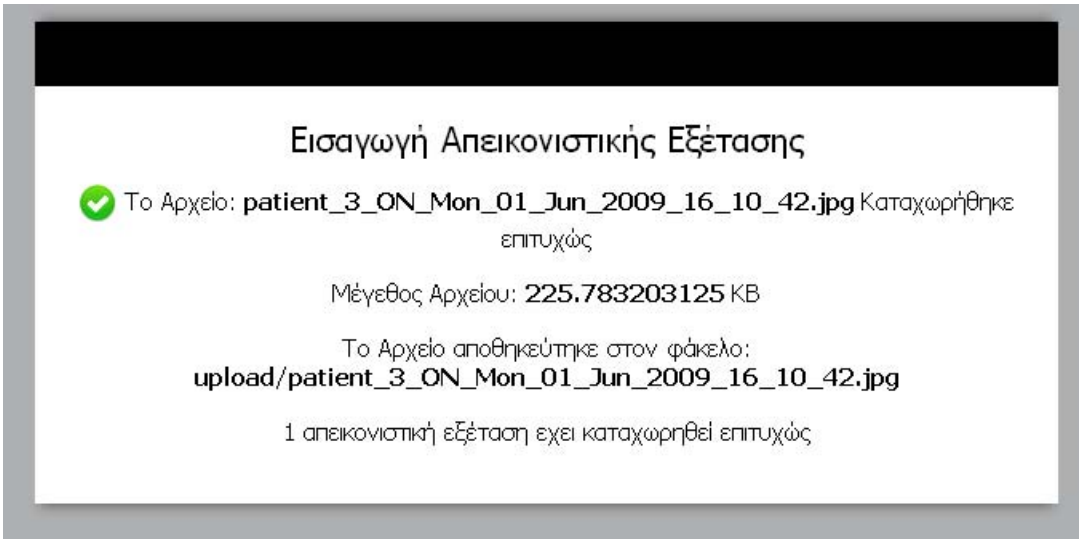


5.19 μήνυμα επιτυχημένης καταχώρησης εξέτασης.

Στην περίπτωση καταχώρησης απεικονιστικής εξέτασης η φόρμα διαφέρει λίγο, γιατί σε αυτήν την περίπτωση ο χρήστης εισάγει και μια εικόνα στο σύστημα, η φόρμα φαίνεται παρακάτω καθώς και τα μηνύματα τα οποία μπορεί να προκύψουν κατά την καταχώριση της.



5.20 Φόρμα απεικονιστικής εξέτασης.

