Εικόνα που περιέχει σχεδίαση

Περιγραφή που δημιουργήθηκε αυτόματα

**ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ**

**Τμήμα Πληροφορικής**

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ**

**ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΤΟ ΔΙΑΔΙΚΤΥΟ ΚΑΙ ΤΟΝ ΠΑΓΚΟΣΜΙΟ ΙΣΤΟ**

**ΤΕΛΙΚΗ ΕΡΓΑΣΙΑ - 2020**

**ΔΗΜΗΤΡΙΟΣ ΜΟΣΧΟΥ – Π18209  
ΑΠΟΣΤΟΛΟΣ ΣΙΑΤΡΑΣ – Π18215**

Πίνακας περιεχομένων

ΕΚΦΩΝΗΣΗ ΑΣΚΗΣΗΣ3

ΓΕΝΙΚΗ ΛΥΣΗ4

ΤΕΧΝΙΚΕΣ ΛΕΠΤΟΜΕΡΕΙΕΣ6

**ΤΕΛΙΚΗ ΥΛΟΠΟΙΗΣΗ7**

ΚΩΔΙΚΑΣ ΚΛΑΣΕΩΝ-SERVLETS8

ΒΙΒΛΙΟΓΡΑΦΙΑ-ΠΗΓΕΣ47

**1.ΕΚΦΩΝΗΣΗ ΑΣΚΗΣΗΣ**

Στην τελική εργασία του μαθήματος επεκτείνονται οι προηγούμενες ασκήσεις ώστε να δημιουργηθεί μία εφαρμογή τριών επιπέδων (3-tier), η οποία θα υλοποιεί τις λειτουργίες (μεθόδους) που ορίστηκαν στις προηγούμενες ασκήσεις.

Στόχοι εργασίας: Ολοκλήρωση λειτουργικότητας 3-tier εφαρμογής, ολοκλήρωση server-side τεχνολογιών (servlets, jsp, βάση δεδομένων), επικοινωνία με βάση δεδομένων, ολοκλήρωση λειτουργιών.

**2. ΓΕΝΙΚΗ ΛΥΣΗ**

***2.1) Ολοκλήρωση λειτουργιών όλων των κατηγοριών χρηστών (servlet)***

2.1.1 Λειτουργίες που αφορούν όλες τις κατηγορίες χρηστών:

* Σύνδεση (login): Κάθε χρήστης συνδέεται με το μοναδικό username-password. Το password διατηρείται στη βάση salted+hashed.
* Παρακολούθηση συνόδου (session management): Εάν η σύνδεση είναι επιτυχής, η εφαρμογή διατηρεί πληροφορία που αφορά τη σύνοδο του συγκεκριμένου χρήστη , από τη στιγμή της σύνδεσης μέχρι την αποσύνδεση του χρήστη. Η πληροφορία συνόδου “ακολουθεί” τον χρήστη κατά την πλοήγησή του στην εφαρμογή, μέχρι την αποσύνδεσή του.
* Αποσύνδεση (logout): Κάθε χρήστης αποσυνδέεται με ασφάλεια από την εφαρμογή. Κατά την αποσύνδεση υλοποιείται ο καθαρισμό της cache και την ακύρωση του session (invalidate session).

2.1.2 Υλοποίηση λειτουργιών ανά κατηγορία χρήστη. Υπάρχει ένα κεντρικό μενού σε μία index.html σελίδα, οποία είναι η αρχική σελίδα για όλους τους χρήστες. Μετά τo login προβάλλεται το μενού λειτουργιών κάθε χρήστη ανάλογα με την κατηγορία στην οποία ανήκει.

* Λειτουργίες Πωλητών (Seller): Οι Πωλητές μπορούν να εκτελούν κατ’ ελάχιστο τις λειτουργίες: καταχώρηση νέου πελάτη, έκδοση λογαριασμού πελατών, αλλαγή προγράμματος πελάτη.
* Λειτουργίες Πελατών (Client): Οι Πελάτες μπορούν να εκτελούν κατ’ ελάχιστο τις λειτουργίες: μέθοδοι όπως προβολή λογαριασμού, προβολή ιστορικού κλήσεων, εξόφληση λογαριασμού.
* Λειτουργίες Διαχειριστή (Administrator). Οι Διαχειριστές μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: δημιουργία νέου πωλητή, δημιουργία νέων προγραμμάτων, αλλαγή χαρακτηριστικών προγράμματος.

***2.2) Ολοκλήρωση επιπέδου Δεδομένων***

Έχουν προστεθεί επιπλέον δοκιμαστικά δεδομένα στη βάση όπου απαιτείται.

***2.3) Ολοκλήρωση επιπέδου προβολής (html, jsp)***

2.3.1 Σε συνέχεια του προηγούμενου βήματος, υλοποιούνται όλες οι απαραίτητες σελίδες που χρειάζονται για την προβολή/εμφάνιση των αποτελεσμάτων, όπως jsp και html σελίδες. Ενδεικτικά, για κάθε λειτουργία των χρηστών, να δημιουργείται μία jsp σελίδα που θα λαμβάνει τα αποτελέσματα από το αντίστοιχο servlet και θα δημιουργεί δυναμικά τη σελίδα που θα προβάλει το αντίστοιχο αποτέλεσμα.

**3. ΤΕΧΝΙΚΕΣ ΛΕΠΤΟΜΕΡΕΙΕΣ**

Η εργασία υλοποιήθηκε με βάση την 3-tier architecture, με χρήση του προγράμματος JetBrains IntelliJ IDEA.

Παρακάτω παρατίθενται τεχνικές λεπτομέρειες ανά επίπεδο.

***3.1) Επίπεδο 1***

Χρησιμοποιήθηκαν οι φυλλομετρητές Chrome (Έκδοση 83.0.4103.106-64bit), Safari (Έκδοση 13.1.1 (15609.2.9.1.2)).

***3.2) Επίπεδο 2***

Χρησιμοποιήθηκε ο εξυπηρετητής εφαρμογής (application server) Apache Tomcat (Έκδοση 9.0.35 και έκδοση 8.5.55).

ΣΗΜΕΙΩΣΗ: Κατά την δημιουργία Datasource στο αρχείο **context.xml** χρησιμοποιήθηκε το παρακάτω URL (λόγω προβλήματος με την ζώνη ώρας του εξυπηρετητή της MySQL)

url="***jdbc:mysql://localhost:3306/mydb?useSSL=false&amp;useUnicode=true&amp;serverTimezone=EET***"

***3.3) Επίπεδο 3***

Χρησιμοποιήθηκε το Σύστημα Διαχείρισης Βάσεων Δεδομένων MySQL Workbench.

**4.ΤΕΛΙΚΗ ΥΛΟΠΟΙΗΣΗ**

***4.1) Λειτουργίες Πωλητών (Seller):***

Οι Sellers μπορούν να καταχωρήσουν νέους πελάτες μέσω, της μεθόδου addClient() και του servlet SellerServlet, να εμφανίσουν τα διαθέσιμα προγράμματα κινητής τηλεφωνίας μέσω της μεθόδου showPrograms() και του servlet SellerServlet, να εκδώσουν λογαριασμό κινητής τηλεφωνίας για τους πελάτες μέσω της μεθόδου billIssuance(), του Servlet BillIssuanceServlet. Τέλος, έχουν τη δυνατότητα αλλαγής προγράμματος σε πελάτη, την επεξεργασία και διαγραφή ενός πελάτη μέσω των μεθόδων editClient(), deleteFromClients(), deleteFromNumbers(), deleteFromCalls(), deleteFromBills() και του servlet EditClientServlet, Για να συσχετιστεί πελάτης με πρόγραμμα χρησιμοποιείται το servlet Cust\_programServlet

και οι μέθοδοι showClientPartially(), showPrograms(), clientProgram().

***4.2) Λειτουργίες Πελατών (Client):***

Οι πελάτες μπορούν να βλέπουν τον λογαριασμό τους με την μέθοδο viewBill() και το servlet CallServlet. Επίσης έχει την δυνατότητα να βλέπουν το ιστορικό των κλήσεων τους μέσω της μεθόδου viewCallHistory() και το servlet CallServlet. Ακόμη, βλέπουν τις προσωπικές τους πληροφορίες, μέσω της μεθόδου viewPersonalData() και το servlet ClientServlet. Τέλος μπορεί να εξοφλήσει τον λογαριασμό του με την μέθοδο paid() και το servlet BillServlet.

***4.3) Λειτουργίες Διαχειριστών (Admin):***

Οι διαχειριστές έχουν τη δυνατότητα να εισάγουν νέους πωλητές, να τους επεξεργαστούν και να τους διαγράψουν μέσω των μεθόδων addSeller(), editSeller(), deleteSeller() και των servlet AdminServlet, EditSellerServlet. Επίσης, μπορούν να προσθέσουν, επεξεργαστούν και να διαγράψουν προγράμματα κινητής τηλεφωνίας, μέσω των μεθόδων addProgram(), editProgram(), updateNumbersDelete(), deleteProgram(), όπως επίσης και μέσω των servlet NewProgramServlet, EditProgramServlet.

**5.ΚΩΔΙΚΑΣ ΚΛΑΣΕΩΝ-SERVLETS**

***5.1) Πακέτα Άσκησης (Packages)***

Δημιουργήθηκαν δύο ξεχωριστά πακέτα τα οποία περιλαμβάνουν τις κλάσεις (netprog.classes) και τα servlets (netprog.servlets).

