ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος «Προγραμματισμός στο διαδίκτυο και στον παγκόσμιο ιστό»

Άσκηση:	***Τελική Εργασία***
Όνομα φοιτητή – Αρ. Μητρώου (όλων σε περίπτωση ομαδικής εργασίας)	Ευάγγελος Αβδελάς – Π15009
	Δημήτριος Ζωγραφάκης – Π15041
	Νικόλαος Μάμης – Π15175
Ημερομηνία παράδοσης	04/07/2017

Εκφώνηση της άσκησης

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΤΟ ΔΙΑΔΙΚΤΥΟ ΚΑΙ ΣΤΟΝ ΠΑΓΚΟΣΜΙΟ ΙΣΤΟ

ΕΡΓΑΣΙΑ ΜΑΘΗΜΑΤΟΣ 2016-17

ΤΕΛΙΚΟ ΠΑΡΑΔΟΤΕΟ ΜΑΘΗΜΑΤΟΣ

Δημιουργία 3-tier εφαρμογής για την διαχείριση ραντεβού ιατρικών εξετάσεων

Στόχοι εργασίας: Ολοκλήρωση λειτουργικότητας 3-tier εφαρμογής, ολοκλήρωση server-side τεχνολογιών

(servlets και προαιρετικά jsp), επικοινωνία με βάση δεδομένων, ολοκλήρωση λειτουργιών.

Στην τελική εργασία του μαθήματος θα επεκτείνετε τις προηγούμενες ασκήσεις ώστε να δημιουργήσετε μία

εφαρμογή τριών επιπέδων (3-tier), η οποία θα υλοποιεί όλες τις λειτουργίες (μεθόδους) που ορίσατε στις

προηγούμενες ασκήσεις.

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 υλοποιείστε 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. Προσθέσετε 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 λύσης

Πρόκειται για μια εφαρμογή τριών επιπέδων. Ο βασικός κορμός της εφαρμογής είναι τα servlet τα οποία εκτελούν όλες τις λειτουργίες που απαιτούνται. Αυτά συνδέονται με την βάση δεδομένων η οποία έχει γραφτεί σε PostgreSQL.

Κάθε servlet έχει τις δικές του λειτουργίες και υπάρχουν διαφορετικά servlet για τους ασθενείς, τους γιατρούς, τους διαχειριστές, την εγγραφή και την διαγραφή χρηστών.

Τέλος υπάρχει και servlet που δημιουργεί τις απαραίτητες δυναμικές HTML σελίδες που χρειάζεται η εφαρμογή μας.

1.1 Σύνδεση με βάση δεδομένων

Η σύνδεση με τη βάση δεδομένων, πέρα από τις εντολές που απαιτεί σε κάθε servlet, απαιτεί και μια συγκεκριμένη προετοιμασία όσον αφορά τον Apache Tomcat Server.

Συγκεκριμένα πρέπει να γράψουμε στο αρχείο context.xml του server τις ακόλουθες εντολές.

```
<Resource name="jdbc/LiveDataSource" auth="Container"
 type="javax.sql.DataSource"
 username="postgres" password="root" driverClassName="org.postgresql.Driver"
 url="jdbc:postgresql://localhost:5433/Askisi02"
 global="jdbc/LiveDataSource"/>
```

1.2 Μέθοδοι που επαναχρησιμοποιήθηκαν

Μέσα στην εφαρμογή υπάρχουν ορισμένες μέθοδοι και εντολές που χρησιμοποιούνται περισσότερες από μια φορές όπως είναι αυτή της σύνδεσης με τη βάση. Η συγκεκριμένη εντολή καλείται κάθε φορά που επιλέγεται μια λειτουργία που απαιτεί σύνδεση με τη βάση δεδομένων προκειμένου να αποκτήσει πρόσβαση στις πληροφορίες της. Επίσης κατά τη δημιουργία ενός servlet εκτελείται μια μέθοδος που εξασφαλίζει ότι δημιουργήθηκε η σύνδεση με τη βάση δεδομένων.

```
public void init() throws ServletException {
    try {
        InitialContext ctx = new InitialContext();
        datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/LiveDataSource");
    } catch(Exception e) {
        throw new ServletException(e.toString());
    }
}
```

1.3 Servlet mapping

Τέλος για να τρέξει σωστά η εφαρμογή μας πρέπει να ορίσουμε την ονομασία των servlets στα urls, το οποίο επιτυγχάνεται με τις ανάλογες εντολές στο αρχείο web.xml του project μας.

```
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
</welcome-file-list>
<resource-ref>
  <description>Connection Pool</description>
  <res-ref-name>jdbc/LiveDataSource</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
<servlet>
  <description></description>
  <display-name>PatientServlet</display-name>
  <servlet-name>PatientServlet</servlet-name>
  <servlet-class>servlets.PatientServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>PatientServlet</servlet-name>
  <url-pattern>/patientservlet</url-pattern>
</servlet-mapping>
<servlet>
  <description></description>
  <display-name>DoctorServlet</display-name>
  <servlet-name>DoctorServlet</servlet-name>
  <servlet-class>servlets.DoctorServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>DoctorServlet</servlet-name>
  <url-pattern>/doctorservlet</url-pattern>
</servlet-mapping>
```

Προφανώς η ίδια διαδικασία συνεχίζεται και με τα υπόλοιπα servlets.