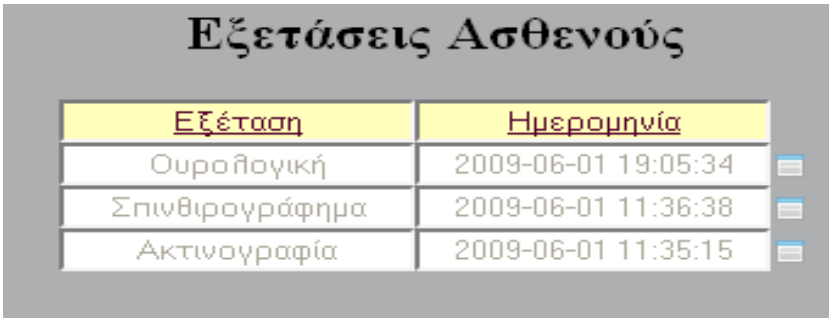


5.21 Μήνυμα επιτυχημένης καταχώρησης απεικονιστικής εξέτασης

5.9 Προβολή εξέτασης

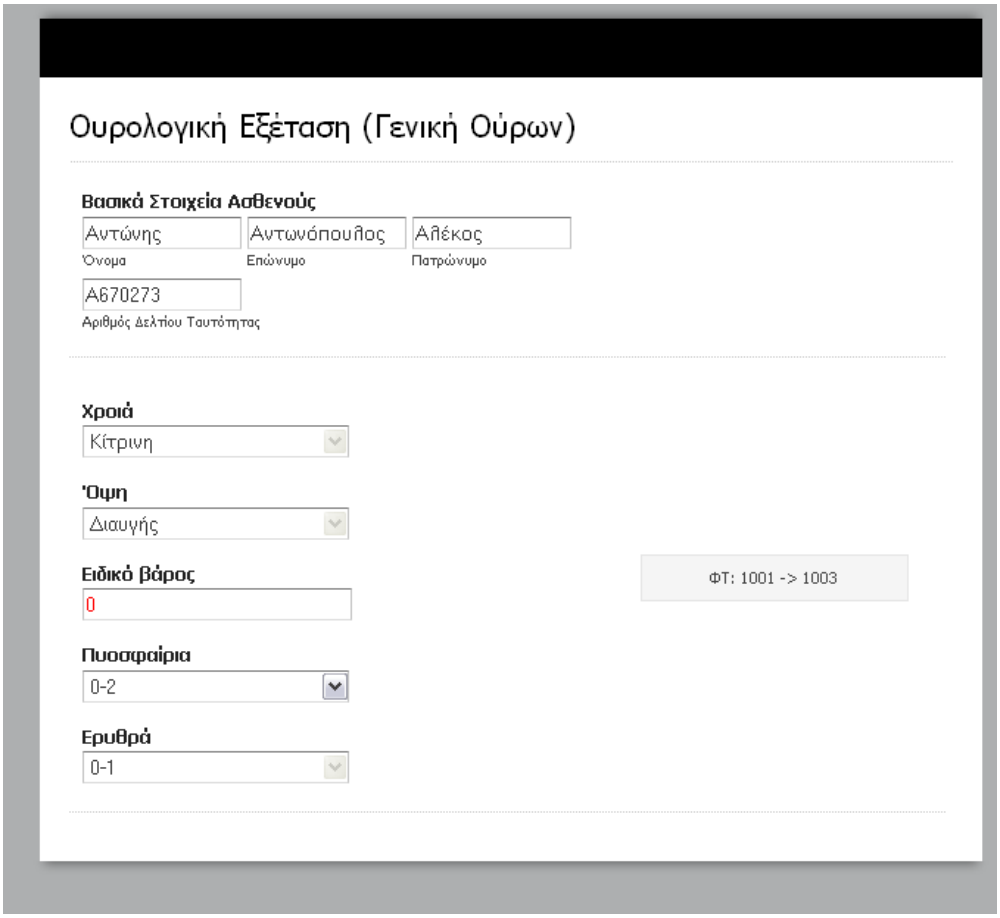
Η προβολή εξέτασης (όπως και της διάγνωσης) στο σύστημα μας γίνεται αφού ο χρήστης αναζητήσει τον ασθενή για τον οποίο θέλει να πληροφορηθεί για τις εξετάσεις του και αφού έχει επιλέξει τον τύπο της, η διαδικασία αναζήτησης

είναι ακριβώς ίδια με αυτήν της απλής αναζήτησης που αναφέραμε παραπάνω αφού επιλέξουμε τον ασθενή, εμφανίζεται μια λίστα με τις εξετάσεις τις οποίες έχει υποβληθεί ο ασθενής η οποία φαίνεται στην παρακάτω εικόνα 5.22.



