

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής



Εργασία Μαθήματος «Προγραμματισμός στο διαδίκτυο και
στον παγκόσμιο ιστό»

Ημερομηνία Παράδοσης: 06/09/2017

Ακαδημαϊκό Έτος 2016-2017



Εκφώνηση της άσκησης

Δημιουργία 3-tier εφαρμογής για την διαχείριση ραντεβού ιατρικών εξετάσεων.

Αναλυτικά Βήματα:

1. Επέκταση web project προηγούμενης άσκησης

1.1. Στην τελική εργασία θα επεκτείνετε τη λειτουργικότητα του web project που δημιουργήσατε στην προηγούμενη άσκηση και θα υλοποιήσετε όλη την ζητούμενη λειτουργικότητα για κάθε κατηγορία χρηστών.

2. Δημιουργία διαδικτυακής διεπαφής

2.1. Για την είσοδο των χρηστών στο σύστημα θα υλοποιείτε μηχανισμό login με username και password. Το password θα αποθηκεύεται σε κρυπτογραφημένη (hashed+salted) μορφή. Από την αρχική σελίδα οι διάφοροι χρήστες θα μπορούν να έχουν πρόσβαση στις λειτουργίες τους.

2.2. Σε αυτό το βήμα, θα υλοποιήσετε τις διαδικτυακές διεπαφές (html σελίδες) που θα χρησιμοποιούν οι χρήστες όλων των κατηγοριών (Ασθενείς, Ιατροί, Διαχειριστές) για να αλληλεπιδρούν με την εφαρμογή και να χρησιμοποιούν τις αντίστοιχες μεθόδους που απαιτούνται.

2.2.1. Θα υπάρχει ένα κεντρικό μενού σε μία index.html σελίδα, η οποία θα είναι η αρχική σελίδα για όλους τους χρήστες. Μετά το login θα προβάλλεται το μενού λειτουργιών κάθε χρήστη ανάλογα με την κατηγορία στην οποία ανήκει.

2.2.2. Λειτουργίες Ασθενών (Patient): Οι Ασθενείς θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: προβολή ιστορικού προηγούμενων ραντεβού, προβολή διαθέσιμων κενών για κλείσιμο ραντεβού με έναν γιατρό κάποιας ειδικότητας, κλείσιμο ραντεβού ακύρωση ραντεβού (σε περίπτωση που είναι τουλάχιστον 3 ημέρες μετά).

2.2.3. Λειτουργίες Ιατρών (Doctor): Οι Ιατροί θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: καταχώρηση διαθεσιμότητας για ραντεβού (ανά μήνα), προβολή πίνακα ραντεβού, ακύρωση ραντεβού (σε περίπτωση που είναι τουλάχιστον 3 ημέρες μετά).

2.2.4. Λειτουργίες Διαχειριστή (Administrator). Οι Διαχειριστές θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: εισαγωγή νέου Ιατρού και χρήστη, διαγραφή Ιατρού.

2.3. Η εφαρμογή θα υποστηρίζει διαχείριση συνόδου (session management) από τη στιγμή που ο χρήστης συνδέεται, μέχρι την αποσύνδεσή του από την εφαρμογή. Κατά την αποσύνδεση του χρήστη θα πρέπει να διαγράφεται το session.

3. Υλοποίηση επιπέδου Δεδομένων και σύνδεση εφαρμογής με τη βάση

3.1. Όλα τα δεδομένα θα αποθηκεύονται σε βάση δεδομένων, την οποία έχετε σχεδιάσει από την 2η Άσκηση (στο παράδειγμα χρησιμοποιήσαμε mysql και mysql workbench). Μπορείτε να προβείτε σε όποιες τροποποιήσεις θεωρείτε απαραίτητες. Προσθέστε δοκιμαστικά δεδομένα στη βάση.

3.2. Διαμορφώστε κατάλληλα το project σας ώστε να συνδέσετε τη Βάση Δεδομένων που έχετε δημιουργήσει με τον application server σας, ως μία 3-tier εφαρμογή (μπορείτε να βρείτε αντίστοιχο παράδειγμα στα παραδείγματα κώδικα που περιλαμβάνονται στη σελίδα του μαθήματος).



4. Υλοποίηση επιπέδου επεξεργασίας (servlet)

4.1. Διαμορφώστε κατάλληλα το project σας ώστε να επικοινωνεί με τον application server της επιλογής σας (στα java παραδείγματα έχουμε χρησιμοποιήσει apache tomcat).

4.2. Υλοποιήσετε όλες τις λειτουργίες που προσφέρει η εφαρμογή σας χρησιμοποιώντας τεχνολογία servlet. Δημιουργήστε ένα ή περισσότερα servlet τα οποία θα δέχονται είσοδο από το επίπεδο διεπαφής (html σελίδες και φόρμες), θα αναζητούν στη βάση δεδομένων τα στοιχεία που απαιτούνται ότι απαιτείται και θα επιστρέφουν το αποτέλεσμα στον εκάστοτε χρήστη ως δυναμική html σελίδα.

4.3. Προαιρετικά, μπορείτε να χρησιμοποιήσετε τεχνολογία jsp για τη δημιουργία και την διαμόρφωση των ιστοσελίδων.



ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1. Γενική περιγραφή της εφαρμογής

1.1. Δημιουργία web project και υλοποίηση λειτουργιών

1.2. Δημιουργία διαδικτυακής επαφής

1.2.1. Για την είσοδο των χρηστών στο σύστημα θα υλοποιήσετε μηχανισμό login με username και password

1.2.2. Σε αυτό το βήμα, θα υλοποιήσετε τις διαδικτυακές διεπαφές που θα χρησιμοποιούν οι χρήστες όλων των κατηγοριών για να αλληλεπιδρούν με την εφαρμογή και να χρησιμοποιούν τις αντίστοιχες μεθόδους που απαιτούνται.

1.2.2.1. Θα υπάρχει ένα κεντρικό μενού σε μία index.html σελίδα, η οποία θα είναι η αρχική σελίδα για όλους τους χρήστες. Μετά το login θα προβάλλεται το μενού λειτουργιών κάθε χρήστη ανάλογα με την κατηγορία στην οποία ανήκει.

1.2.2.2. Λειτουργίες Διαχειριστή (Administrator). Οι Διαχειριστές θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: εισαγωγή νέου Ιατρού και διαγραφή Ιατρού.

1.2.2.3. Λειτουργίες Ιατρών (Doctor): Οι Ιατροί θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: καταχώρηση διαθεσιμότητας για ραντεβού (ανά μήνα), προβολή πίνακα με τις διαθέσιμες ημερομηνίες και ώρες (ανά μισάωρα) για ραντεβού που έχει εισάγει και δυνατότητα ακύρωσης αυτών.

1.2.2.4. Λειτουργίες Ασθενών (Patient): Οι Ασθενείς θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: προβολή ιστορικού προηγούμενων ραντεβού, προβολή διαθέσιμων των διαθέσιμων Ιατρών στο σύστημα, κλείσιμο ραντεβού, ακύρωση ραντεβού

1.2.3. Η εφαρμογή θα υποστηρίζει διαχείριση συνόδου (session management) από τη στιγμή που ο χρήστης συνδέεται, μέχρι την αποσύνδεσή του από την εφαρμογή. Κατά την αποσύνδεση του χρήστη θα πρέπει να διαγράφεται το session.

1.3. Αλληλεπίδραση με τη Βάση Δεδομένων

1.3.1. Registrations αρχεία

1.3.2. Login αρχεία

1.3.3. Delete αρχεία



1. Γενική περιγραφή της εφαρμογής

Η διαδικτυακή εφαρμογή e-γεία αποτελεί μια 3-tier εφαρμογή, η οποία δίνει την δυνατότητα οργάνωσης και διαχείρισης ιατρικών ραντεβού. Υπάρχουν τρεις κατηγορίες χρηστών: Ασθενείς, Ιατροί και Διαχειριστές, ο καθένας εκ των οποίων έχει διαφορετικά δικαιώματα και επιλογές.