1.4 Φόρμες και requestType

Το μεγαλύτερο κομμάτι της εφαρμογής έχει δημιουργηθεί με βάση τις φόρμες και την λειτουργία τους form action.

Κάθε κουμπί βρίσκεται μέσα σε μια φόρμα με διαφορετικό action δηλαδή διαφορετικό προορισμό των δεδομένων που στέλνει.

Επίσης κάθε κουμπί έχει το χαρακτηριστικό name=”requestType” και ένα value ανάλογα με την λειτουργία που εκτελεί. Αυτό μας επιτρέπει να έχουμε άμεση πρόσβαση στις λειτουργίες μέσω της τιμής value που καθορίζει τι θα κάνει κάθε λειτουργία.

Παραδείγματα κώδικα:

```
<form method="post" action="./loginservlet" name="form-login">
    <span class="fontawesome-user"></span>
        <input type="text" size="40" maxlength="40" name="username" placeholder="Username"/><br>
    <span class="fontawesome-lock"></span>
        <input type="password" size="40" maxlength="40" name="password" placeholder="Password" /><br>
    <input type="SUBMIT" name="requestType" value="Login">
```

1 Κώδικας από τη φόρμα στη σελίδα login.html

```
String requestType= request.getParameter("requestType");

if (requestType == null) {
    dynPage.DynamicPageError(response, "Invalid request type","./index.html");
}
if (requestType.equalsIgnoreCase("Login")){
    String username = request.getParameter("username");
    String password = request.getParameter("password");
    char user=' ';
    boolean checkUsername = true;
    boolean checkPassword = true;
```

2 Κώδικας από το LoginServlet που χρησιμοποείται το requestType

```
<form>
    <input type="submit" formaction="./patientservlet" formmethod="post" name="requestType" value="View Info"></input><br><br>
    <input type="submit" formaction="./patientservlet" formmethod="get" name="requestType" value="View Appointments"></input><br><br>
    <input type="submit" formaction="./newappointment.html" value="New appointment"></input><br><br>
    <input type="submit" formaction="./patientservlet" formmethod="get" name="requestType" value="Cancel Appointment"></input><br><br>
    <input type="submit" formaction="./patientservlet" formmethod="post" name="requestType" value="Logout"></input>
</form>
```

3 Κώδικας από τη φόρμα στο patient.html

```
if (requestType.equalsIgnoreCase("View Appointments")){
    String apps;
    Timestamp currentDate = new Timestamp(System.currentTimeMillis());
    String status="";
```

4 PatientServlet λειτουργία View Appointments

```
//Cancel an appointment
if (requestType.equalsIgnoreCase("Cancel Appointment")) {
    String table;
    Timestamp currentDate = new Timestamp(System.currentTimeMillis());
```

5 PatientServlet λειτουργία Cancel Appointment

```
if (requestType.equalsIgnoreCase("View Info")){
```

6 PatientServlet λειτουργία View Info

2 Κώδικας προγράμματος

Ακολουθεί η αναλυτική περιγραφή του προγράμματος.

2.1 Κώδικας του servlet LoginServlet

Το LoginServlet είναι το πρώτο servlet που θα αντικρύσει ο χρήστης αν επιλέξει Login στην αρχική σελίδα της εφαρμογής, index.html.

Το LoginServlet περιλαμβάνει τον κώδικα για την σύνδεση του χρήστη στην εφαρμογή, είτε αυτός είναι ασθενής, είτε γιατρός, είτε διαχειριστής.

Αρχικά δέχεται σαν παραμέτρους την απάντηση του χρήστη στα textboxes username και password της σελίδας login.html. Στη συνέχεια ελέγχει με τη σειρά τους πίνακες patient, doctor και admin της βάσης δεδομένων. Ανάλογα σε ποιο πίνακα θα βρει τον χρήστη (το username και το password θα υπάρχουν στον αντίστοιχο πίνακα της βάσης) , αποθηκεύει σε μια μεταβλητή τον αντίστοιχο χαρακτήρα p για patient, d για doctor, a για admin.

Στη συνέχεια ελέγχει ότι ο χρήστης όντως βρέθηκε μέσω δύο μεταβλητών Boolean που η τιμή τους αλλάζει όταν βρεθεί ο χρήστης με το σωστό username και password σε οποιονδήποτε από τους 3 πίνακες. Αν ο χρήστης βρέθηκε, τότε ανάλογα με τον χαρακτήρα που αποθηκεύτηκε πριν, η εφαρμογή σε προωθεί στο αντίστοιχο μενού επιλογών.

Αν ο χρήστης δεν βρέθηκε τότε η εφαρμογή σε στέλνει πίσω στο αρχικό μενού.