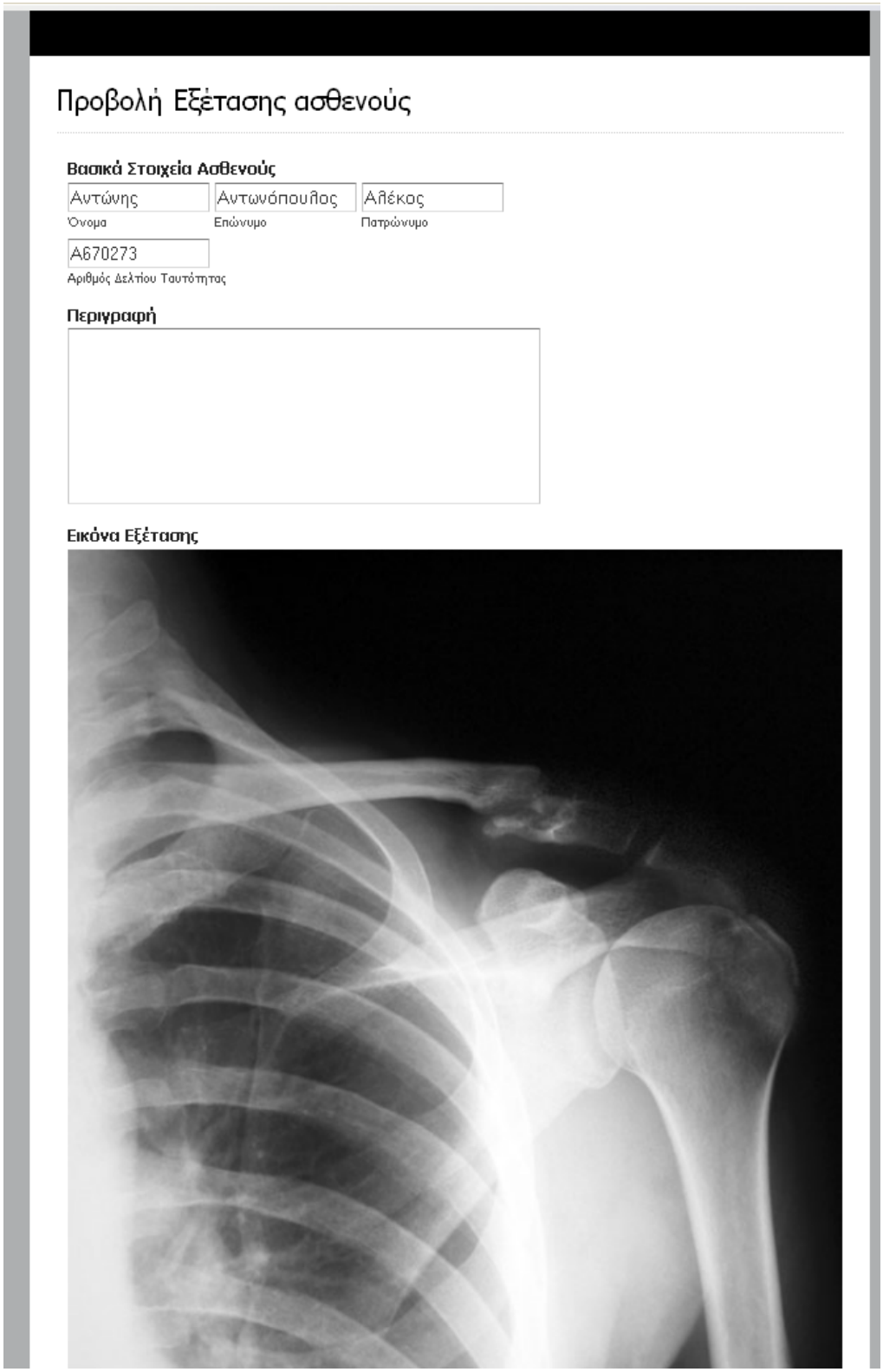
5.22 λίστα εξετάσεων ασθενούς.

Ο χρήστης στην συνεχεία μπορεί να επιλέξει μια από τις εξετάσεις για τις οποίες θέλει να πληροφορηθεί πατώντας στο κουμπί το οποίο είναι διπλά στην κάθε εξέταση το αποτέλεσμα της εμφάνισης φαίνεται παρακάτω στην εικόνα 5.23, μπορούμε μάλιστα να παρατηρήσουμε ότι κάποια πεδία είναι με κόκκινο χρώμα, αυτό γίνεται γιατί η τιμή στο συγκεκριμένο πεδίο δεν βρίσκεται μέσα στο όριο τον φυσιολογικών τιμών.



5.23 παράδειγμα προβολής εξέτασης.

Στην προβολή της απεικονιστικής εξέτασης έχουμε εκτός από εμφάνιση των πληροφοριών και εμφάνιση της φωτογραφίας την οποία έχει εισάγει στο σύστημα ο χρήστης, η προβολή μιας απεικονιστικής εξέτασης φαίνεται στην παρακάτω εικόνα 5.24.



5.24 προβολή απεικονιστικής εξέτασης(ακτινογραφία).