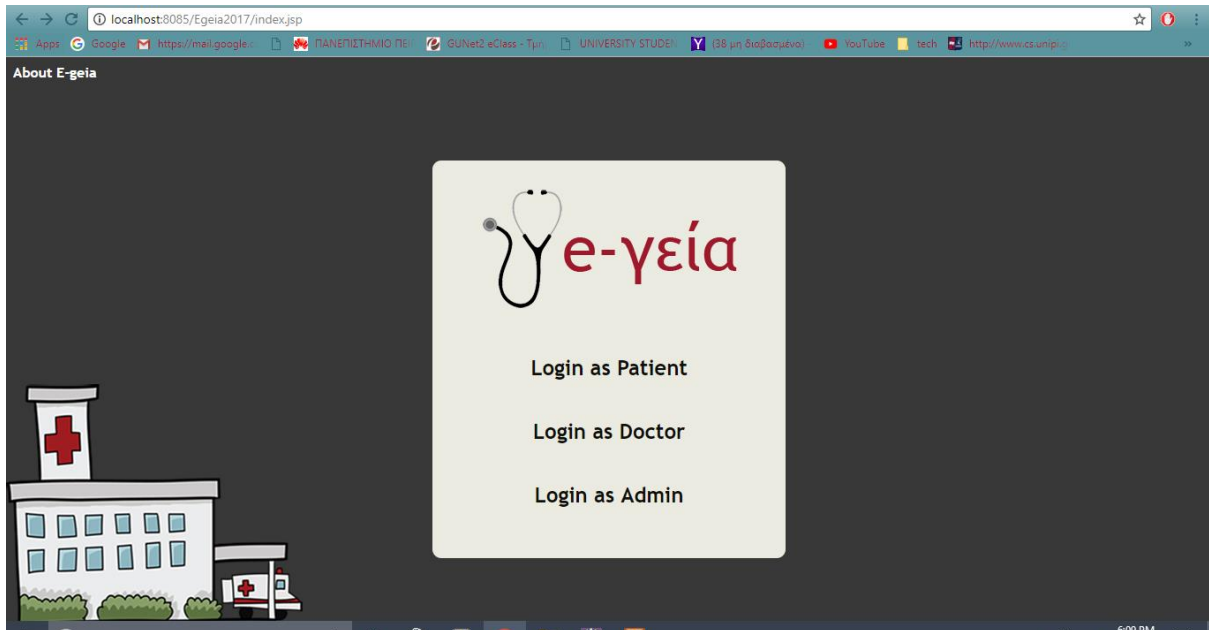
Πιο συγκεκριμένα, η εργασία υλοποιήθηκε σε γλώσσα προγραμματισμού Java, Servlets (Dynamic Web Project), μαζί με HTML, CSS, JS. Για τη βάση δεδομένων χρησιμοποιήσαμε τη γλώσσα MySQL, καθώς και το ανάλογο εργαλείο του phpMyAdmin. Για το ανέβασμα της εργασίας στο Local Host, χρησιμοποιήσαμε τον Tomcat 8.5 Server. Τέλος το project δημιουργήθηκε με τη βοήθεια του Eclipse IDE. Βασίζεται στην αρχιτεκτονική MVC (Model-View-Controller) , η οποία αποτελείται από:

- **Model:** Στο model τοποθετούμε τις λειτουργίες της εφαρμογής που σχετίζονται με την πρόσβαση στη βάση δεδομένων.
- **View:** Το view αντιστοιχεί στο html κομμάτι της σελίδας της εφαρμογής.
- **Controller:** Ελέγχει τις αλληλεπιδράσεις μεταξύ view και model.

[Models] <-----> [Repositories] <-----> [Services] <----->
[Views]

Εικόνα 1. #EgeiaApp hierarchy

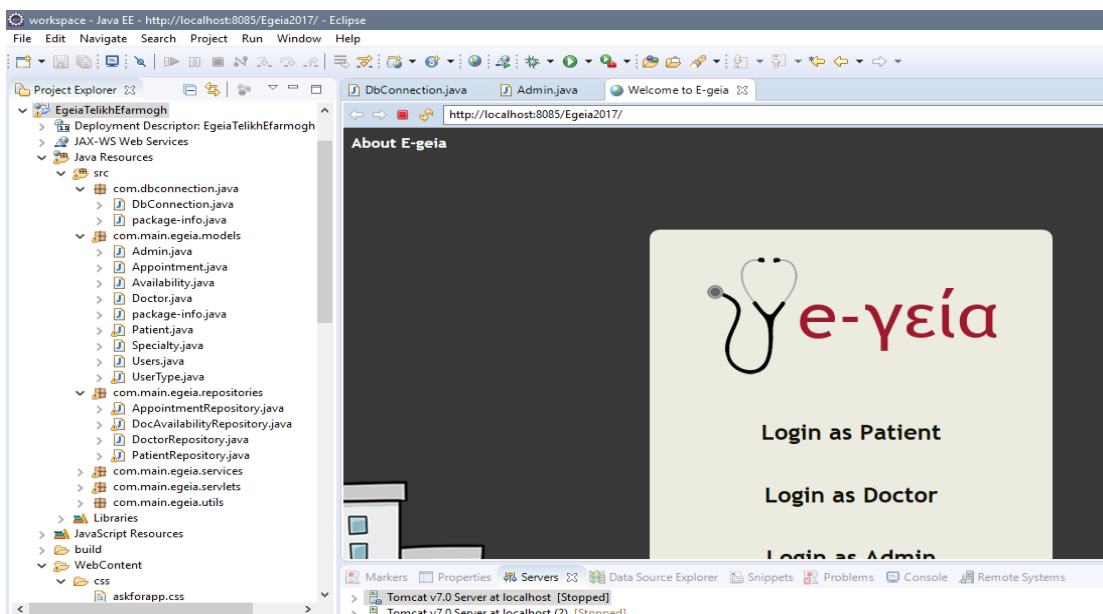
1.1. Δημιουργία web project και υλοποίηση λειτουργιών



Εικόνα 2. Αρχική σελίδα της εφαρμογής e-γεία

Όπως φαίνεται παρακάτω, έχουμε δημιουργήσει τα εξής πακέτα κλάσεων:

- com.dbconnection.java
- com.main.egeia.models
- com.main.egeia.repositories
- com.main.egeia.services
- com.main.egeia.servlets
- com.main.egeia.utils



Εικόνα 3. Πακέτα κλάσεων που δημιουργήθηκαν



Πιο αναλυτικά:

- Το πακέτο κλάσεων **com.dbconnection.java** εμπεριέχει την κλάση DBConnection.java στην οποία επιτυγχάνεται η σύνδεση με τη βάση που φτιάξαμε για την εφαρμογή. Η κλάση αυτή εμπεριέχει τον pull container του application server, ώστε να επιτυγχάνεται αυτόματη διαχείριση των queries.
 - Το πακέτο κλάσεων **com.main.egeia.models** εμπεριέχει τις κλάσεις Admin.java, Appointment.java, Doctor.java, Patient.java, Appointment.java και Availability.java. Αναλύοντας τις απαιτήσεις της εργασίας, καταλήξαμε στην εφαρμογή των συγκεκριμένων μοντέλων προκειμένου για να διατηρήσουμε τα δεδομένα και να απεικονίσουμε τις κατάλληλες οντότητες. Τα παραπάνω μοντέλα αντιστοιχούν κατά ένα μεγάλο ποσοστό στα tables της βάσης.
 - Το πακέτο κλάσεων **com.main.egeia.repositories** εμπεριέχει τις κλάσεις PatientRepository.java, DoctorRepository.java, DocAvailabilityRepository.java, AppointmentRepository.java. Το κομμάτι των repositories αποτελεί το control layer της εφαρμογής. Εδώ εκτελούνται αποκλειστικά τα sql queries που είναι απαραίτητα για να υλοποιηθεί το κομμάτι της επιχειρησιακής λογικής.
 - Το πακέτο κλάσεων **com.main.egeia.services** εμπεριέχει τις κλάσεις PatientService.java, DoctorService.java, DocAvailabilityService.java, AppointmentService.java. Τα services δομούνται με βάση τα repositories, ενώ εδώ πραγματοποιούνται οι διάφορες λειτουργίες μεταξύ των μοντέλων.
 - Το πακέτο κλάσεων **com.main.egeia.servlets** εμπεριέχει όλες τις java κλάσεις που διαχειρίζονται και εμφανίζουν τα αποτελέσματα όλων των αλληλεπιδράσεων που περιγράφηκαν προηγουμένως. Επίσης οι συγκεκριμένες κλάσεις δίνουν τη δυνατότητα στο χρήστη να τροποποιήσει τα δεδομένα.
- Τέλος
- Το πακέτο κλάσεων **com.main.egeia.utils** εμπεριέχει την κλάση PasswordEncoder.java ώστε να αποθηκεύονται hashed τα passwords στην βάση καθώς και την κλάση SessionUtils η οποία βοηθάει στο να κρατήσουμε το id του χρήστη όταν μπει στο session και να το χρησιμοποιήσουμε σε διάφορες λειτουργίες (όπως αναζήτηση για το συγκεκριμένο id σε κάποιο table) .

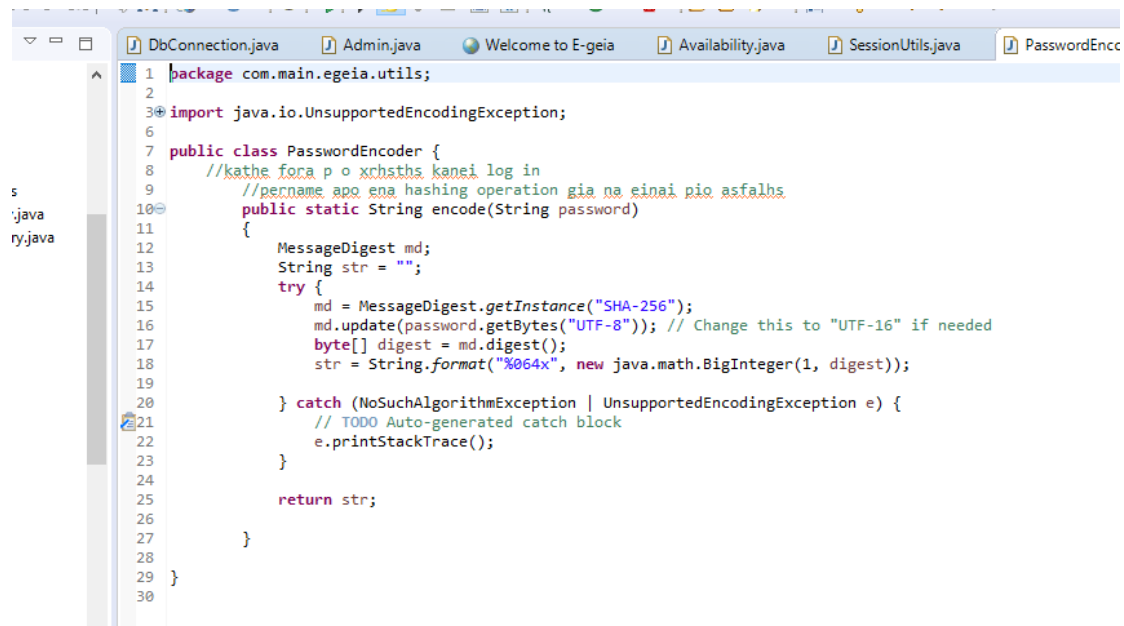
Τέλος, στο φάκελο WebContent, υπάρχουν οι προαπαιτούμενες HTML, JSP σελίδες, καθώς και στο φάκελο WEB-INF, το αρχείο web.xml στο οποίο δηλώνονται, η αρχική σελίδα (index.jsp).