2.2 Κώδικας του servlet DynamicPages

Ένα από τα πιο βασικά servlet της εφαρμογής μας καθώς περιλαμβάνει τον κώδικα για τις δυναμικές σελίδες που δημιουργούνται από τις διάφορες λειτουργίες. Στο servlet αυτό δε χρειάζεται να γίνει κάποια σύνδεση με τη βάση δεδομένων καθώς υπάρχουν μόνο μέθοδοι οι οποίες περιλαμβάνουν κώδικα html.

Οι μέθοδοι αυτές είναι οι εξής:

1. Δυναμική δημιουργία νέου ασθενή από τον διαχειριστή: Η μέθοδος αυτή περιλαμβάνει μια φόρμα που συμπληρώνεται από τον διαχειριστή προκειμένου να προσθέσει στο σύστημα έναν νέο ασθενή.
2. Δυναμική δημιουργία νέου γιατρού από τον διαχειριστή: Η μέθοδος αυτή περιλαμβάνει μια φόρμα που συμπληρώνεται από τον διαχειριστή προκειμένου να προσθέσει στο σύστημα έναν νέο γιατρό.
3. Δυναμική διαγραφή γιατρού από τον διαχειριστή: Η μέθοδος αυτή περιλαμβάνει μια φόρμα που συμπληρώνεται από τον διαχειριστή προκειμένου να διαγράψει από το σύστημα έναν γιατρό. Σημειώνεται ότι μαζί με τον γιατρό θα διαγραφούν και όλα τα ραντεβού που είναι συνδεδεμένα με αυτόν.

4. Δυναμική σελίδα σφάλματος: Μια σελίδα που εμφανίζεται στον χρήστη όταν υπάρχει κάποιο σφάλμα. Αυτό περιλαμβάνει λάθη κατά την είσοδο δεδομένων από το χρήστη, αδυναμία σύνδεσης με τη βάση και οποιοδήποτε άλλο σφάλμα μπορεί να προκύψει. Παρέχεται ένα κουμπί ώστε ο χρήστης να πάει πίσω στην κύρια σελίδα ή στην αντίστοιχη σελίδα επιλογής λειτουργιών του αν έχει ήδη συνδεθεί.
5. Δυναμική σελίδα επιτυχίας: Μια σελίδα που εμφανίζεται στον χρήστη όταν πραγματοποίησε μια ενέργεια επιτυχώς. Για παράδειγμα, διαγραφή ραντεβού, κλείσιμο ραντεβού, προγραμματισμό ραντεβού (από τον γιατρό). Παρέχεται ένα κουμπί ώστε ο χρήστης να πάει πίσω στην αντίστοιχη σελίδα επιλογής λειτουργιών του.
6. Δυναμικές γραμμές πίνακα: Υπάρχουν αρκετές μέθοδοι που δημιουργούν δυναμικά τις γραμμές του πίνακα ανάλογα με την λειτουργία που έχει επιλέξει ο χρήστης. Για παράδειγμα, προβολή των διαθέσιμων ραντεβού ή τον ραντεβού που έχει προγραμματίσει.

2.3 Κώδικας του servlet AddUser

Στο servlet AddUser υπάρχει ο κώδικας για την εγγραφή ενός νέου ασθενή αν αυτός επιλέξει την επιλογή register στο αρχικό μενού, ο κώδικας για την εγγραφή ενός νέου ασθενή από τον διαχειριστή και ο κώδικας για την εισαγωγή ενός γιατρού από τον διαχειριστή.

Για την εγγραφή οποιουδήποτε χρήστη ελέγχεται αρχικά αν όλα τα πεδία έχουν συμπληρωθεί, αν όχι τον προτρέπει να ξεκινήσει από την αρχή. Στην εγγραφή ασθενών υπάρχει και ο έλεγχος το ΑΜΚΑ να είναι αριθμός.

Αν περάσει από τους ελέγχους τότε η εφαρμογή εισάγει τον νέο χρήστη στη βάση δεδομένων. Αν ο ασθενής έκανε μόνος του την εγγραφή του από το αρχικό μενού, τότε η εφαρμογή τον συνδέει αυτόματα και του εμφανίζει το μενού του ασθενή. Σε κάθε άλλη περίπτωση προτρέπει τον χρήστη να επιστρέψει στο μενού του διαχειριστή.

2.4 Κώδικας του servlet RemoveUser

Το servlet RemoveUser είναι παρόμοιο σε λειτουργία με το servlet AddUser. Η λειτουργία του είναι να διαγράφει τους γιατρούς από το σύστημα.

Οι λειτουργίες των παραπάνω servlet υλοποιούνται με την μέθοδο doPost καθώς δε θέλουμε να προβληθούν ευαίσθητες πληροφορίες στην διεύθυνση url.

2.5 Κώδικας του servlet PatientServlet

Το PatientServlet είναι το βασικό servlet που χρησιμοποιεί ο ασθενής για τις λειτουργίες του. Οι λειτουργίες είναι μοιρασμένες, δηλαδή άλλες χρησιμοποιούν την μέθοδο doGet και άλλες την doPost.