***5.2) Πακέτο netprog.classes***

Σε σχέση με την προηγούμενη άσκηση, υπήρξαν αλλαγές στον κώδικα των κλάσεων Seller, Client, Admin οι οποίες βρίσκονται στο υπο-πακέτο users. Παρακάτω παρατίθεται η νέα έκδοση των κλάσεων αυτών:

1. **package** netprog.classes.users;
3. **import** java.util.Scanner;
5. **import** netprog.classes.phone\_related.Bill;
6. **import** netprog.classes.phone\_related.PhoneNumber;

9. **public** **class** Seller **extends** Users {
11. /\*\*
12. \* Constructor
13. \*/
14. **public** Seller(){
15. **super**("Seller");
16. }
18. /\*\*
19. \* Function used as a query in Prepared Statement
20. \* @return the SQL query which adds a new Client in Database
21. \*/
22. **public** String addClient() {
23. **return** "INSERT INTO clients VALUES (?,?,?,?,?,?)";
24. }
26. /\*\*
27. \* Function used as a query in Prepared Statement
28. \* @return the SQL query which associates a client's number with a telecommunications program
29. \*/
30. **public** String clientProgram(){
31. **return** "INSERT INTO numbers VALUES (?, ?)";
32. }
34. /\*\*
35. \* Function used as a query in Prepared Statement
36. \* @param phone used in the query to specify WHERE
37. \* @return the SQL query which shows some information about the Client
38. \*/
39. **public** String showClientPartially(String phone){
40. **return** "SELECT username, name, surname FROM clients WHERE phone\_number = " + phone + ";";
41. }
43. /\*\*
44. \* Function used as a query in Prepared Statement
45. \* @return the SQL query which shows all telecommunications programs
46. \*/
47. **public** String showPrograms(){
48. **return** "SELECT \* FROM programs;";
49. }
51. /\*\*
52. \* Edits the attributes of the specified Client object
53. \* @param c Client object
54. \* @param name Client's Name
55. \* @param surname Client's Surname
56. \* @param username Client's username
57. \*/
59. **public** **final** **void** editClient(Client c, String name, String surname, String username){
60. c.name= name;
61. c.surname = surname;
62. c.username = username;
63. System.out.println("\nEdit of " + c.username + " has been completed!");

66. }
68. /\*\*
69. \* Deletes a Clients
70. \* @param c Client object
71. \*/
73. **public** **final** **void** deleteClient(Client c){
74. // Code should be comleted on servlet
75. System.out.println("\nClient " + c.username + " has been deleted!");
77. }
79. /\*\*
80. \* When this method is executed, a value needs to be inserted through console, which indicates the total amount of phone calls the client has made and a new Bill object is being created
81. \* @param number PhoneNumber object (associated with this Bill)
82. \* @param month The month of the requested bill
83. \* @return Bill object
84. \* @throws NumberFormatException Thrown when the insertetd value is not a number
85. \* @throws ArithmeticException Thrown when the inserted value is a negative integer
86. \*/
88. **public** **final** Bill billIssuance(PhoneNumber number, String month) **throws** NumberFormatException {
89. Scanner sc = **new** Scanner(System.in);
90. System.out.print("\nPlease insert the total number of your calls: ");
91. **int** calls\_number;
92. **while** (**true**) {
93. **try** {
94. calls\_number = Integer.parseInt(sc.nextLine());
95. **if**(calls\_number < 0)
96. **throw** **new** ArithmeticException();
97. }
98. **catch** (NumberFormatException e) {
99. System.out.println("\nPlease insert an integer: ");
100. **continue**;
101. }
102. **catch**(ArithmeticException e) {
103. System.out.println("Please insert positive integer");
104. **continue**;
105. }
106. **break**;
108. }
109. Bill b = **new** Bill(number, calls\_number, month);
110. System.out.println("\nBill for month " + b.month + " has issued!");
111. **return** b;
113. }
115. /\*\*
116. \* Just a message showing that a Seller has been created
117. \*/
118. @Deprecated
119. **public** **void** Register() {
120. System.out.println("\nSeller " + **this**.username + " has been created! ");
122. }
124. /\*\*
125. \* Just a message showing that a Seller has Logged in
126. \*/
128. @Deprecated
129. **public** **void** Login() {
130. System.out.println("\nSeller " +  **this**.username + " has successfully logged in ");
132. }
134. /\*\*
135. \* Just a message showing that a Seller has Logged out
136. \*/
138. @Deprecated
139. **public** **void** Logout() {
140. System.out.println("\nSeller " +  **this**.username + " has successfully logged out ");

143. }
144. }
145. **package** netprog.classes.users;
147. **public** **class** Admin **extends** Users{

150. /\*\*
151. \* Constructor
152. \*/
154. **public** Admin(){
155. **super**("Admin");
156. }
158. /\*\*
159. \* Creates a new Seller object and then return it
160. \* @return Seller object
161. \*/
162. **public** **final** String addSeller(){ **return** "INSERT INTO sellers VALUES (?,?,?,?)";}
164. /\*\*
165. \* Edits the attributes of Seller object
166. \*/
168. **public** **final** String editSeller() {
169. **return** "UPDATE sellers SET username = ?, name = ?, surname = ? WHERE username = ?;";
170. }
172. /\*\*
173. \* Deletes the specified Seller
174. \*/
176. **public**  **final** String deleteSeller(){ **return** "DELETE FROM sellers WHERE username = ?;";}
178. /\*\*
179. \* Creates a new Program object and then returns it
180. \* @return Program object
181. \*/
182. **public** **final** String addPrograms(){ **return** "INSERT INTO programs VALUES (?,?,?,?,?,?)";}

185. /\*\*
186. \* Edits the attributes of a specified Program object
187. \*/
189. **public** **final** String editPrograms(){
190. **return** "UPDATE programs SET program\_name = ?, fixed\_charge = ?, minutes\_charge = ?, messages\_charge = ?, minutes = ?, messages = ? WHERE program\_name = ?;";
191. }
193. /\*\*
194. \* Deletes a specified Program
195. \*/
197. **public** **final** String deletePrograms(){
198. **return** "DELETE FROM programs WHERE program\_name = ?;";
199. }
201. **public** **final** String updateNumbersDelete(){
202. **return** "UPDATE numbers SET program\_name = 'Basic500' WHERE program\_name = ?;";
203. }
205. **public** **final** String showProgramInfo(){
206. **return** "SELECT \* FROM programs WHERE program\_name = ?;";
207. }
209. **public** **final** String showSellerInfo(){
210. **return** "SELECT \* FROM sellers WHERE username = ?;";
211. }
213. /\*\*
214. \* Just a message showing that an Admin has been created
215. \*/
216. @Deprecated
217. **public** **void** Register() {
218. System.out.println("\nAdmin " + **this**.username + " has been created! ");
220. }
222. /\*\*
223. \* Just a message showing that an Admin has Logged in
224. \*/
225. @Deprecated
226. **public** **void** Login() {
227. System.out.println("\nAdmin " +  **this**.username + " has successfully logged in ");
229. }
231. /\*\*
232. \* Just a message showing that an Admin has Logged out
233. \*/
234. @Deprecated
235. **public** **void** Logout() {
236. System.out.println("\nAdmin " +  **this**.username + " has successfully logged out ");

239. }
240. }
241. **package** netprog.classes.users;

244. **import** java.sql.Timestamp;
245. **import** java.time.YearMonth;
247. **public** **class** Client **extends** Users {
248. **public** String number;
249. **protected** **final** String tin;
251. /\*\*
252. \* Constructor
253. \* @param attr\_tin Personal TIN
254. \* @param attr\_number PhoneNumber object associated with this CLient object
255. \*/
256. **public** Client(String attr\_tin, String attr\_number){
257. **super**("Client");
258. tin = attr\_tin; //ΑΦΜ
259. number = attr\_number;
260. }
262. /\*\*
263. \* Method that pays out a bill
264. \*/
265. **public** String payBill(){
266. **return** "UPDATE bills SET paid = 1 WHERE number = ? AND date\_issued LIKE ? ;";
267. }
269. /\*\*
270. \* Method which shows the client's call history
271. \*/
272. **public** String viewCallHistory(){
273. **return** "SELECT \* FROM calls WHERE caller = ? AND time LIKE ?";
274. }
276. /\*\*
277. \* Method that shows the bill for requested number and month
278. \*/
279. **public** String viewBill(){
280. **return** "SELECT \* FROM bills WHERE number = ? AND date\_issued LIKE ?";
282. }
284. /\*\*
285. \* Shows the full personal data of each client
286. \*/
287. **public** String viewPersonalData(){
288. **return** "SELECT \* FROM clients WHERE username = ?;";
289. }
291. /\*\*
292. \* Just a message showing that a Client has been created
293. \*/
294. @Deprecated
295. **public** **void** Register() {
296. System.out.println("\nClient " + **this**.username + " has been created! ");
298. }
300. /\*\*
301. \* Just a message showing that a Client has Logged in
302. \*/
303. @Deprecated
304. **public** **void** Login() {
305. System.out.println("\nClient " +  **this**.username + " has successfully logged in ");
307. }
309. /\*\*
310. \* Just a message showing that a Client has Logged out
311. \*/
312. @Deprecated
313. **public** **void** Logout() {
314. System.out.println("\nClient " +  **this**.username + " has successfully logged out ");

317. }

320. }

***5.3) Πακέτο netprog.servlets***

Στο πακέτο αυτό περιλαμβάνονται τα απαραίτητα servlets και κλάσεις που χρησιμοποιούνται από τα servlets για την άσκηση. Παρακάτω παρατίθεται ο κώδικας των servlets και των κλάσεων:

5.3.1 Κώδικας κλάσης Encryption:

1. **package** netprog.servlets;
3. **import** java.math.BigInteger;
4. **import** java.security.MessageDigest;
5. **import** java.security.NoSuchAlgorithmException;
7. /\*\*
8. \* May 3, 2014, 4:25:00 AM
9. \*
10. \* @author syggouroglou@gmail.com
11. \*/
12. **public** **class** Encryption {
14. /\*\*
15. \* Computes the hash of the given string. Return the hash uppercase.
16. \*
17. \* @param unhashed
18. \* @return Return the hash of the given string uppercase.
19. \*/
20. **public** **static** String getHashMD5(String unhashed) {
21. **return** getHashMD5(unhashed, "uyfhvu43");
22. }
24. /\*\*
25. \* Computes the hash of the given string salted. Return the hash uppercase.
26. \* @param unhashed the string to hash
27. \* @param salt the salt to use
28. \* @return the hash of the given string salted and uppercase.
29. \* @throws NoSuchAlgorithmException
30. \*/
31. **public** **static** String getHashMD5(String unhashed, String salt) {
32. // Hash the password.
33. **final** String toHash = salt + unhashed + salt;
34. MessageDigest messageDigest = **null**;
35. **try** {
36. messageDigest = MessageDigest.getInstance("MD5");
37. } **catch** (NoSuchAlgorithmException ex) {
38. **return** "00000000000000000000000000000000";
39. }
40. messageDigest.update(toHash.getBytes(), 0, toHash.length());
41. String hashed = **new** BigInteger(1, messageDigest.digest()).toString(16);
42. **if** (hashed.length() < 32) {
43. hashed = "0" + hashed;
44. }
45. **return** hashed.toUpperCase();
46. }
48. }

**ΣΗΜΕΙΩΣΗ**: Ο κώδικας της κλάσης αποτελεί μέρος της εργαστηριακής άσκησης 06b-ExamplePost, η οποία βρίσκεται στο GUNet2-eclass.

