ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος «Προγραμματισμός στο διαδίκτυο και στον παγκόσμιο ιστό»

ΤΕΛΙΚΗ ΕΡΓΑΣΙΑ	***Τίτλος άσκησης***
Όνομα φοιτητή – Αρ. Μητρώου (όλων σε περίπτωση ομαδικής εργασίας)	ΗΛΙΑΣ ΚΩΣΤΟΡΡΙΖΟΣ Π20111
	ΑΠΟΣΤΟΛΟΣ ΤΕΡΤΙΓΚΑΣ Π20186
	ΜΙΧΑΗΛ ΧΡΗΣΤΟΥ Π20205
	ΑΘΑΝΑΣΙΟΣ ΠΑΠΠΑΣ Π19212
Ημερομηνία παράδοσης	7 ΙΟΥΛΙΟΥ 2022

Εκφώνηση της άσκησης

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. Ολοκλήρωση λειτουργιών όλων των κατηγοριών χρηστών (servlet)

2.1. Επεκτείνοντας την προηγούμενη άσκηση, θα υλοποιείστε μηχανισμό login (username και password) για όλες τις κατηγορίες χρηστών. 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. Θα υπάρχει ένα κεντρικό μενού σε μία index.html (ή jsp) 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.

2.2.1. Υλοποιήσετε τις διαδικτυακές διεπαφές (html ή jsp σελίδες) που θα χρησιμοποιούν οι χρήστες όλων των κατηγοριών (Καθηγητές, Φοιτητές, Γραμματεία) για να αλληλεπιδρούν με την εφαρμογή και να χρησιμοποιούν τις αντίστοιχες μεθόδους που απαιτούνται.

2.3. 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 (jsp ή html σελίδες και αντίστοιχες φόρμες), θα αναζητούν στη βάση δεδομένων τα στοιχεία που απαιτούνται ότι απαιτείται και θα επιστρέφουν το αποτέλεσμα στον εκάστοτε χρήστη (βλέπε βήμα 4 για το επίπεδο προβολής).

2.3.1. Λειτουργίες Φοιτητών (Student): Οι Φοιτητές θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: προβολή βαθμολογίας ανά μάθημα, προβολή βαθμολογίας ανά εξάμηνο, προβολή συνολικής βαθμολογίας (για όλα τα μαθήματα που έχει εξεταστεί).

2.3.2. Λειτουργίες Καθηγητών (Professor): Οι Καθηγητές θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: προβολή λίστας βαθμολογίας ανά μάθημα (για ήδη βαθμολογημένα μαθήματα), καταχώρηση βαθμολογίας ανά μάθημα (για μη βαθμολογημένα μαθήματα).

2.3.3. Λειτουργίες Γραμματείας (Secretary). Οι Γραμματείς θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: προβολή μαθημάτων (τίτλος μαθήματος, εξάμηνο, υπεύθυνος Καθηγητής), ανάθεση μαθήματος σε Καθηγητή.

2.4. 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 (logout) θα πρέπει να διαγράφεται το session (invalidate session)και επίσης να διαγράφεται η cache μνήμη της εφαρμογής.

3. Ολοκλήρωση επιπέδου Δεδομένων

3.1. Σε συνέχεια από την προηγούμενη άσκηση, 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 στη βάση δεδομένων. Προσθέσετε επιπλέον δοκιμαστικά δεδομένα στη βάση όπου απαιτείται.

4. Ολοκλήρωση επιπέδου προβολής (html, jsp)

4.1. Σε συνέχεια του προηγούμενου βήματος, υλοποιήστε όλες τις απαραίτητες σελίδες που

χρειάζονται για την προβολή/εμφάνιση των αποτελεσμάτων, όπως jsp και html σελίδες. Ενδεικτικά, μπορείτε για κάθε λειτουργία των χρηστών, να δημιουργήσετε μία jsp σελίδα που θα
λαμβάνει τα αποτελέσματα από το αντίστοιχο servlet και θα δημιουργεί δυναμικά τη σελίδα που θα προβάλει το αντίστοιχο αποτέλεσμα.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