Στο servlet Patient έχουμε δημιουργήσει και μια μέθοδο η οποία λαμβάνει το ΑΜΚΑ του ασθενή όταν αυτός συνδέεται προκειμένου να χρησιμοποιηθεί σε κάποιες από τις λειτουργίες του.
Μια από τις λειτουργίες που βρίσκονται μέσα στη μέθοδο doPost είναι αυτή της προβολής των πληροφοριών του χρήστη. Η μόνη πληροφορία που δεν εμφανίζεται στον πίνακα είναι ο κωδικός του χρήστη.

Στην doPost βρίσκονται επίσης οι λειτουργίες ακύρωσης ραντεβού και προγραμματισμού ενός ραντεβού.

Η λειτουργία View Info δημιουργεί έναν δυναμικό πίνακα τον οποίο γεμίζει με τις πληροφορίες που δέχεται μετά την εκτέλεση του query. Τελικά "στέλνει" τον συμπληρωμένο πίνακα στην σελίδα info.jsp.

Προκειμένου να δει τα ραντεβού του ο χρήστης, επιλέγει την λειτουργία View Appointments η οποία με τη σειρά της δημιουργεί έναν δυναμικό πίνακα και προσθέτει δυναμικά γραμμές ανάλογα με το πόσα ραντεβού έχει προγραμματίσει ο χρήστης. Στον πίνακα αυτόν φαίνονται τόσο τα περασμένα όσο και τα προγραμματισμένα ραντεβού. Αυτό το επιτυγχάνουμε λαμβάνοντας τη σημερινή ημερομηνία με την εντολή:

```
Timestamp currentDate = new Timestamp(System.currentTimeMillis());
```

Και συγκρίνοντας την με την ημερομηνία του ραντεβού που λάβαμε από την εκτέλεση του query. Τελικά και αυτός ο πίνακας προωθείται στην σελίδα viewappointments.jsp.

Αν ο χρήστης επιλέξει την λειτουργία Schedule Appointment τότε η εφαρμογή θα τον κατευθύνει στην σελίδα newappointment.html προκειμένου να επιλέξει την ειδικότητα του γιατρού με τον οποίο θέλει να κλείσει ραντεβού. Μόλις επιλέξει ειδικότητα τότε η φόρμα τον επιστρέφει στο patientServlet μέσω της μεθόδου GET όπου εκτελώντας ένα query βρίσει όλα τα διαθέσιμα ραντεβού με γιατρούς αυτής της ειδικότητας. Διαθέσιμα ραντεβού είναι όλα αυτά που στη βάση δεδομένων δεν έχουν αριθμό ΑΜΚΑ ασθενούς.

Στη συνέχεια δημιουργείται αυτόματα ένας δυναμικός πίνακας με όλες τις ημερομηνίες και τα ονόματα των γιατρών προκειμένου να επιλέξει την ημερομηνία που θέλει ο χρήστης.

Η λειτουργία Cancel Appointment εχει υλοποιηθεί παρόμοια με την Schedule Appointment με μονη διαφορά ότι δε χρειάζεται να δημιουργηθεί αλλή html σελίδα πέρα από την δυναμική σελίδα με τον πίνακα που περιλαμβάνει όλα τα ραντεβού που έχει κλείσει ο χρήστης (εκτός από αυτά που έχουν περάσει). Για να ακυρώσει ένα ραντεβού απλά αφαιρείται το ΑΜΚΑ του ασθενή από το αντίστοιχο ραντεβού στον πίνακα appointments.

Τελευταία λειτουργία του ασθενή είναι το Logout το οποίο κάνει invalidate το session που έχει δημιουργηθεί, και στέλνει τον χρήστη στην αρχική σελίδα εισόδου.

2.6 Κώδικας του servlet DoctorServlet

Το DoctorServlet έχει υλοποιηθεί με πολλές ομοιότητες με το PatientServlet.

Πρώτη λειτουργία του είναι η προβολή των πληροφοριών του γιατρού, δηλαδή η δημιουργία ενός δυναμικού πίνακα με τα στοιχεία που επιστρέφει το query.

Δεύτερη λειτουργία είναι η προβολή των ραντεβού του. Στην περίπτωση του γιατρού φαίνονται μόνο τα ραντεβού που είναι προγραμματισμένα και όχι αυτά που έχουν ολοκληρωθεί. Η λειτουργία είναι επίσης υλοποιημένη με τον ίδιο τρόπο που έχει υλοποιηθεί στο PatientServlet.