5.3.2. Κώδικας servlet LoginServlet:

1. **package** netprog.servlets;
3. **import** netprog.classes.users.Admin;
4. **import** netprog.classes.users.Client;
5. **import** netprog.classes.users.Seller;
6. **import** netprog.classes.users.Users;
7. **import** netprog.servlets.Admin.AdminServlet;
8. **import** netprog.servlets.Client.ClientServlet;
9. **import** netprog.servlets.Seller.SellerServlet;
11. **import** javax.naming.InitialContext;
12. **import** javax.servlet.ServletException;
13. **import** javax.servlet.annotation.WebServlet;
14. **import** javax.servlet.http.HttpServlet;
15. **import** javax.servlet.http.HttpServletRequest;
16. **import** javax.servlet.http.HttpServletResponse;
17. **import** javax.servlet.http.HttpSession;
18. **import** javax.sql.DataSource;
19. **import** java.io.IOException;
20. **import** java.io.PrintWriter;
21. **import** java.sql.\*;
23. @WebServlet("/main")
24. **public** **class** LoginServlet **extends** HttpServlet {
25. **private** **static** **final** **long** serialVersionUID = 1L;
26. **public** **static** HttpSession session = **null**;
27. **public** **static** Users user = **null**;
29. **private** DataSource datasource = **null**;
31. /\*\*
32. \* Establishes the Datasource required for the servlet
33. \* @throws ServletException
34. \*/
35. **public** **void** init() **throws** ServletException {
36. **try**{
37. InitialContext ctx = **new** InitialContext();
38. datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/LiveDataSource");
39. }
40. **catch** (Exception e){
41. **throw** **new** ServletException(e.toString());
42. }
43. }
45. /\*\*
46. \* Automatic constructor (created by IntelliJ)
47. \*/
48. **public** LoginServlet(){
49. **super**();
50. }
52. /\*\*
53. \* Function to be executed when submitting a form, using POST method
54. \* @param req
55. \* @param res
56. \* @throws ServletException
57. \* @throws IOException
58. \*/
59. **protected** **void** doPost(HttpServletRequest req, HttpServletResponse res)
60. **throws** ServletException, IOException {
61. res.setContentType("text/html; charset=UTF-8");
62. req.setCharacterEncoding("UTF-8");
63. res.setCharacterEncoding("UTF-8");
65. PrintWriter out = res.getWriter();
66. out.println("<html>");
67. out.println("<head><title>Σύνδεση | Πλατφόρμα πρόσβασης σε ΒΔ</title>" +
68. "<link rel=\"stylesheet\" type=\"text/css\" href=\"style.css\"></head>");
69. out.println("<body>");
71. **try**{
72. Connection con = datasource.getConnection();
73. Statement stmt = con.createStatement();
75. String username = req.getParameter("username");
76. String password = req.getParameter("password");
77. String attribute = req.getParameter("attribute");
79. String hashed = Encryption.getHashMD5(password, "uyfhvu43");
81. ResultSet rs = stmt.executeQuery("SELECT \* FROM " + attribute +
82. " WHERE username='" + username + "' AND password='" + hashed +"';");
84. **boolean** found = **false**;
85. **while** (rs.next()){
86. String name = rs.getString("name");
87. String surname = rs.getString("surname");
88. String URL = "";
90. session = req.getSession(**true**);
91. **switch** (attribute){
92. **case** "sellers":
93. user = (Seller)session.getAttribute("seller");
94. **if** (user == **null**){
95. user = SellerServlet.getSeller(name, surname);
96. session.setAttribute("seller", user);
97. }
98. URL = "/seller.jsp";
99. **break**;
100. **case** "admins":
101. user = (Admin)session.getAttribute("admin");
102. **if** (user == **null**){
103. user = AdminServlet.getAdmin(name, surname);
104. session.setAttribute("admin", user);
105. }
106. URL = "/admin.jsp";
107. **break**;
108. **case** "clients":
109. user = (Client)session.getAttribute("client");
110. **if** (user == **null**){
111. user = ClientServlet.getClient(name, surname, username, rs.getString("phone\_number"));
112. session.setAttribute("client", user);
113. }
114. URL = "/client.jsp";
115. }
116. req.getRequestDispatcher(URL).forward(req,res);
118. found = **true**;
119. }
121. rs.close();
122. con.close();
124. **if** (!found){
125. out.println("<h2 style='text-align: center'>Λανθασμένο Όνομα Χρήστη ή/και Κωδικός Πρόσβασης</h2>");
126. out.println("<br>");
127. out.println("<h3 style='text-align: center'><a href='index.html'>Επιστροφή στην σελίδα σύνδεσης</a></h3>");
128. }
129. }
130. **catch** (Exception e){
131. out.println("Database connection problem");
132. out.println("<br>");
133. out.println(e.getLocalizedMessage());
134. }
135. **finally** {
136. out.println("</body>");
137. out.println("</html>");
138. }
139. }
140. }

5.3.3. Κώδικας servlet LogoutServlet:

1. **package** netprog.servlets;
3. **import** netprog.classes.users.Admin;
4. **import** netprog.classes.users.Client;
5. **import** netprog.classes.users.Seller;
7. **import** javax.naming.InitialContext;
8. **import** javax.servlet.ServletException;
9. **import** javax.servlet.annotation.WebServlet;
10. **import** javax.servlet.http.HttpServlet;
11. **import** javax.servlet.http.HttpServletRequest;
12. **import** javax.servlet.http.HttpServletResponse;
13. **import** javax.servlet.http.HttpSession;
14. **import** javax.sql.DataSource;
15. **import** java.io.IOException;
16. **import** java.io.PrintWriter;
17. **import** java.sql.Connection;
18. **import** java.sql.PreparedStatement;
19. **import** java.sql.ResultSet;
20. **import** java.util.Locale;
22. @WebServlet("/logoutservlet")
23. **public** **class** LogoutServlet **extends** HttpServlet {
24. **private** DataSource datasource = **null**;
25. **public** HttpSession session = **null**;
27. /\*\*
28. \* Establishes the Datasource required for the servlet
29. \* @throws ServletException
30. \*/
31. **public** **void** init() **throws** ServletException{
32. **try** {
34. InitialContext ctx = **new** InitialContext();
35. datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/LiveDataSource");
36. } **catch**(Exception e) {
37. **throw** **new** ServletException(e.toString());
38. }
39. }
41. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
42. response.setContentType("text/html; charset=UTF-8");
43. response.setCharacterEncoding("UTF-8");
44. request.setCharacterEncoding("UTF-8");
45. PrintWriter out = response.getWriter();
46. out.println("<html>");
47. out.println("<head><title>Παράδειγμα Servlet με Βάση</title></head>");
48. out.println("<body>");
50. **try** {
51. session = request.getSession(**false**);
52. Seller value = (Seller)session.getAttribute("seller");
53. Client value2 = (Client)session.getAttribute("client");
54. Admin value3 = (Admin)session.getAttribute("admin");
55. **if** (value != **null** | value2 != **null** | value3 != **null**) {
56. //                response.setHeader("Cache-Control","no-cache");
57. //                response.setDateHeader("Expires", 0);
58. //                response.setHeader("Pragma","no-cache");
59. response.setHeader("Cache-Control","no-cache");
60. response.setDateHeader("Expires", 0);
61. response.setHeader("Pragma","no-cache");
62. session.removeAttribute("seller");
63. session.removeAttribute("client");
64. session.removeAttribute("admin");
65. LoginServlet.session = **null**;
66. session.invalidate();
67. response.sendRedirect("disconnect.html");
69. //LoginServlet.session.invalidate();
70. }
71. **else** {
72. out.println("Your session has expired!");
73. out.println("You will be redirected to Log In Screen...");
74. response.sendRedirect("index.html");
75. }
76. } **catch**(Exception e) {
77. out.println("Database connection problem");
78. out.println("<br>");
79. out.println(e.getLocalizedMessage());
80. }
81. }
82. }

5.3.4. Κώδικας SellerServlet:

1. **package** netprog.servlets.Seller;
3. **import** netprog.classes.users.Admin;
4. **import** netprog.classes.users.Seller;
5. **import** netprog.classes.users.Users;
6. **import** netprog.servlets.Encryption;
7. **import** netprog.servlets.LoginServlet;
9. **import** javax.naming.InitialContext;
10. **import** javax.servlet.ServletException;
11. **import** javax.servlet.annotation.WebServlet;
12. **import** javax.servlet.http.HttpServlet;
13. **import** javax.servlet.http.HttpServletRequest;
14. **import** javax.servlet.http.HttpServletResponse;
15. **import** javax.servlet.http.HttpSession;
16. **import** javax.sql.DataSource;
17. **import** java.io.IOException;
18. **import** java.io.PrintWriter;
19. **import** java.sql.\*;
21. @WebServlet("/sellerservlet")
22. **public** **class** SellerServlet **extends** HttpServlet{
23. **private** **static** **final** **long** serialVersionUID = 1L;
25. **private** **static** Seller seller = **new** Seller();
27. **private** DataSource datasource = **null**;
29. /\*\*
30. \* Establishes the Datasource required for the servlet
31. \* @throws ServletException
32. \*/
33. **public** **void** init() **throws** ServletException{
34. **try** {
36. InitialContext ctx = **new** InitialContext();
37. datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/LiveDataSource");
38. } **catch**(Exception e) {
39. **throw** **new** ServletException(e.toString());
40. }
41. }
43. /\*\*
44. \* Function to be executed when submitting a form, using GET method
45. \* @param request
46. \* @param response
47. \* @throws ServletException
48. \* @throws IOException
49. \*/
50. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
51. response.setContentType("text/html; charset=UTF-8");
52. response.setCharacterEncoding("UTF-8");
53. request.setCharacterEncoding("UTF-8");
54. PrintWriter out = response.getWriter();
55. out.println("<html>");
56. out.println("<head><title>Παράδειγμα Servlet με Βάση</title></head>");
57. out.println("<body>");
59. **if**(LoginServlet.session!=**null**) {
60. **try** {
61. Connection con = datasource.getConnection();
63. PreparedStatement showPrograms = con.prepareStatement(((Seller) LoginServlet.user).showPrograms());
64. ResultSet rs = showPrograms.executeQuery();
65. **int** i = 0;
66. String htmlRow = "";
67. **while** (rs.next()) {
68. i++;
70. String pr\_name = rs.getString("program\_name");
71. **int** fixed\_charge = rs.getInt("fixed\_charge");
72. **double** minutes\_charge = rs.getDouble("minutes\_charge");
73. **double** messages\_charge = rs.getDouble("messages\_charge");
74. **int** minutes = rs.getInt("minutes");
75. **int** messages = rs.getInt("messages");
76. htmlRow = createHTMLRow(pr\_name, fixed\_charge, minutes\_charge, messages\_charge, minutes, messages);
77. request.setAttribute("htmlrow" + i, htmlRow);
79. }
80. request.setAttribute("numberofrows", i);
81. //                request.setAttribute("session\_info", ((Seller)LoginServlet.session.getAttribute("seller")).getName() + " "
82. //                        + ((Seller)LoginServlet.session.getAttribute("seller")).getSurname());
83. request.getRequestDispatcher("/view\_prog.jsp").forward(request, response);
85. showPrograms.close();
86. rs.close();
88. con.close();
89. } **catch** (Exception e) {
90. out.println("Database connection problem");
91. out.println("<br>");
92. out.println(e.getLocalizedMessage());
93. }
94. }
95. **else** {
96. out.println("Session Timeout");
97. }
98. }
100. /\*\*
101. \* This function creates a simple HTML table row, which includes information about telecommunications program
102. \* @param prog\_name Program's name
103. \* @param f\_charge Fixed charge
104. \* @param min\_charge charge per minute (except contract charge)
105. \* @param mes\_charge charge per message (except contract charge)
106. \* @param min free minutes
107. \* @param mes free messages
108. \* @return the complete row
109. \*/
110. **private** String createHTMLRow(String prog\_name, **int** f\_charge, **double** min\_charge, **double** mes\_charge, **int** min, **int** mes) {
111. String row = "<tr>";
112. row  += "<td>" + prog\_name + "</td>";
113. row  += "<td>" + f\_charge + "</td>";
114. row  += "<td>" + min\_charge + "</td>";
115. row  += "<td>" + mes\_charge + "</td>";
116. row  += "<td>" + min + "</td>";
117. row  += "<td>" + mes + "</td>";
118. row +="</tr>";
119. **return** row;
121. }
123. /\*\*
124. \* Function to be executed when submitting a form, using POST method
125. \* @param request
126. \* @param response
127. \* @throws ServletException
128. \* @throws IOException
129. \*/
130. **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException{
131. response.setContentType("text/html; charset=UTF-8");
132. response.setCharacterEncoding("UTF-8");
133. request.setCharacterEncoding("UTF-8");
134. PrintWriter out = response.getWriter();
136. **if**(LoginServlet.session!=**null**) {
137. String username = request.getParameter("new\_user");
138. String password = request.getParameter("new\_pass");
139. String name = request.getParameter("new\_name");
140. String surname = request.getParameter("new\_surname");
141. String tin = request.getParameter("tin");
142. String phone\_number = request.getParameter("phone");
143. **try** {
144. Connection con = datasource.getConnection();
145. Statement stmt = con.createStatement();
147. PreparedStatement preparedStatement = con.prepareStatement(((Seller) LoginServlet.user).addClient());
148. preparedStatement.setString(1, username);
149. preparedStatement.setString(2, Encryption.getHashMD5(password));
150. preparedStatement.setString(3, name);
151. preparedStatement.setString(4, surname);
152. preparedStatement.setString(5, tin);
153. preparedStatement.setString(6, phone\_number);
155. preparedStatement.executeUpdate();
156. request.getRequestDispatcher("/cust\_program.jsp").forward(request, response);
158. preparedStatement.close();
159. con.close();
160. } **catch** (Exception e) {
161. out.println("Database connection problem");
162. //out.println("<br>");
163. //out.println(e.getLocalizedMessage());
164. }
165. }
166. **else**{
167. response.sendRedirect("seller.jsp");
168. }
169. }
171. /\*\*
172. \* Getter method
173. \* @return a static Seller object
174. \*/
175. **public** **static** Seller getSeller(String name, String surname){
176. seller.setName(name);
177. seller.setSurname(surname);
178. **return** seller;
179. }
180. }

5.3.5. Κώδικας servlet ClientServlet:

1. **package** netprog.servlets.Client;
3. **import** netprog.classes.users.Client;
4. **import** netprog.classes.users.Seller;
5. **import** netprog.servlets.Encryption;
6. **import** netprog.servlets.LoginServlet;
8. **import** javax.naming.InitialContext;
9. **import** javax.servlet.ServletException;
10. **import** javax.servlet.annotation.WebServlet;
11. **import** javax.servlet.http.HttpServlet;
12. **import** javax.servlet.http.HttpServletRequest;
13. **import** javax.servlet.http.HttpServletResponse;
14. **import** javax.sql.DataSource;
15. **import** java.io.IOException;
16. **import** java.io.PrintWriter;
17. **import** java.sql.Connection;
18. **import** java.sql.PreparedStatement;
19. **import** java.sql.ResultSet;
21. @WebServlet("/clientservlet")
22. **public** **class** ClientServlet **extends** HttpServlet {
23. **private** **static** **final** **long** serialVersionUID = 1L;
25. **private** **static** Client client = **new** Client("", "");
27. **private** DataSource datasource = **null**;
29. /\*\*
30. \* Establishes the Datasource required for the servlet
31. \* @throws ServletException
32. \*/
33. **public** **void** init() **throws** ServletException{
34. **try** {
36. InitialContext ctx = **new** InitialContext();
37. datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/LiveDataSource");
38. } **catch**(Exception e) {
39. **throw** **new** ServletException(e.toString());
40. }
41. }

44. **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
46. }
48. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
49. response.setContentType("text/html; charset=UTF-8");
50. response.setCharacterEncoding("UTF-8");
51. request.setCharacterEncoding("UTF-8");
52. PrintWriter out = response.getWriter();
53. out.println("<html>");
54. out.println("<head><title>Παράδειγμα Servlet με Βάση</title></head>");
55. out.println("<body>");
57. **if**(LoginServlet.session!=**null**) {
58. **try** {
59. Connection con = datasource.getConnection();
61. PreparedStatement clientinfo = con.prepareStatement(((Client) LoginServlet.user).viewPersonalData());
62. clientinfo.setString(1, ((Client)LoginServlet.user).getUsername());
63. ResultSet rs = clientinfo.executeQuery();
64. **int** i = 0;
65. String username = "";
66. String name = "";
67. String surname = "";
68. String tin = "";
69. String phone = "";
71. **while** (rs.next()) {
72. i++;
74. username = rs.getString("username");
75. name = rs.getString("name");
76. surname = rs.getString("surname");
77. tin = rs.getString("tin");
78. phone = rs.getString("phone\_number");
80. request.setAttribute("username", username);
81. request.setAttribute("name", name);
82. request.setAttribute("surname", surname);
83. request.setAttribute("tin", tin);
84. request.setAttribute("phone", phone);
86. }
87. //                request.setAttribute("session\_info", ((Seller)LoginServlet.session.getAttribute("seller")).getName() + " "
88. //                        + ((Seller)LoginServlet.session.getAttribute("seller")).getSurname());
89. request.getRequestDispatcher("/client\_info.jsp").forward(request, response);
91. clientinfo.close();
92. rs.close();
94. con.close();
95. } **catch** (Exception e) {
96. out.println("Database connection problem");
97. out.println("<br>");
98. out.println(e.getLocalizedMessage());
99. }
100. }
101. **else** {
102. out.println("Session Timeout");
103. }
104. }
106. **public** **static** Client getClient(String name, String surname, String username, String number){
107. client.setName(name);
108. client.setSurname(surname);
109. client.setUsername(username);
110. client.number = number;
111. **return** client;
112. }
113. }

5.3.6. Κώδικας servlet AdminServlet:

1. **package** netprog.servlets.Admin;
3. **import** netprog.classes.users.Admin;
4. **import** netprog.classes.users.Client;
5. **import** netprog.classes.users.Seller;
6. **import** netprog.servlets.Encryption;
7. **import** netprog.servlets.LoginServlet;
9. **import** javax.naming.InitialContext;
10. **import** javax.servlet.ServletException;
11. **import** javax.servlet.annotation.WebServlet;
12. **import** javax.servlet.http.HttpServlet;
13. **import** javax.servlet.http.HttpServletRequest;
14. **import** javax.servlet.http.HttpServletResponse;
15. **import** javax.sql.DataSource;
16. **import** java.io.IOException;
17. **import** java.io.PrintWriter;
18. **import** java.sql.Connection;
19. **import** java.sql.PreparedStatement;
20. **import** java.sql.Statement;
22. @WebServlet("/adminservlet")
23. **public** **class** AdminServlet **extends** HttpServlet {
24. **private** **static** **final** **long** serialVersionUID = 1L;
26. **private** **static** Admin admin = **new** Admin();
28. **private** DataSource datasource = **null**;
30. /\*\*
31. \* Establishes the Datasource required for the servlet
32. \* @throws ServletException
33. \*/
34. **public** **void** init() **throws** ServletException{
35. **try** {
37. InitialContext ctx = **new** InitialContext();
38. datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/LiveDataSource");
39. } **catch**(Exception e) {
40. **throw** **new** ServletException(e.toString());
41. }
42. }
44. **public** **static** Admin getAdmin(String name, String surname){
45. admin.setName(name);
46. admin.setSurname(surname);
47. **return** admin;
48. }
50. **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException{
51. response.setContentType("text/html; charset=UTF-8");
52. response.setCharacterEncoding("UTF-8");
53. request.setCharacterEncoding("UTF-8");
54. PrintWriter out = response.getWriter();
55. String username = request.getParameter("new\_user");
56. String password = request.getParameter("new\_pass");
57. String name = request.getParameter("new\_name");
58. String surname = request.getParameter("new\_surname");
59. **try** {
60. Connection con = datasource.getConnection();
62. PreparedStatement preparedStatement = con.prepareStatement(((Admin) LoginServlet.user).addSeller());
63. preparedStatement.setString(1, username);
64. preparedStatement.setString(2, Encryption.getHashMD5(password));
65. preparedStatement.setString(3, name);
66. preparedStatement.setString(4, surname);
68. preparedStatement.executeUpdate();
69. request.getRequestDispatcher("/admin.jsp").forward(request, response);
71. preparedStatement.close();
72. con.close();
73. } **catch**(Exception e) {
74. out.println("Database connection problem");
75. //out.println("<br>");
76. //out.println(e.getLocalizedMessage());
77. }
78. }
80. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
82. }
83. }

5.3.7. Κώδικας BillIssuanceServlet:

1. **package** netprog.servlets;
3. **import** netprog.classes.users.Admin;
4. **import** netprog.classes.users.Client;
5. **import** netprog.classes.users.Seller;
6. **import** javax.naming.InitialContext;
7. **import** javax.servlet.ServletException;
8. **import** javax.servlet.annotation.WebServlet;
9. **import** javax.servlet.http.HttpServlet;
10. **import** javax.servlet.http.HttpServletRequest;
11. **import** javax.servlet.http.HttpServletResponse;
12. **import** javax.sql.DataSource;
13. **import** java.io.IOException;
14. **import** java.io.PrintWriter;
15. **import** java.sql.\*;
16. **import** java.time.LocalDate;
18. @WebServlet("/billissuanceservlet")
19. **public** **class** BillIssuanceServlet **extends** HttpServlet {
20. **private** **static** **final** **long** serialVersionUID = 1L;
22. **public** Seller seller;
23. **public** String joined = "";
24. **public** String callto = "";
26. **private** DataSource datasource = **null**;
28. /\*\*
29. \* Establishes the Datasource required for the servlet
30. \* @throws ServletException
31. \*/
32. **public** **void** init() **throws** ServletException{
33. **try** {
34. InitialContext ctx = **new** InitialContext();
35. datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/LiveDataSource");
36. } **catch**(Exception e) {
37. **throw** **new** ServletException(e.toString());
38. }
39. }
41. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException{
42. response.setContentType("text/html; charset=UTF-8");
43. response.setCharacterEncoding("UTF-8");
44. request.setCharacterEncoding("UTF-8");
45. PrintWriter out = response.getWriter();
46. out.println("<html>");
47. out.println("<head><title>Παράδειγμα Servlet με Βάση</title></head>");
48. out.println("<body>");
50. **if** (LoginServlet.session!=**null**) {
51. **try** {
52. Connection con = datasource.getConnection();
53. String phone = request.getParameter("client\_phone");
54. PreparedStatement showClient = con.prepareStatement(((Seller) LoginServlet.user).showClientPartially());
55. showClient.setString(1, phone);
57. ResultSet rs = showClient.executeQuery();
59. String username = **null**;
60. String name = **null**;
61. String surname = **null**;
62. **boolean** found = **false**;
63. **while** (rs.next()) {
64. username = rs.getString("username");
65. name = rs.getString("name");
66. surname = rs.getString("surname");
67. found = **true**;
68. }
69. request.setAttribute("username", username);
70. request.setAttribute("name", name);
71. request.setAttribute("surname", surname);
72. request.setAttribute("phone", phone);
74. rs.close();
75. showClient.close();
76. con.close();
78. request.getRequestDispatcher("/bill\_issuance.jsp").forward(request, response);
79. } **catch** (Exception e) {
80. out.println("Database connection problem");
81. //out.println("<br>");
82. //out.println(e.getLocalizedMessage());
83. }
84. }
85. **else**
86. out.println("Session Timeout!");
87. }
89. **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException{
90. response.setContentType("text/html; charset=UTF-8");
91. response.setCharacterEncoding("UTF-8");
92. request.setCharacterEncoding("UTF-8");
93. PrintWriter out = response.getWriter();
95. **if** (LoginServlet.session!=**null**) {
96. **try** {
97. Connection con = datasource.getConnection();
99. **int** calls\_number = Integer.parseInt(request.getParameter("i"));
100. String number = request.getParameter("phone");
102. **int** mes\_number = Integer.parseInt(request.getParameter("mes\_quantity"));
103. **int** total\_duration = 0;
105. **for** (**int** i = 0; i < calls\_number; i++) {
106. callto = request.getParameter("callto" + (i + 1));
107. joined = request.getParameter("date" + (i + 1)) + " " + request.getParameter("time" + (i + 1));
108. Timestamp datetime = Timestamp.valueOf(joined);
109. **int** min\_dur = Integer.parseInt(request.getParameter("min\_dur" + (i + 1)));
110. **double** sec\_dur = Double.parseDouble(request.getParameter("sec\_dur" + (i + 1)));
112. total\_duration += min\_dur;
113. total\_duration += Math.ceil(sec\_dur / 60.0);
115. PreparedStatement ps = con.prepareStatement(((Seller) LoginServlet.user).insertCalls());
116. ps.setString(1, number);
117. ps.setString(2, callto);
118. ps.setTimestamp(3, datetime);
119. ps.setInt(4, min\_dur);
120. ps.setInt(5, (**int**) sec\_dur);
122. ps.executeUpdate();
124. ps.close();
125. }
127. **double** total\_cost = 0.0;
129. PreparedStatement clientprogram = con.prepareStatement(((Seller) LoginServlet.user).getClientProgram());
131. clientprogram.setString(1, number);
133. ResultSet resultSet = clientprogram.executeQuery();
135. **int** fixed\_charge = 0;
136. **double** minutes\_charge = 0.0;
137. **double** message\_charge = 0.0;
138. **int** minutes = 0;
139. **int** messages = 0;
140. **while** (resultSet.next()) {
141. fixed\_charge = resultSet.getInt("fixed\_charge");
142. minutes\_charge = resultSet.getDouble("minutes\_charge");
143. message\_charge = resultSet.getDouble("messages\_charge");
144. minutes = resultSet.getInt("minutes");
145. messages = resultSet.getInt("messages");
146. }
148. resultSet.close();
149. clientprogram.close();
151. **if** (minutes >= total\_duration && messages >= mes\_number) {
152. total\_cost = fixed\_charge;
153. } **else** {
154. **double** call\_cost, message\_cost = 0.0;
155. **if** (minutes < total\_duration && messages < mes\_number) {
156. call\_cost = (total\_duration - minutes) \* minutes\_charge;
157. message\_cost = (mes\_number - messages) \* message\_charge;
158. total\_cost = call\_cost + message\_cost + fixed\_charge;
159. } **else** **if** (minutes < total\_duration) {
160. call\_cost = (total\_duration - minutes) \* minutes\_charge;
161. total\_cost = call\_cost + fixed\_charge;
162. } **else** {
163. message\_cost = (mes\_number - messages) \* message\_charge;
164. total\_cost = message\_cost + fixed\_charge;
165. }
166. }
168. PreparedStatement billissuance = con.prepareStatement(((Seller) LoginServlet.user).billIssuance());
169. billissuance.setString(1, number);
170. billissuance.setString(2, LocalDate.now().toString());
171. billissuance.setInt(3, calls\_number);
172. billissuance.setInt(4, mes\_number);
173. billissuance.setInt(5, total\_duration);
174. billissuance.setDouble(6, total\_cost);
175. billissuance.setByte(7, (**byte**) 0);
177. billissuance.executeUpdate();
179. billissuance.close();
181. request.getRequestDispatcher("/seller.jsp").forward(request, response);
182. con.close();
183. } **catch** (Exception e) {
184. out.println("Database connection problem");
185. out.println("<br>");
186. out.println(e.getLocalizedMessage());
187. out.println("<br>");
188. out.println(callto);
189. out.println(joined);
190. }
191. }
192. **else**
193. out.println("Session Timeout!");
194. }
195. }

5.3.8 Κώδικας BillServlet:

1. **package** netprog.servlets;
3. **import** netprog.classes.users.Client;
5. **import** javax.naming.InitialContext;
6. **import** javax.servlet.ServletException;
7. **import** javax.servlet.annotation.WebServlet;
8. **import** javax.servlet.http.HttpServlet;
9. **import** javax.servlet.http.HttpServletRequest;
10. **import** javax.servlet.http.HttpServletResponse;
11. **import** javax.sql.DataSource;
12. **import** java.io.IOException;
13. **import** java.io.PrintWriter;
14. **import** java.sql.Connection;
15. **import** java.sql.PreparedStatement;
16. **import** java.sql.ResultSet;
17. **import** java.sql.Timestamp;
19. @WebServlet("/billservlet")
20. **public** **class** BillServlet **extends** HttpServlet {
21. **private** **static** **final** **long** serialVersionUID = 1L;
23. **private** DataSource datasource = **null**;
25. /\*\*
26. \* Establishes the Datasource required for the servlet
27. \* @throws ServletException
28. \*/
29. **public** **void** init() **throws** ServletException{
30. **try** {
32. InitialContext ctx = **new** InitialContext();
33. datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/LiveDataSource");
34. } **catch**(Exception e) {
35. **throw** **new** ServletException(e.toString());
36. }
37. }
39. **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
40. response.setContentType("text/html; charset=UTF-8");
41. response.setCharacterEncoding("UTF-8");
42. request.setCharacterEncoding("UTF-8");
43. PrintWriter out = response.getWriter();
44. out.println("<html>");
45. out.println("<head><title>Παράδειγμα Servlet με Βάση</title></head>");
46. out.println("<body>");
48. **if**(LoginServlet.session!=**null**) {
49. **try** {
50. Connection con = datasource.getConnection();
52. PreparedStatement payment = con.prepareStatement(((Client) LoginServlet.user).payBill());
54. payment.setString(1, ((Client) LoginServlet.user).number);
55. payment.setString(2, request.getParameter("date\_issued") + "%");
56. **int** i = payment.executeUpdate();
58. response.sendRedirect("client.jsp");
60. payment.close();
62. con.close();
63. } **catch** (Exception e) {
64. out.println("Database connection problem");
65. out.println("<br>");
66. out.println(e.getLocalizedMessage());
67. }
68. }
69. **else** {
70. out.println("Session Timeout");
71. }
72. }
74. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
75. response.setContentType("text/html; charset=UTF-8");
76. response.setCharacterEncoding("UTF-8");
77. request.setCharacterEncoding("UTF-8");
78. PrintWriter out = response.getWriter();
79. out.println("<html>");
80. out.println("<head><title>Παράδειγμα Servlet με Βάση</title></head>");
81. out.println("<body>");
83. **if**(LoginServlet.session!=**null**) {
84. **try** {
85. Connection con1 = datasource.getConnection();
87. PreparedStatement showBill = con1.prepareStatement(((Client) LoginServlet.user).viewBill());
88. showBill.setString(1, ((Client)LoginServlet.user).number);
89. showBill.setString(2, request.getParameter("callformonth") + "-%");
90. ResultSet rs1 = showBill.executeQuery();
92. String billrow = "";
93. **while** (rs1.next()) {
94. String number\_of\_client = rs1.getString("number");
95. String date\_issued = rs1.getString("date\_issued");
96. **int** calls\_number = rs1.getInt("calls\_number");
97. **int** messages\_number = rs1.getInt("messages\_number");
98. **int** total\_dur = rs1.getInt("total\_duration");
99. **double** total\_cost = rs1.getDouble("total\_cost");
100. **byte** paid = rs1.getByte("paid");
102. billrow = createHTMLRow(number\_of\_client, date\_issued, calls\_number, messages\_number, total\_dur);
103. request.setAttribute("billrow", billrow);
104. request.setAttribute("total\_cost", total\_cost);
105. request.setAttribute("paid", paid);
106. }
107. request.setAttribute("month", request.getParameter("callformonth"));
109. request.getRequestDispatcher("/paybill.jsp").forward(request, response);
111. showBill.close();
112. rs1.close();
113. con1.close();
114. }
115. **catch** (Exception e) {
116. out.println("Database connection problem");
117. out.println("<br>");
118. out.println(e.getLocalizedMessage());
119. }
120. }
121. **else**{
122. out.println("Session Timeout");
123. }
124. }
126. **private** String createHTMLRow(String n, String d, **int** calls\_n, **int** mes\_n, **int** total\_dur) {
127. String row = "<tr>";
128. row  += "<td>" + n + "</td>";
129. row  += "<td>" + d + "</td>";
130. row  += "<td>" + calls\_n + "</td>";
131. row  += "<td>" + mes\_n + "</td>";
132. row  += "<td>" + total\_dur + "</td>";
133. row += "</tr>";
134. **return** row;
136. }
137. }

5.3.9 Κώδικας CallServlet:

1. **package** netprog.servlets;
3. **import** netprog.classes.users.Client;
4. **import** netprog.classes.users.Seller;
6. **import** javax.naming.InitialContext;
7. **import** javax.servlet.ServletException;
8. **import** javax.servlet.annotation.WebServlet;
9. **import** javax.servlet.http.HttpServlet;
10. **import** javax.servlet.http.HttpServletRequest;
11. **import** javax.servlet.http.HttpServletResponse;
12. **import** javax.sql.DataSource;
13. **import** java.io.IOException;
14. **import** java.io.PrintWriter;
15. **import** java.sql.Connection;
16. **import** java.sql.PreparedStatement;
17. **import** java.sql.ResultSet;
18. **import** java.sql.Timestamp;
19. **import** java.time.YearMonth;
21. @WebServlet("/callservlet")
22. **public** **class** CallServlet **extends** HttpServlet {
23. **private** **static** **final** **long** serialVersionUID = 1L;
25. **private** DataSource datasource = **null**;
27. **private** String URL = "";
29. /\*\*
30. \* Establishes the Datasource required for the servlet
31. \* @throws ServletException
32. \*/
33. **public** **void** init() **throws** ServletException{
34. **try** {
36. InitialContext ctx = **new** InitialContext();
37. datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/LiveDataSource");
38. } **catch**(Exception e) {
39. **throw** **new** ServletException(e.toString());
40. }
41. }
43. **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
45. }
47. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
48. response.setContentType("text/html; charset=UTF-8");
49. response.setCharacterEncoding("UTF-8");
50. request.setCharacterEncoding("UTF-8");
51. PrintWriter out = response.getWriter();
52. out.println("<html>");
53. out.println("<head><title>Παράδειγμα Servlet με Βάση</title></head>");
54. out.println("<body>");
56. **if**(LoginServlet.session!=**null**) {
57. **try** {
58. Connection con = datasource.getConnection();
60. PreparedStatement showCalls = con.prepareStatement(((Client) LoginServlet.user).viewCallHistory());
61. showCalls.setString(1, ((Client)LoginServlet.user).number);
62. showCalls.setString(2, request.getParameter("callformonth") + "-%");
63. ResultSet rs = showCalls.executeQuery();
65. **int** i = 0;
66. String htmlRow = "";
67. **while** (rs.next()) {
68. i++;
70. String caller = rs.getString("caller");
71. String number = rs.getString("number");
72. Timestamp time = rs.getTimestamp("time");
73. String min\_duration = rs.getString("minutes\_duration");
74. String sec\_duration = rs.getString("seconds\_duration");
76. htmlRow = createHTMLRow(caller, number, time, min\_duration, sec\_duration);
77. request.setAttribute("callrow" + i, htmlRow);
79. }
80. request.setAttribute("time", request.getParameter("callformonth"));
81. request.setAttribute("numberofcalls", i);
83. showCalls.close();
84. rs.close();
85. con.close();
87. **if** (request.getParameter("bill")!= **null** && request.getParameter("bill").equals("bill")){
88. Connection con1 = datasource.getConnection();
90. PreparedStatement showBill = con1.prepareStatement(((Client)LoginServlet.user).viewBill());
91. showBill.setString(1, ((Client)LoginServlet.user).number);
92. showBill.setString(2, request.getParameter("callformonth") + "-%");
93. ResultSet rs1 = showBill.executeQuery();
95. String billrow = "";
96. **while** (rs1.next()){
97. String number\_of\_client=rs1.getString("number");
98. String date\_issued = rs1.getString("date\_issued");
99. **int** calls\_number = rs1.getInt("calls\_number");
100. **int** messages\_number = rs1.getInt("messages\_number");
101. **int** total\_dur = rs1.getInt("total\_duration");
102. **double** total\_cost = rs1.getDouble("total\_cost");
103. **byte** paid = rs1.getByte("paid");
105. billrow = createHTMLRow(number\_of\_client, date\_issued, calls\_number, messages\_number, total\_dur);
106. request.setAttribute("billrow", billrow);
107. request.setAttribute("total\_cost", total\_cost);
108. request.setAttribute("paid", paid);
109. }
111. showBill.close();
112. rs1.close();
113. con1.close();
115. URL = "/showbill.jsp";
116. }
117. **else** {
118. URL = "/showcalls.jsp";
119. }
120. //                request.setAttribute("session\_info", ((Seller)LoginServlet.session.getAttribute("seller")).getName() + " "
121. //                        + ((Seller)LoginServlet.session.getAttribute("seller")).getSurname());
122. request.getRequestDispatcher(URL).forward(request, response);
124. } **catch** (Exception e) {
125. out.println("Database connection problem");
126. out.println("<br>");
127. out.println(e.getLocalizedMessage());
128. }
129. }
130. **else** {
131. out.println("Session Timeout");
132. }
133. }
135. /\*\*
136. \* This function creates a simple HTML table row, which includes information about telecommunications program
137. \* @param caller Caller
138. \* @param number Number of the call
139. \* @param timestamp time
140. \* @param min\_dur Call's duration (minutes part)
141. \* @param sec\_dur Call's duration (seconds part)
142. \* @return the complete row
143. \*/
144. **private** String createHTMLRow(String caller, String number, Timestamp timestamp, String min\_dur, String sec\_dur) {
145. String row = "<tr>";
146. row  += "<td>" + caller + "</td>";
147. row  += "<td>" + number + "</td>";
148. row  += "<td>" + timestamp + "</td>";
149. row  += "<td>" + min\_dur + "</td>";
150. row  += "<td>" + sec\_dur + "</td>";
151. row += "</tr>";
152. **return** row;
154. }
156. **private** String createHTMLRow(String n, String d, **int** calls\_n, **int** mes\_n, **int** total\_dur) {
157. String row = "<tr>";
158. row  += "<td>" + n + "</td>";
159. row  += "<td>" + d + "</td>";
160. row  += "<td>" + calls\_n + "</td>";
161. row  += "<td>" + mes\_n + "</td>";
162. row  += "<td>" + total\_dur + "</td>";
163. row += "</tr>";
164. **return** row;
166. }
167. }