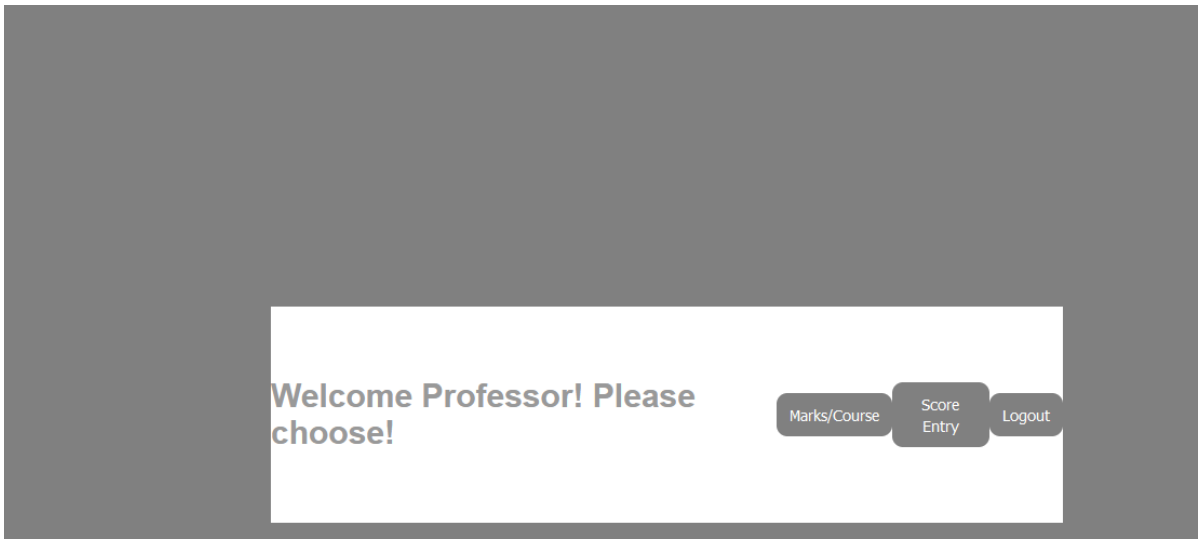
1 Γενική Περιγραφή της λύσης

- Το προγραμματιστικό περιβάλλον που χρησιμοποιήθηκε για την υλοποιήση της εργασίας ήταν το Eclipse.
- Το application server που χρησιμοποιήθηκε στην εργασία ήταν ο Tomcat.
- Η βάση δεδομένων δημιουργήθηκε με την χρήση της PostgreSQL και την χρήση του εργαλείου pgadmin4.

- Στη συγκεκριμένη εργασία , είναι επέκταση της εργασίας 2, όπου ο φοιτητής ,κάνοντας σωστό Login , μπορεί να βλέπει τις βαθμολογίες των μαθημάτων του, προβολή βαθμολογίας ανά εξάμηνο και προβολή συνολικής βαθμολογίας (για όλα τα μαθήματα που έχει εξεταστεί).
- Ο καθηγητής ,θα μπορεί να βλέπει τις βαθμολογίες ανά μάθημα και καταχώρηση ανά μάθημα.
- Η γραμματεία θα μπορούν να έχουν προβολή μαθημάτων και ανάθεση μαθήματος σε καθηγητή.

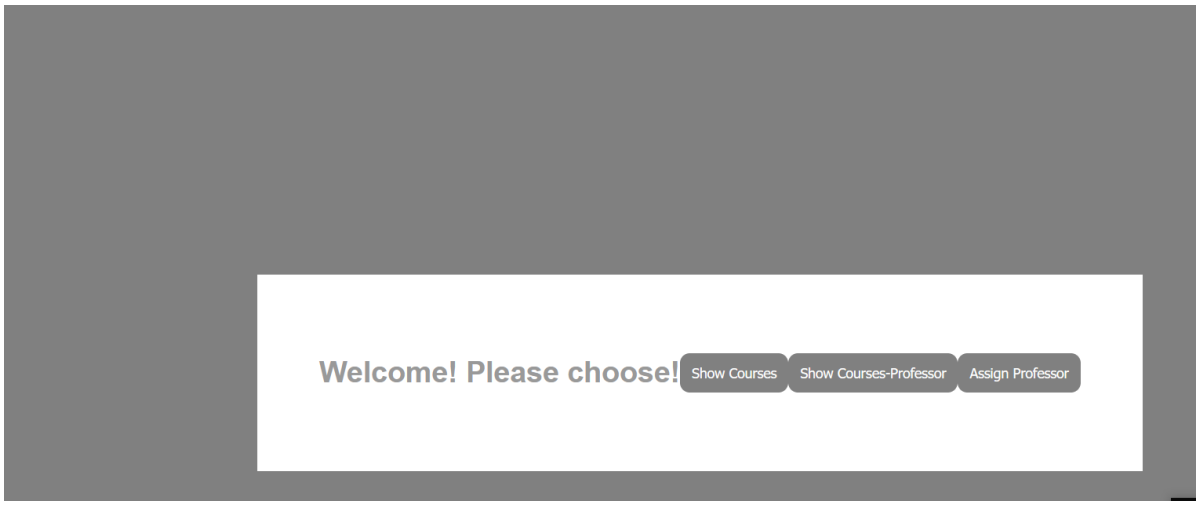
- Δημιουργία και σύνδεση φοιτητή ή καθηγητή