5.10 Διαγραφή εξέτασης

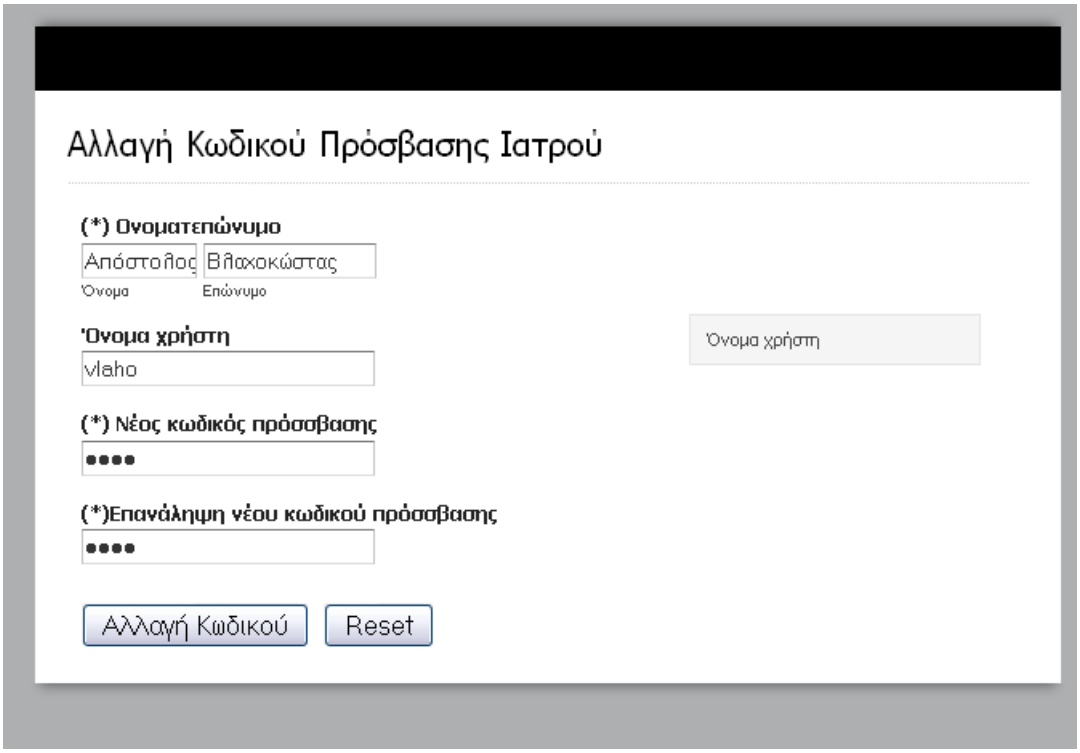
Η διαγραφή εξέτασης (όπως και της διάγνωσης) στο σύστημα μας γίνεται αφού ο χρήστης αναζητήσει τον ασθενή για τον οποίο θέλει να πληροφορηθεί για τις εξετάσεις του και αφού έχει επιλέξει τον τύπο της, η διαδικασία αναζήτησης είναι ακριβώς ίδια με αυτήν της απλής αναζήτησης που αναφέραμε παραπάνω, αφού επιλέξουμε τον ασθενή, εμφανίζεται μια λίστα με τις εξετάσεις τις οποίες έχει υποβληθεί ο ασθενής η οποία φαίνεται στην παραπάνω εικόνα 5.22 στην ενότητα 5.8.

Ο χρήστης στην συνεχεία μπορεί να επιλέξει μια από τις εξετάσεις την οποία θέλει να διαγράψει πατώντας στο κουμπί το οποίο είναι διπλά στην κάθε εξέταση. Στην συνεχεία το σύστημα αποκρίνεται με την εμφάνιση μηνύματος που πληροφορεί τον χρήστη για την διαγραφή όπως φαίνεται στην παρακάτω εικόνα.

5.25Μηνυμα διαγραφής εξέτασης.