5.3.10 Κώδικας Cust\_programServlet:

1. **package** netprog.servlets;
3. **import** netprog.classes.users.Seller;
5. **import** javax.naming.InitialContext;
6. **import** javax.servlet.ServletException;
7. **import** javax.servlet.annotation.WebServlet;
8. **import** javax.servlet.http.HttpServlet;
9. **import** javax.servlet.http.HttpServletRequest;
10. **import** javax.servlet.http.HttpServletResponse;
11. **import** javax.sql.DataSource;
12. **import** java.io.IOException;
13. **import** java.io.PrintWriter;
14. **import** java.sql.Connection;
15. **import** java.sql.PreparedStatement;
16. **import** java.sql.ResultSet;
17. **import** java.sql.Statement;
19. @WebServlet("/custprogservlet")
20. **public** **class** Cust\_programServlet **extends** HttpServlet {
21. **private** **static** **final** **long** serialVersionUID = 1L;
22. **private** DataSource datasource = **null**;
24. /\*\*
25. \* Establishes the Datasource required for the servlet
26. \* @throws ServletException
27. \*/
28. **public** **void** init() **throws** ServletException{
29. **try** {
31. InitialContext ctx = **new** InitialContext();
32. datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/LiveDataSource");
33. } **catch**(Exception e) {
34. **throw** **new** ServletException(e.toString());
35. }
36. }
38. /\*\*
39. \* Function to be executed when submitting a form, using POST method
40. \* @param request
41. \* @param response
42. \* @throws ServletException
43. \* @throws IOException
44. \*/
45. **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
46. response.setContentType("text/html; charset=UTF-8");
47. response.setCharacterEncoding("UTF-8");
48. request.setCharacterEncoding("UTF-8");
49. PrintWriter out = response.getWriter();
51. **if** (LoginServlet.session!=**null**) {
52. **try** {
53. Connection con = datasource.getConnection();
54. Statement stmt = con.createStatement();
56. PreparedStatement insertClientProgram = con.prepareStatement(((Seller) LoginServlet.user).clientProgram());
58. insertClientProgram.setString(1, request.getParameter("number"));
59. insertClientProgram.setString(2, request.getParameter("rad\_htmlrow"));
61. insertClientProgram.executeUpdate();
62. request.getRequestDispatcher("/seller.jsp").forward(request, response);
64. insertClientProgram.close();
65. con.close();
67. } **catch** (Exception e) {
68. out.println("Database connection problem");
69. //out.println("<br>");
70. //out.println(e.getLocalizedMessage());
71. }
72. }
73. **else**
74. out.println("Session Timeout!");
75. }
77. /\*\*
78. \* This function creates a simple HTML table row, which includes information about telecommunications program
79. \* @param prog\_name Program's name
80. \* @param f\_charge Fixed charge
81. \* @param min\_charge charge per minute (except contract charge)
82. \* @param mes\_charge charge per message (except contract charge)
83. \* @param min free minutes
84. \* @param mes free messages
85. \* @return the complete row, without row-closing character (</tr>)
86. \*/
87. **private** String createHTMLRow(String prog\_name, **int** f\_charge, **double** min\_charge, **double** mes\_charge, **int** min, **int** mes) {
88. String row = "<tr>";
89. row  += "<td>" + prog\_name + "</td>";
90. row  += "<td>" + f\_charge + "</td>";
91. row  += "<td>" + min\_charge + "</td>";
92. row  += "<td>" + mes\_charge + "</td>";
93. row  += "<td>" + min + "</td>";
94. row  += "<td>" + mes + "</td>";
95. **return** row;
96. }
98. /\*\*
99. \* Function to be executed when submitting a form, using GET method
100. \* @param request
101. \* @param response
102. \* @throws ServletException
103. \* @throws IOException
104. \*/
105. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
106. response.setContentType("text/html; charset=UTF-8");
107. response.setCharacterEncoding("UTF-8");
108. request.setCharacterEncoding("UTF-8");
109. PrintWriter out = response.getWriter();
111. **if** (LoginServlet.session!=**null**) {
112. **try** {
113. Connection con = datasource.getConnection();
115. String phone = request.getParameter("phone");
117. PreparedStatement showClient = con.prepareStatement(((Seller) LoginServlet.user).showClientPartially());
118. showClient.setString(1, phone);
119. ResultSet rs = showClient.executeQuery();
121. String username = **null**;
122. String name = **null**;
123. String surname = **null**;
125. **boolean** found = **false**;
126. **while** (rs.next()) {
127. username = rs.getString("username");
128. name = rs.getString("name");
129. surname = rs.getString("surname");
131. found = **true**;
132. }
134. request.setAttribute("username", username);
135. request.setAttribute("name", name);
136. request.setAttribute("surname", surname);
137. request.setAttribute("phone", phone);
139. rs.close();
140. showClient.close();
141. con.close();
143. **if** (found) {
144. Connection con1 = datasource.getConnection();
146. PreparedStatement showPrograms = con1.prepareStatement(((Seller) LoginServlet.user).showPrograms());
147. ResultSet rs1 = showPrograms.executeQuery();
149. **int** i = 0;
150. String htmlRow = "";
151. **while** (rs1.next()) {
152. i++;
154. String pr\_name = rs1.getString("program\_name");
155. **int** fixed\_charge = rs1.getInt("fixed\_charge");
156. **double** minutes\_charge = rs1.getDouble("minutes\_charge");
157. **double** messages\_charge = rs1.getDouble("messages\_charge");
158. **int** minutes = rs1.getInt("minutes");
159. **int** messages = rs1.getInt("messages");
160. htmlRow = createHTMLRow(pr\_name, fixed\_charge, minutes\_charge, messages\_charge, minutes, messages);
161. request.setAttribute("htmlrow" + i, htmlRow);
162. request.setAttribute("pr\_name" + i, pr\_name);
164. }
166. request.setAttribute("numberofrows", i);
168. rs1.close();
169. showPrograms.close();
170. con1.close();
171. }
173. request.setAttribute("found", found);
174. request.getRequestDispatcher("/cust\_program.jsp").forward(request, response);
176. } **catch** (Exception e) {
177. out.println("Database connection problem");
178. //            out.println("<br>");
179. //            out.println(e.getLocalizedMessage());
180. }
181. }
182. **else**
183. out.println("Session Timeout!");
184. }
185. }

5.3.11 Κώδικας EditClientServlet:

1. **package** netprog.servlets;
3. **import** netprog.classes.users.Seller;
5. **import** javax.naming.InitialContext;
6. **import** javax.servlet.ServletException;
7. **import** javax.servlet.annotation.WebServlet;
8. **import** javax.servlet.http.HttpServlet;
9. **import** javax.servlet.http.HttpServletRequest;
10. **import** javax.servlet.http.HttpServletResponse;
11. **import** javax.sql.DataSource;
12. **import** java.io.IOException;
13. **import** java.io.PrintWriter;
14. **import** java.sql.Connection;
15. **import** java.sql.PreparedStatement;
16. **import** java.sql.ResultSet;
18. @WebServlet("/editclientservlet")
19. **public** **class** EditClientServlet **extends** HttpServlet {
20. **private** **static** **final** **long** serialVersionUID = 1L;
22. **private** DataSource datasource = **null**;
24. /\*\*
25. \* Establishes the Datasource required for the servlet
26. \* @throws ServletException
27. \*/
28. **public** **void** init() **throws** ServletException{
29. **try** {
30. InitialContext ctx = **new** InitialContext();
31. datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/LiveDataSource");
32. } **catch**(Exception e) {
33. **throw** **new** ServletException(e.toString());
34. }
35. }
37. **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
38. response.setContentType("text/html; charset=UTF-8");
39. response.setCharacterEncoding("UTF-8");
40. request.setCharacterEncoding("UTF-8");
41. PrintWriter out = response.getWriter();
42. out.println("<html>");
43. out.println("<head><title>Παράδειγμα Servlet με Βάση</title></head>");
44. out.println("<body>");
46. **if** (LoginServlet.session != **null**) {
47. **try** {
48. Connection con = datasource.getConnection();
50. String phone = request.getParameter("client\_phone");