- Μενού καθηγητή



- Μενού φοιτητή

- Μενού γραμματείας



2 Κώδικας προγράμματος

Ακολουθεί η αναλυτική περιγραφή του προγράμματος.

2.1 findCategory.java

```
package com.servlet
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import com.connect.ConnectClass;
@WebServlet("/findCategory")
public class findCategory extends HttpServlet {
        private static final long serialVersionUID = 1;
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
                try {
                        Class.forName("org.postgresql.Driver");
                        Connection con = ConnectClass.dbCon();
                        String username= request.getParameter("username");
                        String password= request.getParameter("password");
                PreparedStatement ps= con.prepareStatement("SELECT * FROM kathigites
WHERE username=? and password=?");
                ps.setString(1, username);
                ps.setString(2, password);
                ResultSet rs=ps.executeQuery();
                boolean status=rs.next();
                if (status)
                {
                RequestDispatcher rd= request.getRequestDispatcher("ProfessorMenu.html");
              rd.forward(request, response);
              System.out.println("Successful entry kathigitis");
HttpSession session = request.getSession();
```

```
                    session.setMaxInactiveInterval(300);
                }
                }
                else
                {
                        PreparedStatement ps2= con.prepareStatement("SELECT * FROM
mathites WHERE username=? and password=?");
                        ps2.setString(1, username);
                        ps2.setString(2, password);
                        ResultSet rs2=ps2.executeQuery();
                        boolean status2=rs2.next();
                                if (status2)
                                {
                RequestDispatcher rd= request.getRequestDispatcher("StudentMenu.html");
                                rd.forward(request, respon
                                System.out.println("Successful entry mathitis");

                        HttpSession session = request.getSession();
                           session.setMaxInactiveInterval(300);
                        }       }
                                else
                                {
                        RequestDispatcher rd= request.getRequestDispatcher("error.html");
                                rd.forward(request, response);

                                System.out.println("Unsuccessful entry,try again");
                                }
                        }
                } catch (ClassNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }
}
```

2.2 LogoutServlet.java

```
package com.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
```

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
@WebServlet("/LogoutServlet")
public class LogoutServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;
  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
            String username= request.getParameter("username");
            String password= request.getParameter("password");
             HttpSession session=request.getSession();
            response.setContentType("index.html");
            PrintWriter out = response.getWriter();
             System.out.print("Have a nice day,"+username);
        session.setAttribute("username",username);
            RequestDispatcher rd= request.getRequestDispatcher("index.html");
     rd.forward(request, response);
     session.removeAttribute("username");
     session.invalidate();
  }
}
```

2.3 ConnectClass.java

```
package com.connect;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class ConnectClass {
  public static Connection dbCon() {
            Connection con=null;
            String url ="jdbc:postgresql://localhost:5432/grammateia";
            String user ="postgres";
            String password="0000
     try {
            con=DriverManager.getConnection(url,user,password);
            if(con!=null) {
                     System.out.println("Connected");
            }else {
                     System.out.println("Failed");
            }
     }catch(SQLException e) {
            System.out.println(e.getMessage());
```