Εκεί που διαφέρουν τα 2 servlet είναι ότι ο γιατρός έχει τη δυνατότητα να προγραμματίζει τις ημέρες που δέχεται τα ραντεβού με τον εξής τρόπο:

1. Αρχικά επιλέγει την λειτουργία Plan Appointments. Η λειτουργία αυτή θα τον στείλει στην σελίδα appointmentplans.html στην οποία θα πρέπει να επιλέξει την ημέρα του μήνα για την οποία θέλει να προγραμματίσει ραντεβού αλλά και τις ώρες έναρξης και λήξης της βάρδιας του.
2. Στη συνέχεια πατώντας το κουμπί Plan Appointments στέλνει τα δεδομένα στο DoctorServlet το οποίο με τη σειρά του λαμβάνει την ημέρα και τις ώρες έναρξης και λήξης και τα εισάγει στη βάση δεδομένων με απόσταση μίας ώρας μεταξύ του κάθε ραντεβού. Αυτό σημαίνει ότι αν ο γιατρός επιλέξει να εργαστεί από τις 8 το πρωί μέχρι τις 2 το μεσημέρι τότε θα έχει διαθέσιμα ραντεβού κάθε ώρα από τις 8 μέχρι τις 2.
3. Μετά την εκτέλεση του query και την εισαγωγή των ραντεβού, η εφαρμογή δίνει την δυνατότητα στο γιατρό να ορίσει τις ώρες εργασίας του και για όποια άλλη ημερομηνία επιθυμεί, μέχρις ότου να πατήσει το κουμπί Go Back που θα τον επαναφέρει στο μενού επιλογών του.

Η διαδικασία αφαίρεσης των ραντεβού επίσης μοιάζει πολύ, εμφανίζεται στον γιατρό ένας πίνακας με την ημερομηνία, το όνομα και το επίθετο του ασθενή με τον οποίο έχει ραντεβού και μπορεί επιλέγοντας το συγκεκριμένο ραντεβού να το ακυρώσει. Να σημειωθεί ότι στην περίπτωση του γιατρού, η ακύρωση ενός ραντεβού σημαίνει διαγραφή του ραντεβού από τον πίνακα appointments και όχι απλά διαγραφή του ονόματος του από το αντίστοιχο ραντεβού, όπως γίνεται με την περίπτωση του ασθενή.

Τελευταία λειτουργία του γιατρού είναι επίσης η λειτουργία Logout η οποία λειτουργεί όπως και στον ασθενή.

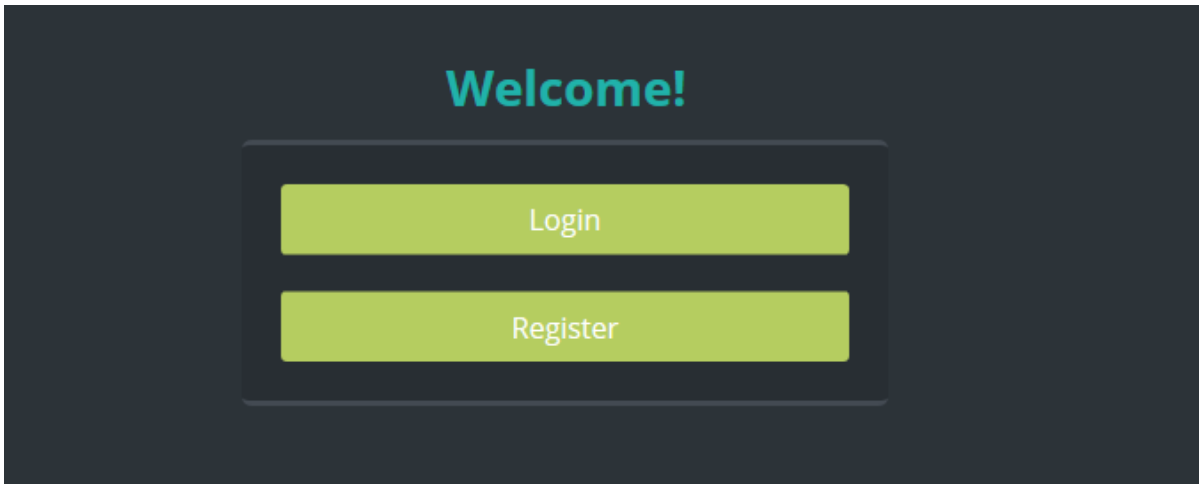
2.7 Κώδικας του servlet AdminServlet

Τελευταίο servlet είναι αυτό του Admin το οποίο περιέχει πιο εύκολα υλοποιήσιμες λειτουργίες.