1.2. Δημιουργία διαδικτυακής επαφής

1.2.1 Για την είσοδο των χρηστών στο σύστημα θα υλοποιήσετε μηχανισμό login με username και password. Το password θα αποθηκεύεται σε κρυπτογραφημένη (hashed+salted) μορφή. Από την αρχική σελίδα οι διάφοροι χρήστες θα μπορούν να έχουν πρόσβαση στις λειτουργίες τους.

Προτού κάποιος χρήστης εισαχθεί στο σύστημα, κάνουμε hashing τον κωδικό του, δηλαδή τον αποθηκεύουμε σε κρυπτογραφημένη μορφή, έτσι ώστε να εισαχθεί με ασφάλεια στη Βάση και να αποφύγουμε τον κίνδυνο SQL injection. Στις παρακάτω εικόνες βλέπουμε τον κώδικα του PasswordEncoder.java και πως τον χρησιμοποιήσαμε για την εγγραφή των Ιατρών στο σύστημα.



Εικόνα 4. Κώδικας του αρχείου PasswordEncoder.java

```
public int InsertDoctor(Doctor doctor)
{
    int res = -1;
    if( doctor.getFirstName().isEmpty() ||
        doctor.getLastName().isEmpty() ||
        doctor.getEmail().isEmpty() ||
        doctor.getSpeciality().isEmpty() ||
        doctor.getGender().isEmpty() ||
        doctor.getPassword().isEmpty() ||
        !(doctor.getPassword().equals(doctor.getCPassword()))
    )
        return res;

    if( doctor.getPassword().length() < 64)
        //Need to hash password - Dummy check
        doctor.setPassword(PasswordEncoder.encode(doctor.getPassword()));
    return repo.InsertDoctor(doctor);
}
```

Εικόνα 5. Χρήση της κλάσης PasswordEncoder.java στη μέθοδο εισαγωγής των Ιατρών InsertDoctor της κλάσης DoctorService.java



























SELECT * FROM 'doctors'

[Επεξεργασία εσωτερικά] [Επεξεργασία] [Ανάλυση SQL] [Δημιουργία κώδικα PHP] [Αναζήτηση]

☐ Εμφάνιση όλων | Αριθμός εγγραφών: 25 | Φιλτράρισμα εγγραφών: Αναζήτηση σε αυτόν τον πίνα

αξινόμηση ανά κλειδί: Καμία

Επιλογές

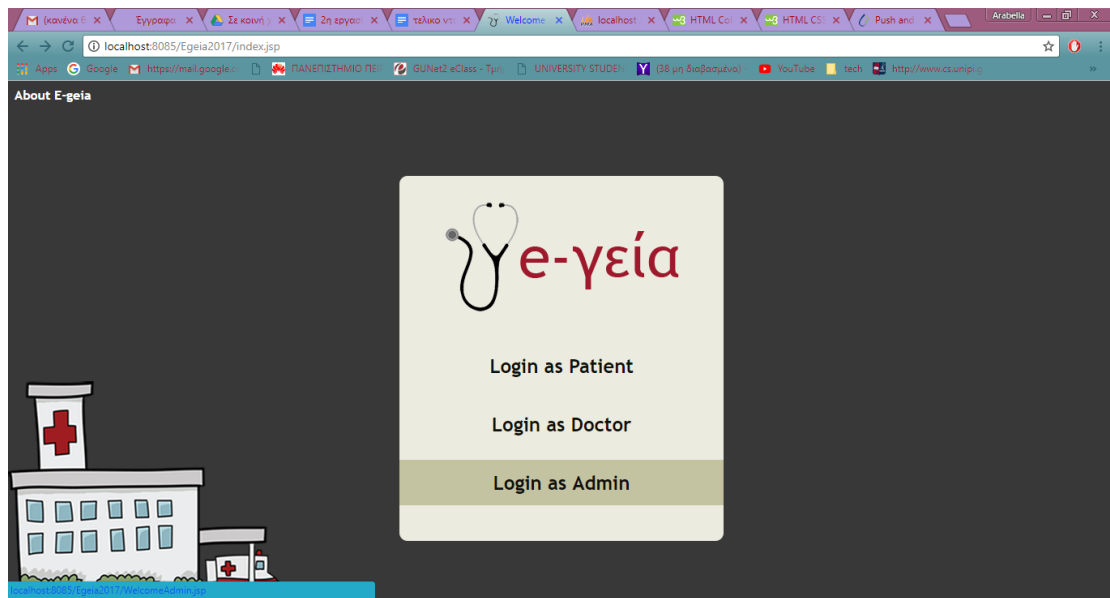
		id	firstname	lastname	username	password	email				
  	Επεξεργασία	  	Αντιγραφή	  	Διαγραφή	10	Elena	Papadakh	elpap	4fc82b26aecb47d2868c4efbe3581732a3e7cbcc6c2efb3206...	elpap@gmail.com
  	Επεξεργασία	  	Αντιγραφή	  	Διαγραφή	11	Giorgos	Makrhs	gmak	79f06f8fde333461739f220090a23cb2a79fd714bee100d0e...	gmak@gmail.com
  	Επεξεργασία	  	Αντιγραφή	  	Διαγραφή	12	Alexandros	Papadakis	alpap	79f06f8fde333461739f220090a23cb2a79fd714bee100d0e...	alpap@gmail.com

Εικόνα 6. Το αποτέλεσμα του hashed κωδικού στη Βάση Δεδομένων

1.2.2. Σε αυτό το βήμα, θα υλοποιήσετε τις διαδικτυακές διεπαφές (html σελίδες) που θα χρησιμοποιούν οι χρήστες όλων των κατηγοριών (Ασθενείς, Ιατροί, Διαχειριστές) για να αλληλεπιδρούν με την εφαρμογή και να χρησιμοποιούν τις αντίστοιχες μεθόδους που απαιτούνται.

1.2.2.1. Θα υπάρχει ένα κεντρικό μενού σε μία index.html σελίδα, η οποία θα είναι η αρχική σελίδα για όλους τους χρήστες. Μετά το login θα προβάλλεται το μενού λειτουργιών κάθε χρήστη ανάλογα με την κατηγορία στην οποία ανήκει.

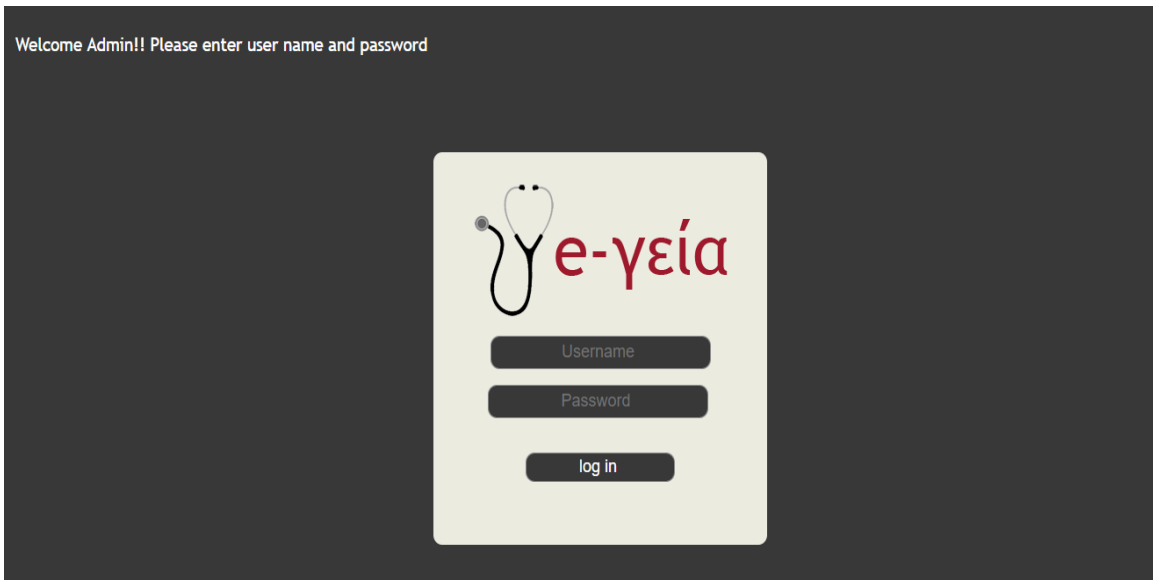
Η αρχική σελίδα της εφαρμογής είναι μια κοινή σελίδα index.jsp για όλους τους χρήστες και ανάλογα με την ιδιότητα τους οδηγούνται στο ανάλογο menu.