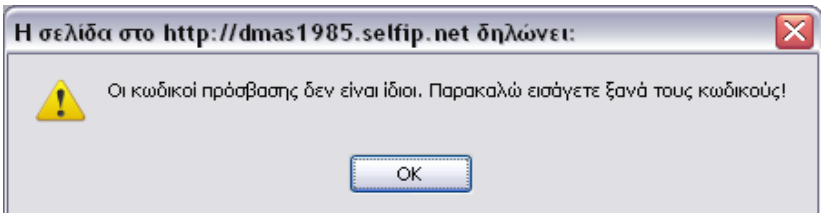
5.11 Αλλαγή κωδικού πρόσβασης Ιατρού.

Στην περίπτωση που κάποιος γιατρός έχει ξεχάσει τον κωδικό πρόσβασης του λογαριασμού του ο διαχειριστής μπορεί να αλλάξει τον κωδικό του σε κάποιον άλλο. Με την επιλογή από το μενού της αλλαγής κωδικού πρόσβασης ιατρού ο διαχειριστής καλείτε να συμπληρώσει μια φόρμα με τον καινούριο κωδικό πρόσβασης του γιατρού και να πατήσει το πλήκτρο υποβολή. Αυτό γίνεται αφού πρώτα έχει αναζητήσει τον γιατρό με τον τρόπο που έχουμε αναφέρει σε προηγούμενη ενότητα. Παρακάτω φαίνεται η φόρμα αλλαγής κωδικού πρόσβασης (εικόνα 5.26).



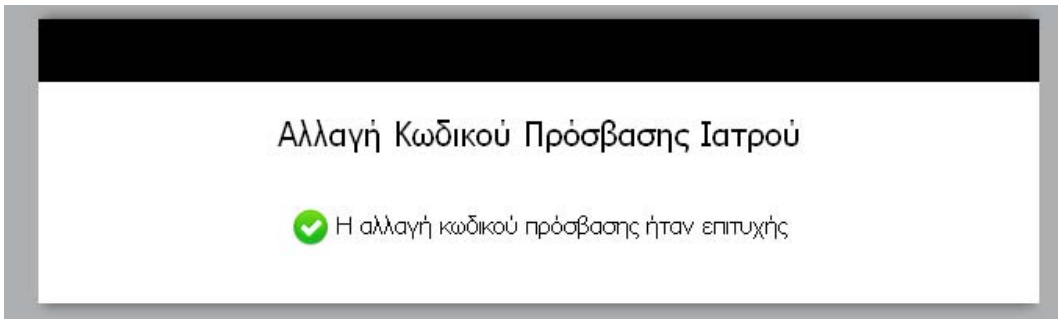
5.26 φόρμα αλλαγής κωδικού πρόσβασης.

Σε περίπτωση που ο κωδικός στα 2 πεδία διαφέρει εμφανίζεται μήνυμα το οποίο πληροφορεί τον χρήστη (εικόνα 5.27).



5.27 Μήνυμα μη ταυτοποίησης κωδικών πρόσβασης.