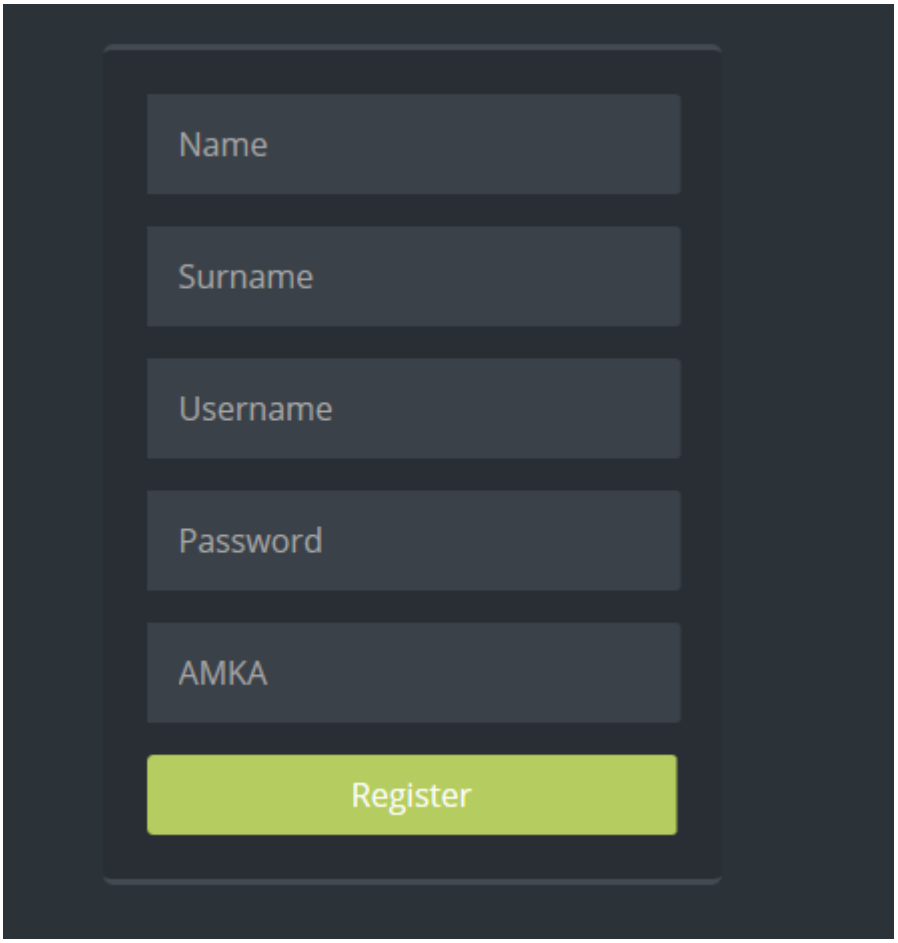
Οι λειτουργίες αυτές ήταν η εγγραφή χρήστη, η εγγραφή γιατρού και η διαγραφή γιατρού.

Όλες οι λειτουργίες υλοποιήθηκαν και λειτουργούν με τον ίδιο τρόπο, καλείται αρχικά από το servlet DynamicPages η αντίστοιχη μέθοδος δημιουργίας της δυναμικής σελίδας με τη φόρμα εγγραφής ή διαγραφής και από εκεί τα στοιχεία στέλνονται είτε στο servlet AddUser είτε στο RemoveUser αντίστοιχα.

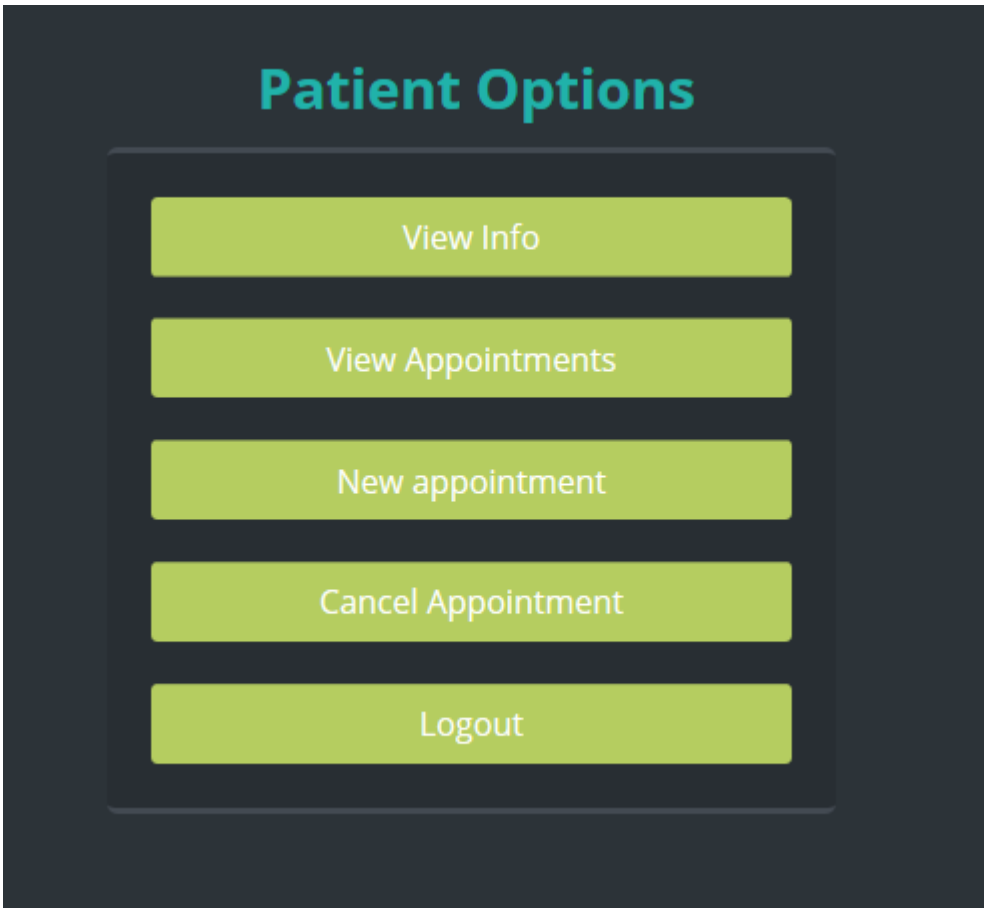
3 Screenshots από την εκτέλεση της εφαρμογής