```
    }
    return con;
  }
}
```

2.4 Index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Home</title>
<style>
input[type=text] {
 width: 80%;
  background-color: Gray;
  color: white;
  padding: 10px;
  margin: 5px ;
  border: 6px;
  border-radius: 10px;
}

input[type=password] {
 width: 80%;
  background-color: Gray;
  color: white;
  padding: 10px;
  margin: 5px ;
  border: 6px;
  border-radius: 10px;
}

input[type=submit] {
  background-color: Gray;

}
button[type=submit] {
   background-color: grey;
  border-radius: 8px;
  border-width: 0;
  color: white;
  cursor: pointer;
   font-size: 14px;
  font-weight: 500;
```

```
  line-height: 20px;
  margin: 0;
  padding: 10px 12px;
  text-align: center;
}
button[type=back] {
   background-color: grey;
  border-radius: 10px;
  border-width: 0;
  color: white;
  cursor: pointer;
   font-size: 14px;
  font-weight: 500;
  line-height: 20px;
  margin: 0;
  padding: 10px 12px;
  text-align: center;
}
.center{

display: flex;
  justify-content: center;
  align-items: center;
  height: 500px;
  border: 100px solid grey;}
table{
  border-radius: 10px;
  border-width: 0;
display: flex;
font-size: 14px;
  font-weight: 50;
  line-height: 2px;
  justify-content: center;
  align-items: center;
  height:100px;}
  button{
    background-color: grey;
  border-radius: 8px;
  border-width: 10;
  color: white;
  cursor: pointer;
   font-size: 10px;
  font-weight: 50;
  line-height: 10px;
  margin: 0;
```

```
  padding: 1px 12px;
  text-align: down;}
</style>
</head>
<body>
<div class="center">
      <table id="table1">
 <tr>
   <th> <button onclick="window.location.href = 'CreateUsers.html';"> <h3> Register as Professor </h3></button></th>
<th><button onclick="window.location.href = 'CreateStud.html';"> <h3> Register as Student </h3></button></th>
    </tr>
  </table>
 <br><br>
<form action='findCategory' method='post' >
  <label for="username">Username:</label><br>
  <input type="text" id="username" name="username" value=""><br>
  <label for="password">Password:</label><br>
  <input type="password" id="password" name="password" value=""><br><br>
  <button type="submit"> Login</button>
    <button type="back" formaction="Homepage.html">Back</button>
  <br><br><br>
</div>
</form>
</body>
```

2.5 Grammateia.sql

```
SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;
CREATE EXTENSION IF NOT EXISTS plpgsql WITH SCHEMA pg_catalog;
COMMENT ON EXTENSION plpgsql IS 'PL/pgSQL procedural language';
SET default_tablespace = '';
SET default_with_oids = false;
```

```
CREATE TABLE public.kathigites (
    id integer NOT NULL,
    name character varying(20),
    surname character varying(20),
    username character varying(20),
    password character varying(30),
    course character varying(20)
);

ALTER TABLE public.kathigites OWNER TO postgres;


ALTER TABLE public.kathigites ALTER COLUMN id ADD GENERATED ALWAYS AS IDENTITY (
    SEQUENCE NAME public.kathigites_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1
);


CREATE TABLE public.mathimata (
    id integer NOT NULL,
    name character varying(20),
    id_kathigiti integer NOT NULL,
    semester character varying(20)
);


ALTER TABLE public.mathimata OWNER TO postgres;

--
-- TOC entry 199 (class 1259 OID 16736)
-- Name: mathites; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.mathites (
    id integer NOT NULL,
    name character varying(20),
    surname character varying(20),
    username character varying(20),
    password character varying(30),
    semester integer
);
```

```
ALTER TABLE public.mathites OWNER TO postgres;


ALTER TABLE public.mathites ALTER COLUMN id ADD GENERATED ALWAYS AS IDENTITY (
    SEQUENCE NAME public.mathites_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1
);



CREATE TABLE public.vathmoi (
    id_mathiti integer NOT NULL,
    id_mathimatos integer NOT NULL,
    vathmos integer
);


ALTER TABLE public.vathmoi OWNER TO postgres;

-
ALTER TABLE ONLY public.kathigites
    ADD CONSTRAINT kathigites_pkey PRIMARY KEY (id);



ALTER TABLE ONLY public.mathimata
    ADD CONSTRAINT mathimata_pkey PRIMARY KEY (id);


ALTER TABLE ONLY public.mathites
    ADD CONSTRAINT mathites_pkey PRIMARY KEY (id);


ALTER TABLE ONLY public.vathmoi
    ADD CONSTRAINT vathmoi_pkey PRIMARY KEY (id_mathiti, id_mathimatos);



ALTER TABLE ONLY public.mathimata
```

```
    ADD CONSTRAINT mathimata_id_kathigiti_fkey FOREIGN KEY (id_kathigiti) REFERENCES
public.kathigites(id) ON DELETE CASCADE;
ALTER TABLE ONLY public.vathmoi
    ADD CONSTRAINT vathmoi_id_mathiti_fkey FOREIGN KEY (id_mathiti) REFERENCES
public.mathites(id);
```

3 Βιβλιογραφικές Πηγές

https://www.codejava.net/java-ee/servlet/how-to-set-session-timeout-for-java-web-application
Βιβλίο: Συστήματα Βάσεων Δεδομένων,7η έκδοση, Εκδόσεις Γκιουρδας, Abraham Silberschatz.