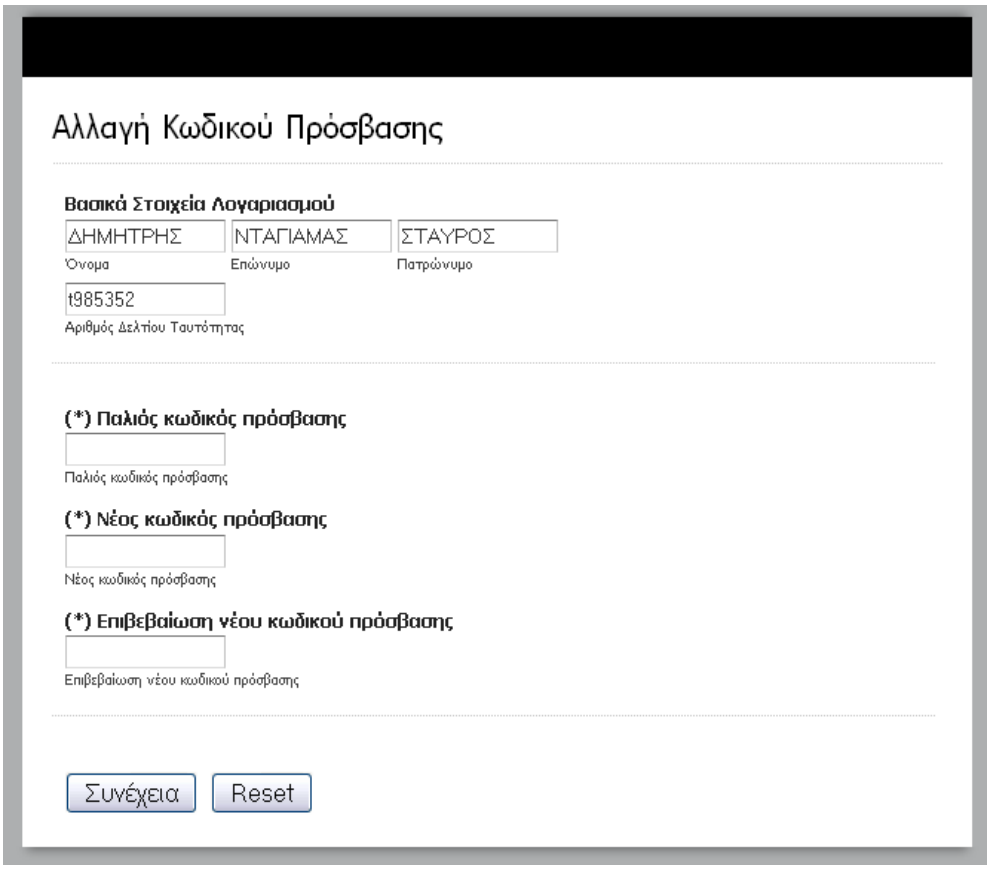
Στην περίπτωση που οι κωδικοί είναι σωστοί και γίνει η αλλαγή εμφανίζεται μήνυμα που πληροφορεί τον χρήστη όπως φαίνεται στην εικόνα 5.28.



5.28 μήνυμα επιβεβαίωσης αλλαγής κωδικού.

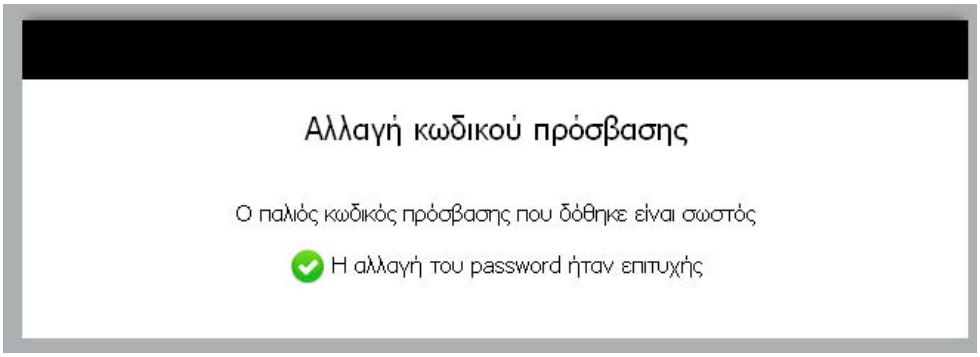
5.12 Αλλαγή κωδικού πρόσβασης

Με αυτή την επιλογή ο χρήστης μπορεί να αλλάξει τον κωδικό πρόσβασης του. Σε αυτήν την περίπτωση του εμφανίζεται η φόρμα αλλαγής κωδικού στην οποία ο χρήστης πρέπει να συμπληρώσει 3 πεδία, τον παλιό του κωδικό καθώς και άλλα 2 με τον καινούργιο όπως φαίνεται στην παρακάτω εικόνα 5.29.



5.29 Φόρμα αλλαγής κωδικού πρόσβασης.

Μετά την αλλαγή του κωδικού εμφανίζεται το παρακάτω μήνυμα όπως φαίνεται στην εικόνα 5.30.



5.30 μήνυμα επιβεβαίωσης αλλαγής κωδικού.