Εικόνα 7. Αρχική σελίδα εφαρμογής

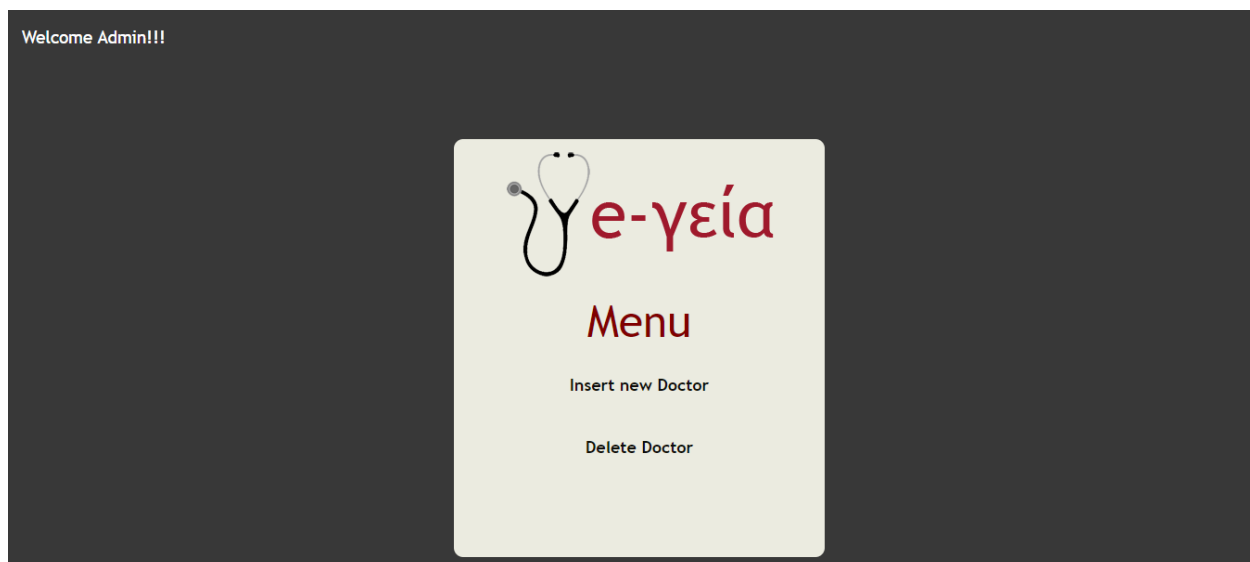
1.2.2.2. Λειτουργίες Διαχειριστή (Administrator). Οι Διαχειριστές θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: εισαγωγή νέου Ιατρού και διαγραφή Ιατρού.

Αφού ο χρήστης επιλέξει Login as Admin στην αρχική σελίδα της εφαρμογής οδηγείται σε μια φόρμα Login όπου πρέπει να εισάγει Username και Password για να εγγραφεί.



Εικόνα 8. Εγγραφή των Διαχειριστών μέσω της φόρμας Login

Πατώντας log in, ο χρήστης οδηγείται στο menu των Διαχειριστών έχοντας τις επιλογές της εισαγωγής νέου Ιατρού και της διαγραφής ενός υπάρχοντος Ιατρού.



Εικόνα 9. Κεντρικό menu Διαχειριστών



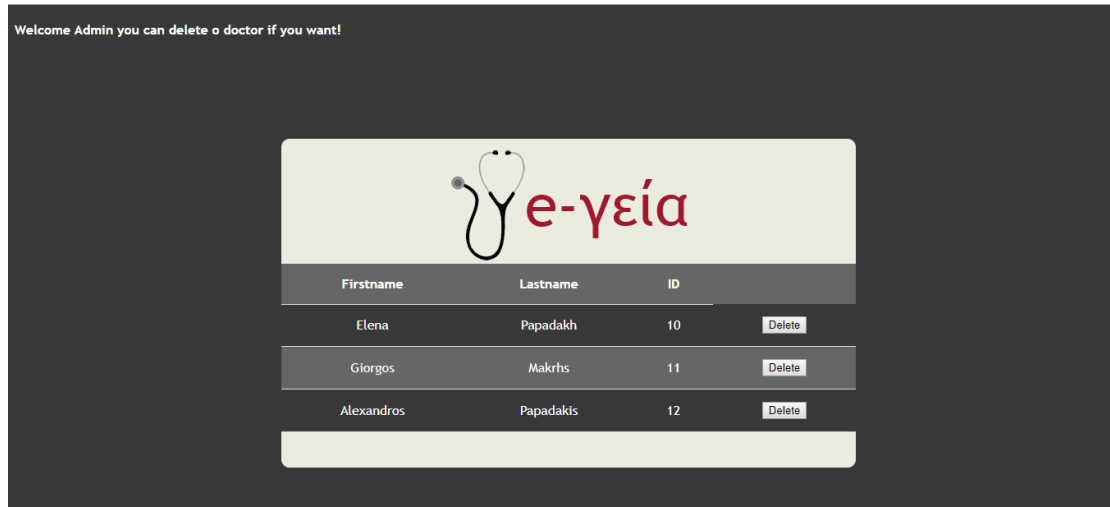
Πατώντας στο menu την επιλογή Insert new Doctor εμφανίζεται η αντίστοιχη φόρμα, όπως φαίνεται παρακάτω:

Εικόνα 10. Φόρμα εισαγωγής Ιατρού από τον Διαχειριστή στο σύστημα

και υπάρχουν 3 ειδικότητες που μπορεί να επιλέξει, pathologist, ophthalmologist και orthopedist, όπως φαίνεται στο παρακάτω drop-down menu:

Εικόνα 11. Οι διαθέσιμες ειδικότητες των Ιατρών

Πατώντας στο menu την επιλογή Delete Doctor εμφανίζεται η λίστα των ήδη υπάρχοντων Ιατρών με κουμπιά "Delete" για την δυνατότητα διαγραφής τους:



Εικόνα 12. Λίστα Ιατρών που μπορούν να διαγραφούν

1.2.2.3. Λειτουργίες Ιατρών (Doctor): Οι Ιατροί θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: καταχώρηση διαθεσιμότητας για ραντεβού (ανά μήνα), προβολή πίνακα με τις διαθέσιμες ημερομηνίες και ώρες (ανά μισάωρα) για ραντεβού που έχει εισάγει και δυνατότητα ακύρωσης αυτών.

Αφού ο χρήστης επιλέξει Login as Doctor στην αρχική σελίδα της εφαρμογής οδηγείται στην αντίστοιχη φόρμα του Ιατρού για να κάνει login (όπως και στον Admin) και στη συνέχεια οδηγείται στο αντίστοιχο menu, έχοντας τις επιλογές της καταχώρησης διαθεσιμότητας για ραντεβού, της προβολής του πίνακα με τις διαθέσιμες ημερομηνίες και ώρες για ραντεβού και της ακύρωσης των διαθέσιμων ημερομηνιών και ωρών για ραντεβού.



Εικόνα 13. Κεντρικό menu Ιατρών

Πατώντας στο menu την επιλογή Availability ο Ιατρός μπορεί να εισάγει τις ημερομηνίες και ώρες (ανά μισάωρα) που είναι διαθέσιμος για να δεχτεί τα ραντεβού του, καθώς και το όνομά του.

Hey Doctor! Here you can add your free time for your patients!




Day	Month	Year	Time	Doctor Name
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Insert"/>				

Εικόνα 14. Καταχώρηση διαθεσιμότητας για ραντεβού

Πατώντας στο menu την επιλογή Availability Schedule γίνεται προβολή του πίνακα με τις διαθέσιμες ημερομηνίες και ώρες (ανά μισάωρα) για ραντεβού, που έχει υποβάλει.

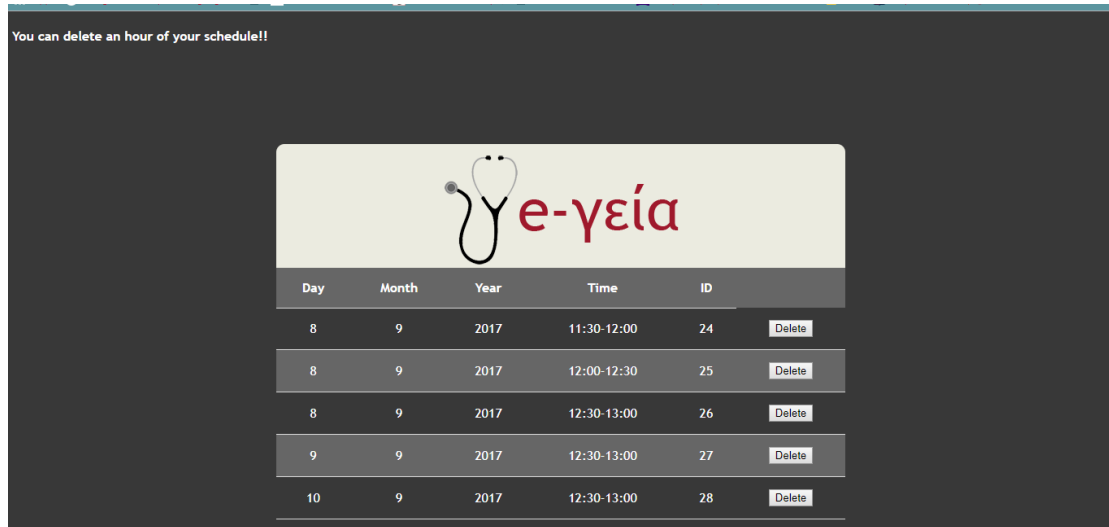
Here's your schedule!!