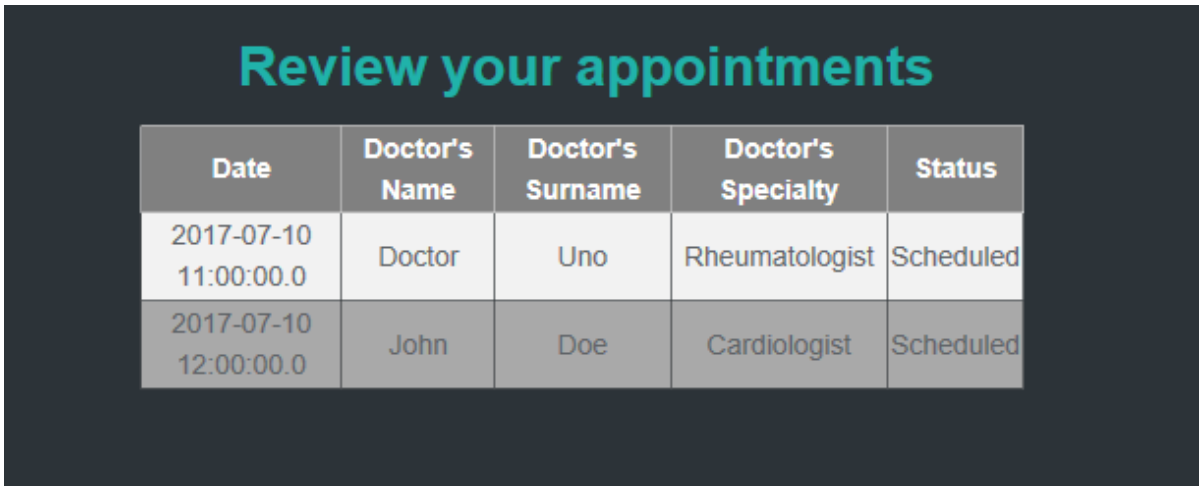
7 index.html ή αλλίως, αρχή της εφαρμογής