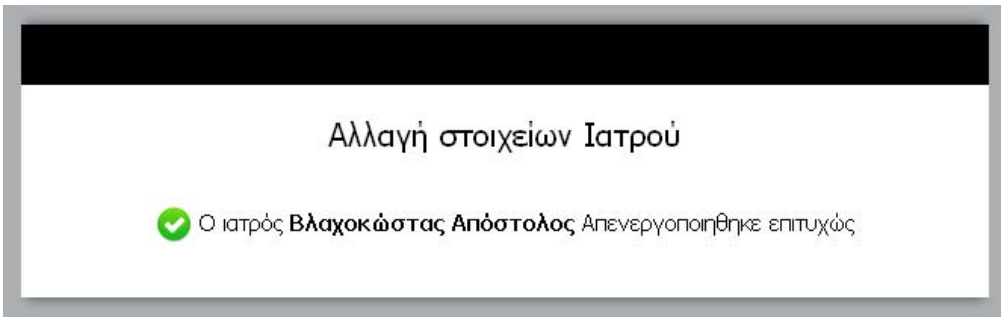
5.13 Κατάσταση λογαριασμού γιατρού.

Με αυτήν την επιλογή ο διαχειριστής είναι σε θέση να απενεργοποιήσει η να ενεργοποιήσει έναν γιατρό από το σύστημα, αυτό σημαίνει ότι ο γιατρός δεν θα μπορεί πλέον να συνδεθεί στο σύστημα με τον λογαριασμό του σε περίπτωση απενεργοποίησης η το αντίστροφο.

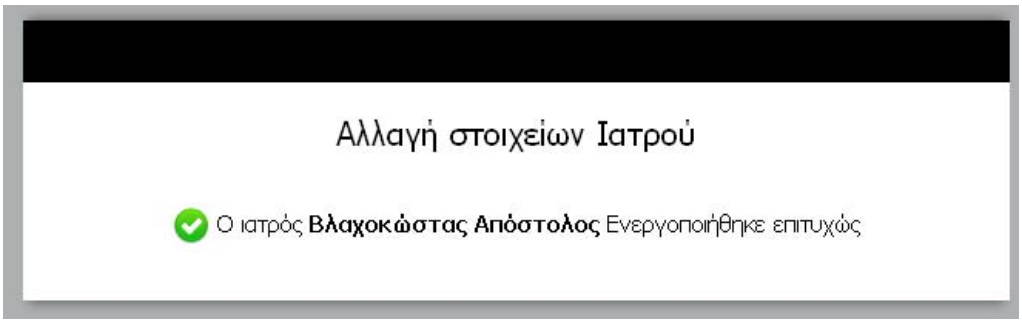
Ο διαχειριστής αφού αναζητήσει τον γιατρό τον οποίο θέλει να τροποποιήσει, επιλέγει την αντίστοιχη επιλογή για την διαχείριση του λογαριασμού του από τα κουμπιά, τα οποία εμφανίζονται δίπλα στο όνομα των γιατρών, όπως φαίνεται στην παρακάτω εικόνα 5.31.

5.31 αποτελέσματα αναζήτησης.

Στην συνέχεια, αφού επιλέξει την κατάλληλη επιλογή εμφανίζονται αντίστοιχα τα παρακάτω μηνύματα, αναλόγως με την επιλογή του.



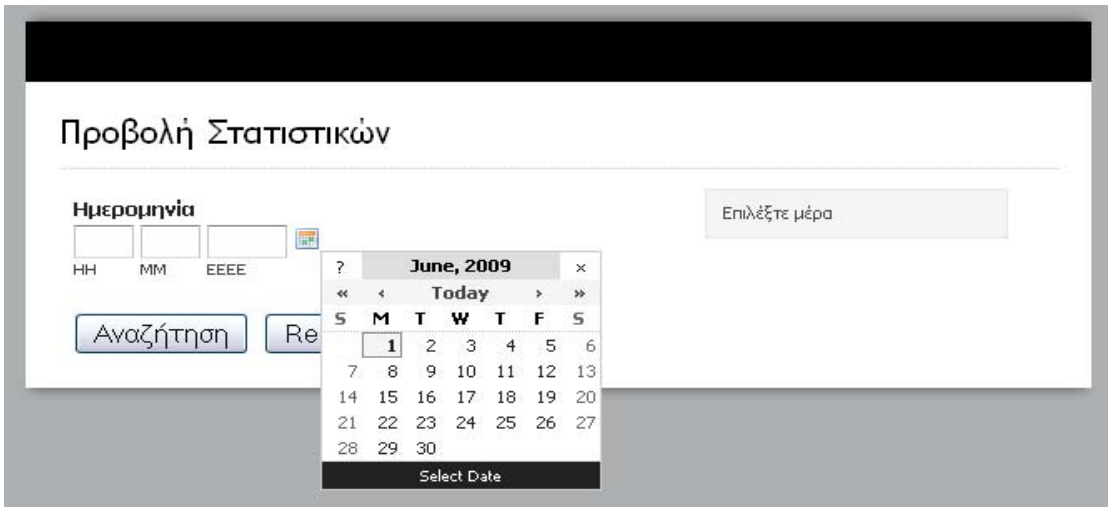
5.32 Επιτυχής ενεργοποίηση γιατρού.