Day	Month	Year	Time
8	9	2017	11:30-12:00
8	9	2017	12:00-12:30
8	9	2017	12:30-13:00
9	9	2017	12:30-13:00
10	9	2017	12:30-13:00

Εικόνα 15. Πίνακας διαθέσιμων ημερομηνιών και ωρών για ραντεβού

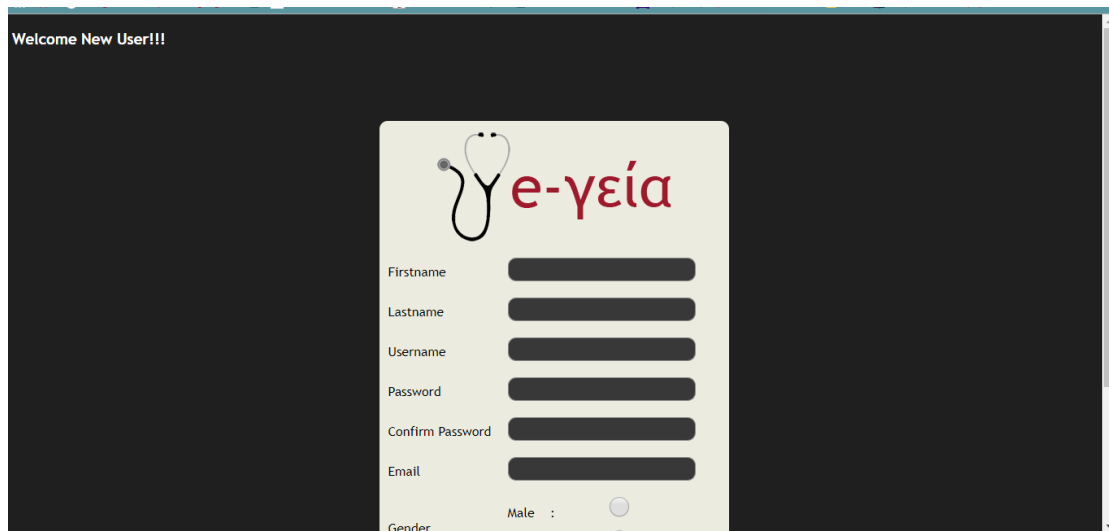
Πατώντας στο menu την επιλογή Cancel Free Hour ο Ιατρός μπορεί να διαγράψει τις διαθέσιμες ημερομηνίες και ώρες για ραντεβού που έχει υποβάλει.



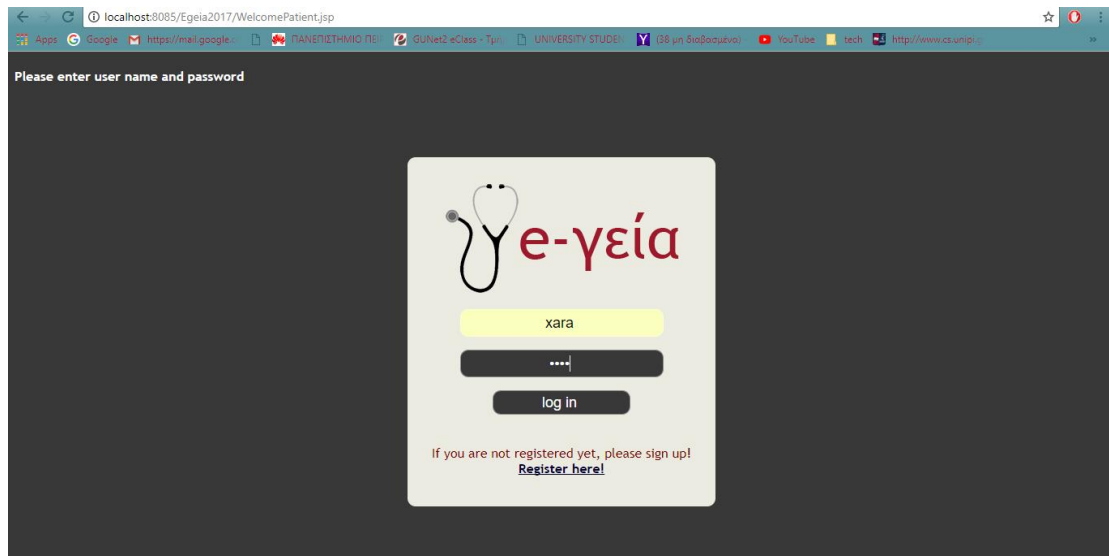
Εικόνα 16. Λίστα διαθέσιμων ημερομηνιών και ωρών για ραντεβού που μπορούν να διαγραφούν

1.2.2.4. Λειτουργίες Ασθενών (Patient): Οι Ασθενείς θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: προβολή ιστορικού προηγούμενων ραντεβού, προβολή διαθέσιμων των διαθέσιμων Ιατρών στο σύστημα, κλείσιμο ραντεβού, ακύρωση ραντεβού

Αφού ο χρήστης επιλέξει Login as Patient στην αρχική σελίδα της εφαρμογής οδηγείται στην αντίστοιχη φόρμα του Ασθενή για να κάνει login με το Username και το Password του. Όμως πρέπει πρώτα να έχει κάνει register, αλλιώς δεν μπορεί να εισαχθεί ποτέ. Οπότε πατάει "Register here!" και του εμφανίζεται η ανάλογη φόρμα για να εισάγει τα στοιχεία του και να υποβάλει το Username και το Password που θα χρησιμοποιεί.



Εικόνα 17. Φόρμα εισαγωγής στοιχείων του Ασθενή για να εγγραφεί στο σύστημα



Εικόνα 18. Σύνδεση των Ασθενών στο σύστημα μέσω της φόρμας Login

Πατώντας το κουμπί log in, ο χρήστης οδηγείται στο menu των Ασθενών έχοντας τις επιλογές της προβολής του πίνακα των κανονισμένων ραντεβού του, της προβολής του πίνακα των διαθέσιμων Ιατρών στο σύστημα, το κλείσιμο ραντεβού σε κάποια διαθέσιμη ημερομηνία και ώρα και την ακύρωση ενός προγραμματισμένου ραντεβού.



Εικόνα 19. Κεντρικό menu Ασθενών



Πατώντας στο menu την επιλογή Appointment History γίνεται προβολή της λίστας με τα προγραμματισμένα ραντεβού του

Here is your programmed appointments!!!

				
id	Day	Month	Year	Time
1	8	9	2017	11:30-12:00
3	9	9	2017	12:30-13:00

Εικόνα 20. Λίστα προγραμματισμένων ραντεβού του Ασθενή

Πατώντας στο menu την επιλογή E-geia Doctors γίνεται προβολή της λίστας των διαθέσιμων Ιατρών στο σύστημα.

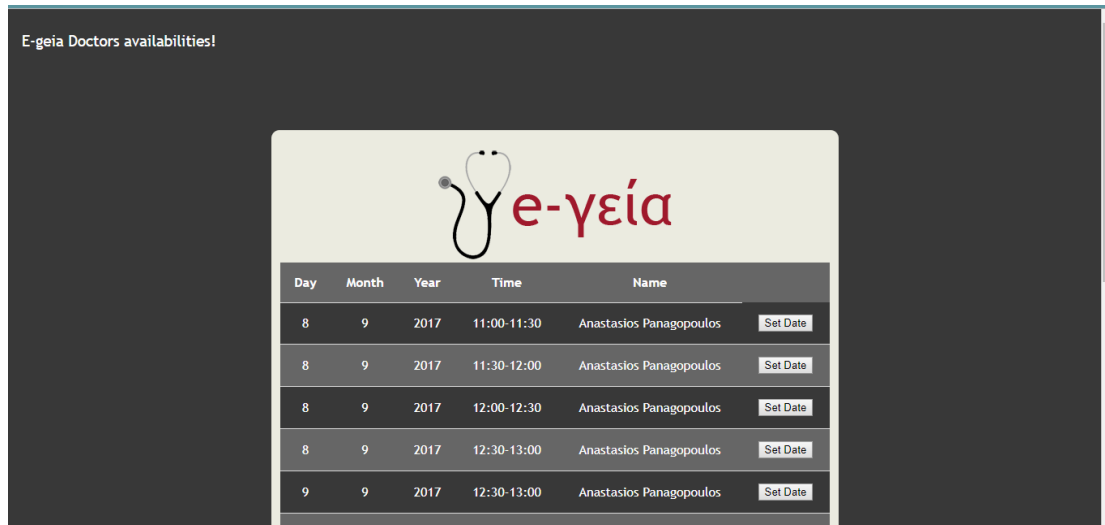
Doctors of e-geia are here for your health!!

		
Id	Name	Speciality
10	Elena Papadakh	pathologist
11	Giorgos Makrhs	ophthalmologist
12	Alexandros Papadakis	ophthalmologist
13	Anastasios Panagopoulos	orthopedist
14	Georgia Sarantou	ophthalmologist

Εικόνα 21. Λίστα των διαθέσιμων Ιατρών

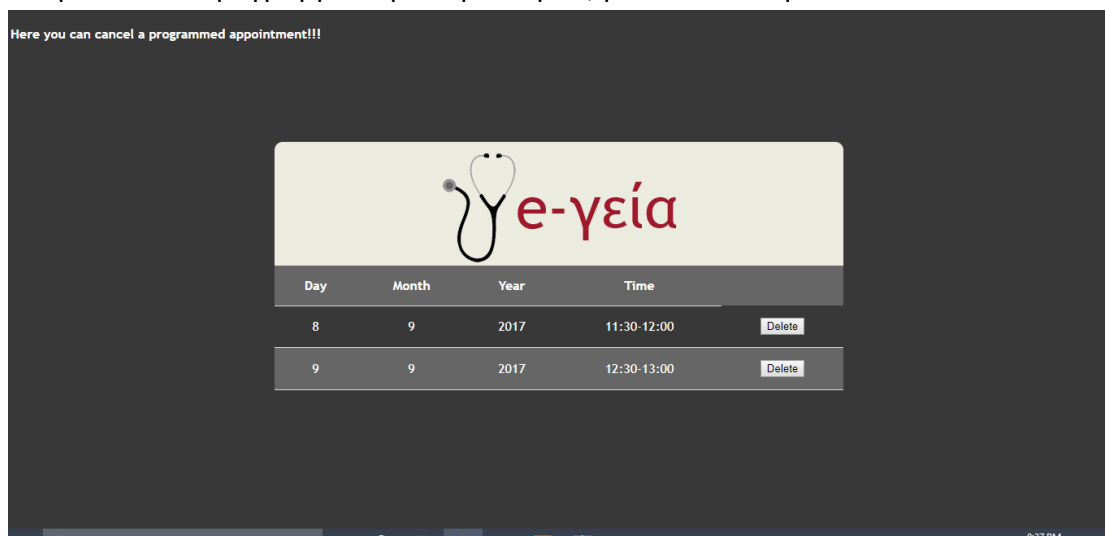


Πατώντας στο menu την επιλογή Set an Appointment ο Ασθενής έχει την επιλογή να κλείσει ένα ραντεβού σε κάποια διαθέσιμη ημερομηνία και ώρα και με τον Ιατρό που επιθυμεί, μέσω του κουμπιού Set Date (δεν έχει υλοποιηθεί η λειτουργία του).