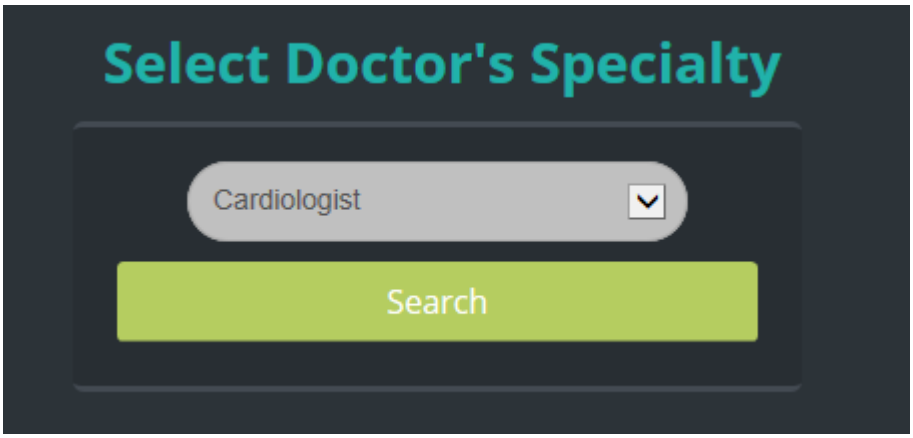
8 Εγγραφή νέου χρήστη



9 Λειτουργίες του ασθενή

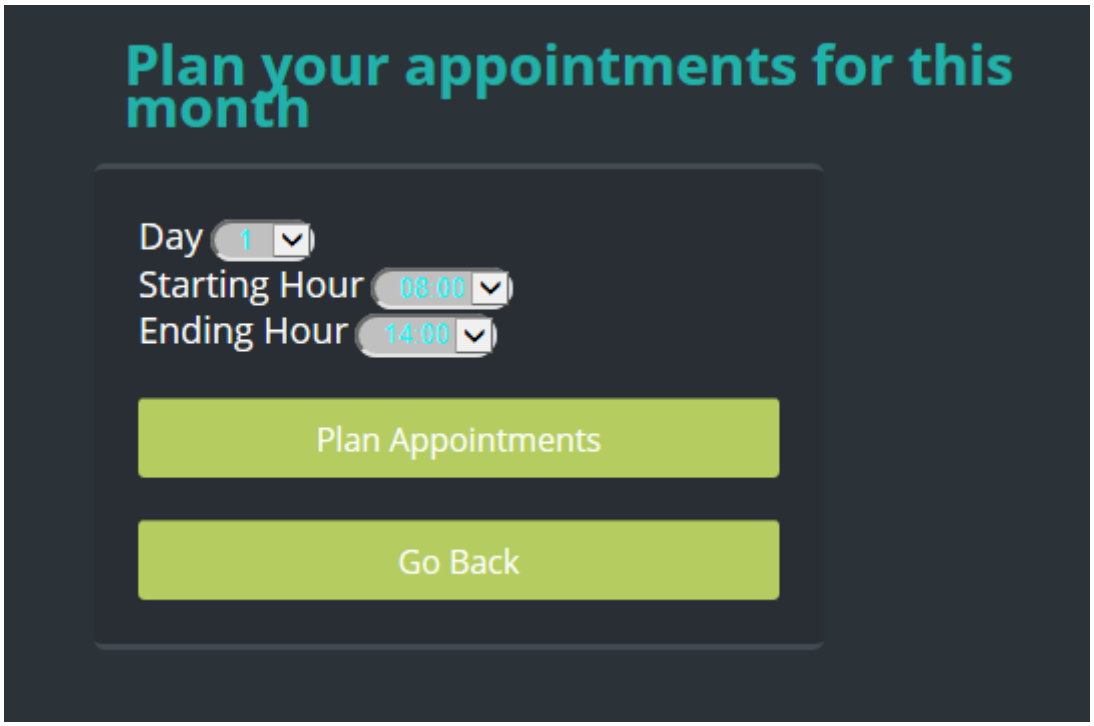


10 Προγραμματισμένα ραντεβού

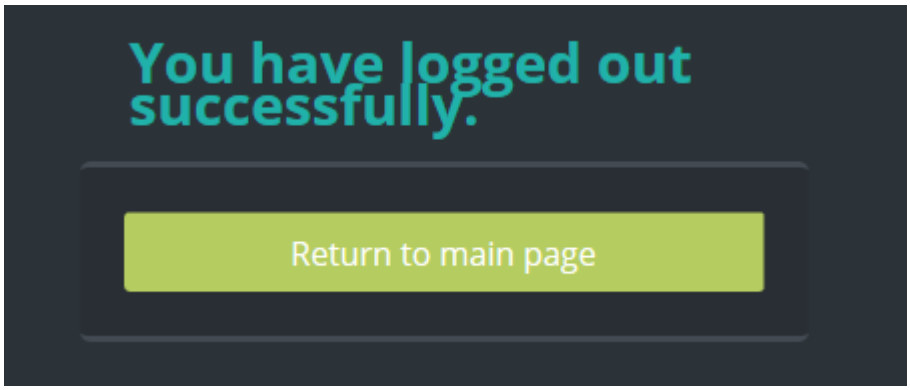


Date	Doctor's Name	Doctor's Surname
2017-07-05 08:00:00	Doctor	Uno
2017-07-05 09:00:00	Doctor	Uno
2017-07-05 10:00:00	Doctor	Uno
2017-07-05 11:00:00	Doctor	Uno
2017-07-05 12:00:00	Doctor	Uno
2017-07-05 13:00:00	Doctor	Uno
2017-07-05 14:00:00	Doctor	Uno
2017-07-10 08:00:00	Doctor	Uno
2017-07-10 09:00:00	Doctor	Uno
2017-07-10 10:00:00	Doctor	Uno
2017-07-10 11:00:00	Mark	Jacobs
2017-07-10 12:00:00	Mark	Jacobs
2017-07-10 12:00:00	Doctor	Uno
2017-07-10 13:00:00	Doctor	Uno
2017-07-10 13:00:00	Mark	Jacobs

Μπορούμε να δούμε ξεκάθαρα ότι υπάρχει και η επιλογή όχι μόνο ημερομηνίας, αλλά και γιατρού, όταν αυτοί δέχονται ραντεβού ίδιες μέρες και ώρες



11 Προγραμματισμός ραντεβού από γιατρό



12 Logging out...

4 Περιεχόμενα παραδοτέου αρχείου

Το συμπιεσμένο αρχείο που παραδίδεται περιέχει το project της εφαρμογής, ένα αρχείο context.xml με τις απαραίτητες αλλαγές που πρέπει να γίνουν στον server προκειμένου να λειτουργήσει η εφαρμογή, καθώς και το αρχείο sql που περιέχει όλες τις πληροφορίες για την δημιουργία της βάσης δεδομένων που χρησιμοποιήθηκε (περιλαμβάνει τους πίνακες που δημιουργήθηκαν, τα constraints που λήφθηκαν υπόψιν και αρκετές άλλες πληροφορίες που προκύπτουν αυτόματα κατά τη διαδικασία backup της PostgreSQL).

Τέλος περιλαμβάνεται η αναλυτική περιγραφή της εφαρμογής σε αρχείο pdf.

5 Βιβλιογραφικές Πηγές

1. Εργαστηριακά παραδείγματα του μαθήματος από το Gunet2
2. Stackoverflow.com για διάφορες απορίες σχετικά με τον κώδικα
3. Google.com για διάφορες απορίες σχετικά με την υλοποίηση της εργασίας αλλά και του κώδικα.