5.33 Επιτυχής απενεργοποίηση γιατρού.

5.14 Προβολή στατιστικών συστήματος.

Με αυτή την επιλογή ο χρήστης μπορεί να πληροφορηθεί για τα στατιστικά στοιχειά για μια συγκεκριμένη ημέρα. Με αυτόν τον τρόπο μπορούν να καταμετρηθούν όλες οι εξετάσεις και οι διαγνώσεις που καταχωρήθηκαν μια συγκεκριμένη ημέρα στο σύστημα. Ο χρήστης το μόνο που έχει να κάνει είναι να επιλέξει την ημέρα από ένα calendar το οποίο είναι διαθέσιμο η ακόμη να την συμπληρώσει χειροκίνητα και να πατήσει το πλήκτρο υποβολή, η φόρμα είναι η παρακάτω (εικόνα 5.34).



5.34 Φόρμα στατιστικών.

Τα αποτελέσματα αυτής της επιλογής φαίνονται παρακάτω στην εικόνα που ακλουθεί (εικόνα 5.35).

5.35 εμφάνιση στατιστικών.

Με πορτοκαλί χρώμα φαίνονται τα πιο σημαντικά στοιχειά του συστήματος. Σε κάθε κελί αναφέρεται από διπλά ένας αριθμός ο όποιος δείχνει πόσα στοιχεία

έχουν προστεθεί στο σύστημα για την συγκεκριμένη ημέρα. Στην εικόνα 5.35 για παράδειγμα βλέπουμε ότι έχει προστεθεί μια εργαστηριακή εξέταση στην 31-05-2009.

Βιβλιογραφία

- PHP Οδηγός Προγραμματισμού
  Συγγραφέας: Hughes, Sterling
- Ανάπτυξη Web Εφαρμογών με PHP και MySQL
  Συγγραφέας: Thomson, Laura | Welling, Luke
- Πλήρης Οδηγός της PHP 5
  Συγγραφέας: Atkinson, Leon | Suraski, Zeev
- Συστηματα διαχειρισης βασεων δεδομενων τομος 1,
  Συγγραφέας:Raghu Ramakrishnan Jonannew Gerhke
- Συστηματα διαχειρισης βασεων δεδομενων τομος 2,
  Συγγραφέας:Raghu Ramakrishnan Jonannew Gerhke
- Μαθηματα βασεων δεδομενων
  Συγγραφέας:Δημητρης Δερβος.
- Professional LAMP:
  Linux, Apache, MySQL and PHP5 Web Development
- PHP and MySQL: Create - Modify - Reuse
  Συγγραφέας:Timothy Boronczyk with Martin E. Psinas
- Php and Mysql Manual: Simple, Yet Powerful Web Programming
  Συγγραφέας: Simon Stobart

Διευθύνσεις Διαδικτύου-Πηγές

- http://www.hotsscripts.com
- http://www.zend.com
- http://www.phpbuilder.com
- http://phplens.com
- http://www.php.net
- http://httpd.apache.org
- http://www.mysql.org
- http://www.tizag.com
- http://php.about.com
- http://www.phpform.org
- http://www.roscripts.com
- http://www.onlamp.com

- http://www.php-mysql-tutorial.com
- http://www.dynamicdrive.com
- http://www.enghiong.com
- http://www.freestuff.gr
- http://www.webdeveloper.com
- http://www.wampserver.com
- http://www.phpclasses.org/
- http://sourceforge.net
- http://en.wikipedia.org