Εικόνα 22. Κλείσιμο ραντεβού σε κάποια διαθέσιμη ημερομηνία και ώρα

Πατώντας στο menu την επιλογή Cancel Appointment ο Ασθενής μπορεί να ακυρώσει ένα προγραμματισμένο ραντεβού, μέσω του κουμπιού Delete.

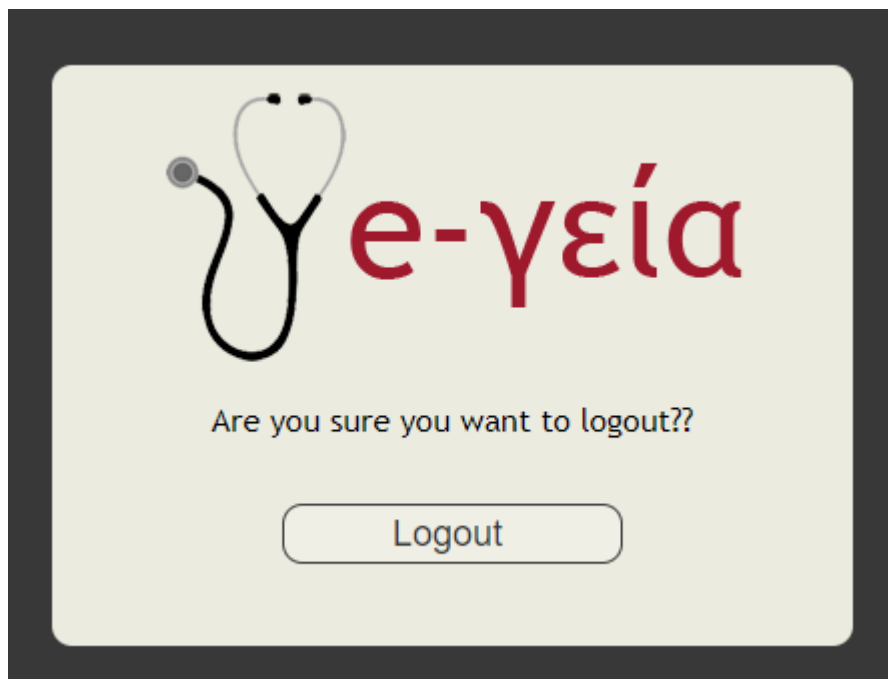


Εικόνα 23. Ακύρωση προγραμματισμένου ραντεβού



1.2.3. Η εφαρμογή θα υποστηρίζει διαχείριση συνόδου (session management) από τη στιγμή που ο χρήστης συνδέεται, μέχρι την αποσύνδεσή του από την εφαρμογή. Κατά την αποσύνδεση του χρήστη θα πρέπει να διαγράφεται το session.

Για να αποσυνδεθεί ο χρήστης από την εφαρμογή, δηλαδή για να κλείσει το session, πρέπει να κάνει Logout. Πατώντας το κουμπί Logout ενεργοποιούμε το servlet Logout.java, όπου κλείνει το session και στέλνει τον χρήστη στην αρχική σελίδα index.jsp.



Εικόνα 24. Το κουμπί Logout για αποσύνδεση του χρήστη από την εφαρμογή



Εικόνα 25. Το Javascript αρχείο Logout.jsp



Εικόνα 26.

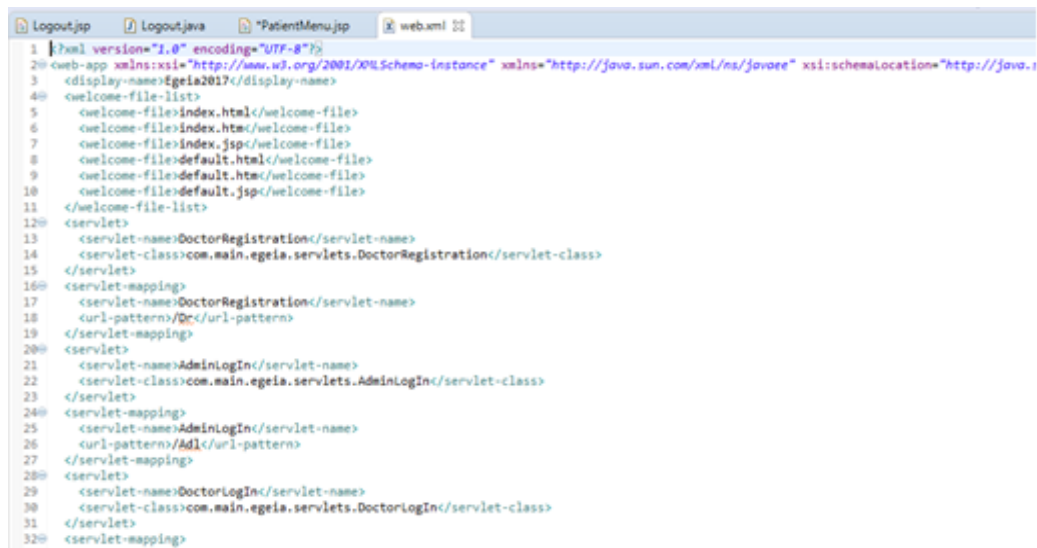
Για να αποφύγουμε την έξοδο του χρήστη με τη βοήθεια του back-button του browser χρησιμοποιήσαμε τον παρακάτω κώδικα javascript:

```
7 <a href="Logout.jsp" STYLE="text-decoration: none; color:#fffff">Log out</a>
8 <script>
9     window.location.hash="no-back-button";
10    window.location.hash="Again-No-back-button";//again because google chrome don't insert first hash into history
11    window.onhashchange=function(){window.location.hash="no-back-button";}
12 </script>
13 </body>
14 </html>
```

Εικόνα 27. Αποφυγή χρήσης back-button



Τέλος, το αρχείο web.xml ελέγχει τον τρόπο πρόσβασης σε ένα servlet.
Παρακάτω βλέπουμε τις διευθύνσεις URL που χρησιμοποιούνται για την κλήση αυτών των Servlet.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
3   <display-name>Egeia2017</display-name>
4   <welcome-file-list>
5     <welcome-file>index.html</welcome-file>
6     <welcome-file>index.htm</welcome-file>
7     <welcome-file>index.jsp</welcome-file>
8     <welcome-file>default.html</welcome-file>
9     <welcome-file>default.htm</welcome-file>
10    <welcome-file>default.jsp</welcome-file>
11  </welcome-file-list>
12  <servlet>
13    <servlet-name>DoctorRegistration</servlet-name>
14    <servlet-class>com.main.egeia.servlets.DoctorRegistration</servlet-class>
15  </servlet>
16  <servlet-mapping>
17    <servlet-name>DoctorRegistration</servlet-name>
18    <url-pattern>/Dr</url-pattern>
19  </servlet-mapping>
20  <servlet>
21    <servlet-name>AdminLogin</servlet-name>
22    <servlet-class>com.main.egeia.servlets.AdminLogin</servlet-class>
23  </servlet>
24  <servlet-mapping>
25    <servlet-name>AdminLogin</servlet-name>
26    <url-pattern>/Adl</url-pattern>
27  </servlet-mapping>
28  <servlet>
29    <servlet-name>DoctorLogin</servlet-name>
30    <servlet-class>com.main.egeia.servlets.DoctorLogin</servlet-class>
31  </servlet>
32  <servlet-mapping>
```

Εικόνα 28. Το αρχείο web.xml



1.3. Αλληλεπίδραση με τη Βάση Δεδομένων

1.3.1. Registrations αρχεία

Υπάρχουν δύο αρχεία Registration, το PatientRegistration.java και το DoctorRegistration.java, τα οποία ανήκουν στο πακέτο κλάσεων servlets. Θα χρησιμοποιήσουμε ως παράδειγμα το PatientRegistration.java για να εξηγήσουμε την αλληλεπίδραση με την Βάση Δεδομένων και τις σχετικές κλάσεις.

Αρχικά, το servlet PatientRegistration.java παίρνει τα δεδομένα μέσω της μεθόδου doPost που έχει εισάγει ο χρήστης στην φόρμα και στη συνέχεια τα στέλνει στην κλάση service PatientService.java, ώστε να γίνουν οι απαραίτητοι έλεγχοι (για κενές τιμές, ηλικία άνω των 100 χρόνων, password confirmation κλπ) μέσω της μεθόδου InsertPatient, καθώς και το hashing του κωδικού. Αφού εκτελεστούν οι έλεγχοι και ικανοποιηθούν όλες οι συνθήκες καλείται η κλάση repository PatientRepository.java και στην μέθοδο InsertPatient θα γίνει η εισαγωγή των στοιχείων στην Βάση. Η εισαγωγή των δεδομένων στη Βάση γίνεται με τον εξής τρόπο: στο PatientRegistration.java δηλώνεται μια μεταβλητή res, η οποία περνάει στο PatientService.java και, αν όλα είναι έγκυρα, καταλήγει στο PatientRepository.java, όπου του θέτουμε την τιμή -1. Αν η εισαγωγή των δεδομένων στη Βάση γίνει με επιτυχία, τότε η τιμή του res γίνεται > -1 και επιστρέφεται η τιμή στο PatientService.java και αυτό με την σειρά του το επιστρέφει στο PatientRegistration.java, όπου ενημερώνει τον Ασθενή ότι εγγράφηκε με επιτυχία. Αντίστοιχα, αν η εισαγωγή των δεδομένων στη Βάση δεν γίνει με επιτυχία, τότε ακολουθείται η ίδια διαδικασία και ο Ασθενής ενημερώνεται με μήνυμα ανεπιτυχούς εγγραφής.

Παρακάτω παραθέτουμε τα screenshots της παραπάνω διαδικασίας:



```
/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    String fn= request.getParameter("fname");
    String ln= request.getParameter("lname");
    String un= request.getParameter("uname");
    String pass= request.getParameter("password");
    String cpass= request.getParameter("cpass");
    String email= request.getParameter("email");
    String gender= request.getParameter("gender");
    String amka= request.getParameter("amka");
    int age= Integer.parseInt(request.getParameter("age"));

    Patient patient = new Patient(0,fn,ln,un,pass,cpass,email,gender,amka,age);
    PatientService service = new PatientService(); //apo edw phgainoume sto service gia tous aparaithτους elegxους
    int res = service.InsertPatient(patient);
    PrintWriter out = response.getWriter();
    if( res > -1 ) //apo to repository pairnoume apantsh to res kai an einai > -1 o userxei eggrafei epituxws sto susthma
        response.getWriter().append("Successfully inserted patient with id : " + patient.getFirstName());
    else
        response.getWriter().append("Unsuccessfully inserted patient");

}
```

Εικόνα 29. Στο servlet PatientRegistration.java παίρνουμε τα δεδομένα που έχει εισάγει ο χρήστης στη φόρμα

```
10 public class PatientService {
11
12
13 private PatientRepository repo = new PatientRepository();
14
15
16
17 public int InsertPatient(Patient patient)
18 {
19     int res = -1;
20     if( patient.getFirstName().isEmpty() || //meta to servlet exomaste edw opou ginontai oi aparaithtoi elegxoi
21         patient.getLastName().isEmpty() || //gia adeies symboloseores , password confirmation klp
22         patient.getEmail().isEmpty() ||
23         patient.getGender().isEmpty() ||
24         patient.getPassword().isEmpty() ||
25         patient.getCPassword().isEmpty() ||
26         patient.getUsername().isEmpty() ||
27         patient.getAge()>100 ||
28         patient.getAmka().isEmpty() || patient.getAmka().length()>11 ||
29         !(patient.getPassword().equals(patient.getCPassword()))
30     )
31     {
32         return res;
33     }
34     if( patient.getPassword().length() < 64 //kathws kai hashing ston kwdiko
35         //Need to hash password - Dummy check //ef oson ola einai ok phgainoume sto repository
36         patient.setPassword(PasswordEncoder.encode(patient.getPassword())); //gia na ginei h eisagwgh sth bash
37     return repo.InsertPatient(patient);
38 }
39
40
41
42 public Patient Login(String username, String password)
43 {
```

Εικόνα 30. Στην κλάση PatientService.java γίνονται οι απαραίτητοι έλεγχοι των δεδομένων



```
11
12 public class PatientRepository {
13
14     public int InsertPatient(Patient patient)
15     {
16         //στο repository γίνεται η εισαγωγή στη βάση το οποίο αλληλεπιδρά με το service
17         //αφού έχουν γίνει οι απαραίτητοι έλεγχοι σουνδεσμεύεται με τη βάση και το συγκεκριμένο table
18         //και εισαγόμενα τα δεδομένα
19
20         int res = -1; //θέτουμε την μεταβλητή res = -1
21         String sql="insert into patients(firstname,lastname,username,password,email,gender,amka,age)values(?,?,?,?,?,?,?)";
22         Connection con=DbConnection.getMysqlConnection();
23         PreparedStatement ps;
24         try {
25             ps = con.prepareStatement(sql);
26             ps.setString(1, patient.getFirstName());
27             ps.setString(2, patient.getLastName());
28             ps.setString(3, patient.getUserName());
29             ps.setString(4, patient.getPassword());
30             ps.setString(5, patient.getEmail());
31             ps.setString(6, patient.getGender());
32             ps.setString(7, patient.getAmka());
33             ps.setInt(8,patient.getAge());
34             res = ps.executeUpdate(); //και αν έχει γίνει επιτυχώς το insertion τότε το res θα είναι > -1
35
36         } catch (SQLException e) {
37             // TODO Auto-generated catch block
38             e.printStackTrace();
39         }
40
41         return res; //επιστρέφουμε στο servlet την τιμή του res
42     }
43
44 }
```

Εικόνα 31. Στην κλάση PatientRepository.java επιχειρείται η σύνδεση με την Βάση και η εισαγωγή των δεδομένων σ' αυτήν

1.3.2. Login αρχεία

Στα Login αρχεία ακολουθείται μια παρόμοια διαδικασία που περιγράφουμε παρακάτω. Ως παράδειγμα έχουμε επιλέξει το PatientLogin.java:

Αρχικά, το servlet PatientLogin.java παίρνει το username και το password από την φόρμα που έχει συμπληρώσει ο Ασθενής και καλεί τη μέθοδο Login της κλάσης service PatientService.java. Η μέθοδος αυτή κάνει ελέγχους για μη συμπληρωμένα πεδία στην φόρμα από τον χρήστη και για το hashing του κωδικού, και στην περίπτωση που οι έλεγχοι είναι έγκυροι καλεί τη μέθοδο Login της κλάσης repository PatientRepository.java, η οποία βρίσκει τον χρήστη αν υπάρχει στη Βάση και κρατάει τα στοιχεία του για περαιτέρω χρήση. Αν δεν υπάρξει κάποιο πρόβλημα, επιστρέφει την τιμή patient στο PatientService.java και αυτό με τη σειρά του την επιστρέφει στο PatientLogin.java, το οποίο αν πάρει σωστή τιμή ανοίγει την .jsp σελίδα.

Παρακάτω παραθέτουμε τα screenshots της παραπάνω διαδικασίας:



```
/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

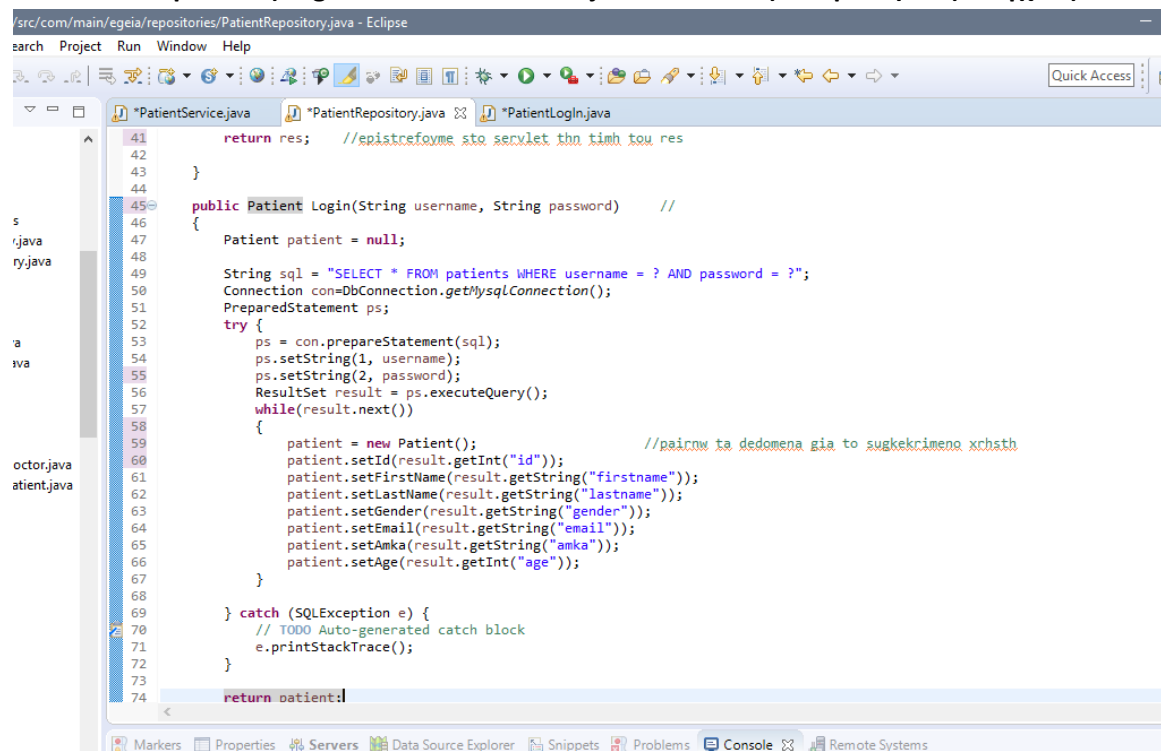
    String username = request.getParameter("uname");           //trabaw apo th forma username kai password
    String password = request.getParameter("password");

    Patient pat = service.Login(username, password);           //phgainw sth methodo log in tou service gia elegxo
    if (pat != null )
    {
        HttpSession session = request.getSession(true);
        SessionUtils.setSessionUserId(session, pat.getId());   //krataw id tou xrhsth
        response.sendRedirect("PatientMenu.jsp");
    } else
    {
        response.sendRedirect("WelcomePatient.jsp");
    }
}
}
```

Εικόνα 32. Ο servlet PatientLogin.java παίρνει τα στοιχεία από την φόρμα που έχει συμπληρώσει ο Ασθενής

```
public Patient Login(String username, String password)
{
    Patient pat = null;           //elegxos an einai kena ta insertion tou xrhsth
    if( username.isEmpty() || password.isEmpty() ) return pat;
    //Needs to hash it
    String hashed_pass = PasswordEncoder.encode(password); //kai elegxos gia hash
    pat = repo.Login(username, hashed_pass); //paw sto repository gia na brw ton xrhsth sth bash
    return pat;
}
```

Εικόνα 33. Η μέθοδος Login στο PatientService.java κάνει τους απαραίτητους ελέγχους



Εικόνα 34. Στην κλάση PatientRepository.java επιχειρείται η σύνδεση με την Βάση και η εισαγωγή των δεδομένων σ' αυτήν

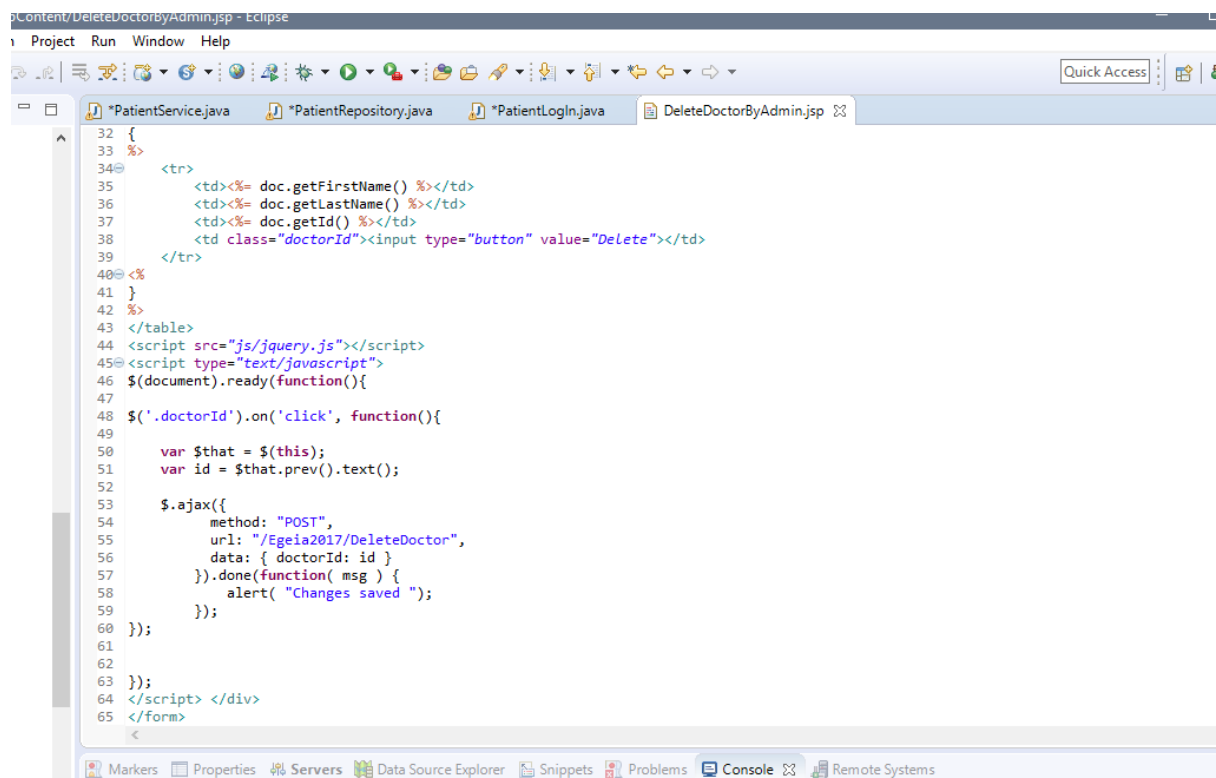


1.3.3. Delete αρχεία

Η διαδικασία που ακολουθείται στα Delete αρχεία περιγράφεται παρακάτω και έχουμε χρησιμοποιήσει ως παράδειγμα το DeleteDoctorByAdmin.jsp:

Αρχικά, γι' αυτήν την ενέργεια χρησιμοποιούμε Javascript, και γι' αυτόν τον λόγο χρειαζόμαστε τη βιβλιοθήκη jquery. Με βάση τη Javascript, όταν πατηθεί το κουμπί "delete" κρατάμε το id του Ιατρού που θέλει ο Διαχειριστής να διαγράψει και καλούμε το servlet DeleteDoctor.java. Σε αυτό το σημείο παίρνουμε το doctorId από το DeleteDoctorByAdmin.jsp και καλείται το service DoctorService.java για να γίνει ο έλεγχος ύπαρξης id. Αντίστοιχα, το DoctorService.java καλεί τη μέθοδο DeleteDoctor() του repository DoctorRepository.java, όπου τελικά συνδέεται με τη Βάση και πηγαίνει στον πίνακα Doctors και διαγράφει τον Ιατρό με το συγκεκριμένο id.

Παρακάτω παραθέτουμε τα screenshots της παραπάνω διαδικασίας:



```
32 {
33 }
34 <tr>
35 <td><%= doc.getFirstName() %></td>
36 <td><%= doc.getLastName() %></td>
37 <td><%= doc.getId() %></td>
38 <td class="doctorId"><input type="button" value="Delete"></td>
39 </tr>
40 <%>
41 }
42 <%>
43 </table>
44 <script src="js/jquery.js"></script>
45 <script type="text/javascript">
46 $(document).ready(function(){
47
48 $('.doctorId').on('click', function(){
49
50     var $that = $(this);
51     var id = $that.prev().text();
52
53     $.ajax({
54         method: "POST",
55         url: "/Egeia2017/DeleteDoctor",
56         data: { doctorId: id }
57     }).done(function( msg ) {
58         alert( "Changes saved ");
59     });
60 });
61
62 });
63 </script> </div>
64 </form>
65 </form>
```

Εικόνα 35. Το javascript αρχείο DeleteDoctorByAdmin.jsp το οποίο κρατάει το id του Ιατρού και καλεί το servlet DeleteDoctor.java



```
1 DeleteDoctor.java 2 DoctorRepository.java 3 DoctorService.java
30 public DeleteDoctor() {
31     super();
32     // TODO Auto-generated constructor stub
33 }
34
35 /**
36  * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
37 */
38 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
39     // TODO Auto-generated method stub
40     //response.getWriter().append("Served at: ").append(request.getContextPath());
41     request.getRequestDispatcher("/WEB-INF/DeleteDoctorByAdmin.jsp").forward(request, response);
42 }
43
44 /**
45  * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
46 */
47 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
48     String id = request.getParameter("doctorId"); //pairnw apo to jsp to doctor id
49
50     DoctorService service = new DoctorService(); // kai kalw to doctor service
51     boolean res = service.DeleteDoctor(Integer.parseInt(id)); //kalw th delete doctor tou DoctorService
52     PrintWriter out = response.getWriter();
53     out.print(res);
54     out.flush();
55 }
56
57 }
```

Εικόνα 36. Το servlet DeleteDoctor.java παίρνει το doctorId από το .jsp αρχείο και καλεί τη μέθοδο DeleteDoctor() του DoctorService.java

```
1 package com.main.egia.zei.vase;
2
3 import java.util.List;
4
5 public class DoctorService {
6
7     private DoctorRepository repo = new DoctorRepository();
8
9     public boolean DeleteDoctor(int id)
10     {
11         boolean res = false; //kanw elegxo an uparxei ontws krathmeno id kai kalw DoctorRepository
12         if (id < 0)
13             return res;
14
15         return repo.DeleteDoctor(id);
16     }
17
18     public int InsertDoctor(Doctor doctor)
19     {
20
21     }
22 }
```

Εικόνα 37. Έλεγχος ύπαρξης του id στο DoctorService.java και κλήση της μεθόδου DeleteDoctor() του DoctorRepository.java

```
12 public class DoctorRepository {
13
14
15
16
17     public boolean DeleteDoctor(int id) //edw diagrafetai apo th bash o iatros me to sugkekrimeno id
18     {
19         boolean res = false;
20         Connection conn = DbConnection.getMysqlConnection();
21         String loginquery = "DELETE FROM Doctors WHERE id = ?";
22         try {
23             PreparedStatement pstmt = conn.prepareStatement(loginquery);
24             pstmt.setInt(1, id);
25             res = pstmt.executeUpdate() == 0;
26
27         } catch (SQLException e) {
28             // TODO Auto-generated catch block
29             e.printStackTrace();
30         }
31
32         try {
33             conn.close();
34         } catch (SQLException e) {
35             // TODO Auto-generated catch block
36             e.printStackTrace();
37         }
38         return res;
39     }
40 }
```

Εικόνα 38. Διαγραφή του Ιατρού από τη Βάση με το συγκεκριμένο id