53. **if** (request.getParameter("choices").equals("delete")){
54. PreparedStatement ps = con.prepareStatement(((Seller)LoginServlet.user).deleteFromClients());
55. ps.setString(1, phone);
56. ps.executeUpdate();
57. ps.close();
59. PreparedStatement ps1 = con.prepareStatement(((Seller)LoginServlet.user).deleteFromNumbers());
60. ps1.setString(1, phone);
61. ps1.executeUpdate();
62. ps1.close();
64. PreparedStatement ps2 = con.prepareStatement(((Seller)LoginServlet.user).deleteFromCalls());
65. ps2.setString(1, phone);
66. ps2.executeUpdate();
67. ps2.close();
69. PreparedStatement ps3 = con.prepareStatement(((Seller)LoginServlet.user).deleteFromBills());
70. ps3.setString(1, phone);
71. ps3.executeUpdate();
72. ps3.close();
73. }
74. **else** **if** (request.getParameter("choices").equals("update")){
75. PreparedStatement preparedStatement = con.prepareStatement(((Seller)LoginServlet.user).editClient());
76. preparedStatement.setString(1, request.getParameter("user"));
77. preparedStatement.setString(2, request.getParameter("name"));
78. preparedStatement.setString(3, request.getParameter("surname"));
79. preparedStatement.setString(4, request.getParameter("tin"));
80. preparedStatement.setString(5, phone);
82. preparedStatement.executeUpdate();
83. preparedStatement.close();
84. }
86. con.close();
88. request.getRequestDispatcher("/seller.jsp").forward(request, response);
89. } **catch** (Exception e) {
90. out.println("Database connection problem");
91. out.println("<br>");
92. out.println(e.getLocalizedMessage());
93. }
94. }
95. **else**
96. out.println("Session timeout!");
97. }
99. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
100. response.setContentType("text/html; charset=UTF-8");
101. response.setCharacterEncoding("UTF-8");
102. request.setCharacterEncoding("UTF-8");
103. PrintWriter out = response.getWriter();
104. out.println("<html>");
105. out.println("<head><title>Παράδειγμα Servlet με Βάση</title></head>");
106. out.println("<body>");
108. **if** (LoginServlet.session != **null**) {
109. **try** {
110. Connection con = datasource.getConnection();
111. String phone = request.getParameter("phone");
112. PreparedStatement showClient = con.prepareStatement(((Seller) LoginServlet.user).showClientInfo());
113. showClient.setString(1, phone);
115. ResultSet rs = showClient.executeQuery();
117. String username = **null**;
118. String password = **null**;
119. String name = **null**;
120. String surname = **null**;
121. String tin = **null**;
122. **while** (rs.next()) {
123. username = rs.getString("username");
124. password = Encryption.getHashMD5(rs.getString("password"));
125. name = rs.getString("name");
126. surname = rs.getString("surname");
127. tin = rs.getString("tin");
128. }
129. request.setAttribute("username", username);
130. request.setAttribute("password", password);
131. request.setAttribute("name", name);
132. request.setAttribute("surname", surname);
133. request.setAttribute("tin", tin);
134. request.setAttribute("phone", phone);
136. rs.close();
137. showClient.close();
138. con.close();
140. request.getRequestDispatcher("/editclient.jsp").forward(request, response);
141. } **catch** (Exception e) {
142. out.println("Database connection problem");
143. out.println("<br>");
144. out.println(e.getLocalizedMessage());
145. out.println(request.getParameter("choices"));
146. }
147. }
148. **else**
149. out.println("Session timeout!");
150. }
151. }

5.3.12 Κώδικας EditProgramServlet:

1. **package** netprog.servlets;
3. **import** netprog.classes.users.Admin;
5. **import** javax.naming.InitialContext;
6. **import** javax.servlet.ServletException;
7. **import** javax.servlet.annotation.WebServlet;
8. **import** javax.servlet.http.HttpServlet;
9. **import** javax.servlet.http.HttpServletRequest;
10. **import** javax.servlet.http.HttpServletResponse;
11. **import** javax.sql.DataSource;
12. **import** java.io.IOException;
13. **import** java.io.PrintWriter;
14. **import** java.sql.Connection;
15. **import** java.sql.PreparedStatement;
16. **import** java.sql.ResultSet;
18. @WebServlet("/editprogramservlet")
19. **public** **class** EditProgramServlet **extends** HttpServlet {
20. **private** **static** **final** **long** serialVersionUID = 1L;
22. **private** DataSource datasource = **null**;
24. /\*\*
25. \* Establishes the Datasource required for the servlet
26. \* @throws ServletException
27. \*/
28. **public** **void** init() **throws** ServletException{
29. **try** {
31. InitialContext ctx = **new** InitialContext();
32. datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/LiveDataSource");
33. } **catch**(Exception e) {
34. **throw** **new** ServletException(e.toString());
35. }
36. }
38. **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
39. response.setContentType("text/html; charset=UTF-8");
40. response.setCharacterEncoding("UTF-8");
41. request.setCharacterEncoding("UTF-8");
42. PrintWriter out = response.getWriter();
43. out.println("<html>");
44. out.println("<head><title>Παράδειγμα Servlet με Βάση</title></head>");
45. out.println("<body>");
47. **if** (LoginServlet.session != **null**) {
48. **try** {
49. Connection con = datasource.getConnection();
51. String program\_name = request.getParameter("original\_name");

54. **if** (request.getParameter("choices").equals("delete")){
55. PreparedStatement ps = con.prepareStatement(((Admin)LoginServlet.user).updateNumbersDelete());
56. ps.setString(1, program\_name);
57. ps.executeUpdate();
58. ps.close();
60. PreparedStatement ps1 = con.prepareStatement(((Admin)LoginServlet.user).deletePrograms());
61. ps1.setString(1, program\_name);
62. ps1.executeUpdate();
63. ps1.close();
64. }
65. **else** **if** (request.getParameter("choices").equals("update")){
66. PreparedStatement preparedStatement = con.prepareStatement(((Admin)LoginServlet.user).editPrograms());
67. preparedStatement.setString(1, request.getParameter("pr\_name"));
68. preparedStatement.setInt(2, Integer.parseInt(request.getParameter("fixed\_charge")));
69. preparedStatement.setDouble(3, Double.parseDouble(request.getParameter("min\_charge")));
70. preparedStatement.setDouble(4, Double.parseDouble(request.getParameter("mes\_charge")));
71. preparedStatement.setInt(5, Integer.parseInt(request.getParameter("minutes")));
72. preparedStatement.setInt(6, Integer.parseInt(request.getParameter("messages")));
73. preparedStatement.setString(7, program\_name);
75. preparedStatement.executeUpdate();
76. preparedStatement.close();
77. }
79. con.close();
81. request.getRequestDispatcher("/admin.jsp").forward(request, response);
82. } **catch** (Exception e) {
83. out.println("Database connection problem");
84. out.println("<br>");
85. out.println(e.getLocalizedMessage());
86. }
87. }
88. **else**
89. out.println("Session timeout!");
90. }
92. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
93. response.setContentType("text/html; charset=UTF-8");
94. response.setCharacterEncoding("UTF-8");
95. request.setCharacterEncoding("UTF-8");
96. PrintWriter out = response.getWriter();
97. out.println("<html>");
98. out.println("<head><title>Παράδειγμα Servlet με Βάση</title></head>");
99. out.println("<body>");
101. **if** (LoginServlet.session != **null**) {
102. **try** {
103. Connection con = datasource.getConnection();
105. String program\_name = request.getParameter("program\_name");
107. PreparedStatement ps = con.prepareStatement(((Admin)LoginServlet.user).showProgramInfo());
108. ps.setString(1, program\_name);
110. ResultSet rs = ps.executeQuery();
112. String name = "";
113. **int** fixed\_charge = 0;
114. **double** min\_charge = 0.0;
115. **double** mes\_charge = 0.0;
116. **int** minutes = 0;
117. **int** messages = 0;
118. **while**(rs.next()){
119. name = rs.getString("program\_name");
120. fixed\_charge = rs.getInt("fixed\_charge");
121. min\_charge = rs.getDouble("minutes\_charge");
122. mes\_charge = rs.getDouble("messages\_charge");
123. minutes = rs.getInt("minutes");
124. messages = rs.getInt("messages");
125. }
127. request.setAttribute("pr\_name", name);
128. request.setAttribute("fixed\_charge", fixed\_charge);
129. request.setAttribute("min\_charge", min\_charge);
130. request.setAttribute("mes\_charge", mes\_charge);
131. request.setAttribute("minutes", minutes);
132. request.setAttribute("messages", messages);
134. rs.close();
135. ps.close();
136. con.close();
138. request.getRequestDispatcher("/edit\_program.jsp").forward(request, response);
139. } **catch** (Exception e) {
140. out.println("Database connection problem");
141. out.println("<br>");
142. out.println(e.getLocalizedMessage());
143. }
144. }
145. **else**
146. out.println("Session timeout!");
147. }
148. }

5.3.13 Κώδικας EditSellerServlet:

1. **package** netprog.servlets;
3. **import** netprog.classes.users.Admin;
5. **import** javax.naming.InitialContext;
6. **import** javax.servlet.ServletException;
7. **import** javax.servlet.annotation.WebServlet;
8. **import** javax.servlet.http.HttpServlet;
9. **import** javax.servlet.http.HttpServletRequest;
10. **import** javax.servlet.http.HttpServletResponse;
11. **import** javax.sql.DataSource;
12. **import** java.io.IOException;
13. **import** java.io.PrintWriter;
14. **import** java.sql.Connection;
15. **import** java.sql.PreparedStatement;
16. **import** java.sql.ResultSet;
18. @WebServlet("/editsellerservlet")
19. **public** **class** EditSellerServlet **extends** HttpServlet {
20. **private** **static** **final** **long** serialVersionUID = 1L;
22. **private** DataSource datasource = **null**;
24. /\*\*
25. \* Establishes the Datasource required for the servlet
26. \* @throws ServletException
27. \*/
28. **public** **void** init() **throws** ServletException{
29. **try** {
31. InitialContext ctx = **new** InitialContext();
32. datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/LiveDataSource");
33. } **catch**(Exception e) {
34. **throw** **new** ServletException(e.toString());
35. }
36. }
38. **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
39. response.setContentType("text/html; charset=UTF-8");
40. response.setCharacterEncoding("UTF-8");
41. request.setCharacterEncoding("UTF-8");
42. PrintWriter out = response.getWriter();
43. out.println("<html>");
44. out.println("<head><title>Παράδειγμα Servlet με Βάση</title></head>");
45. out.println("<body>");
47. **if** (LoginServlet.session != **null**) {
48. **try** {
49. Connection con = datasource.getConnection();
51. String username = request.getParameter("original\_user");

54. **if** (request.getParameter("choices").equals("delete")){
55. PreparedStatement ps = con.prepareStatement(((Admin)LoginServlet.user).deleteSeller());
56. ps.setString(1, username);
57. ps.executeUpdate();
58. ps.close();
60. }
61. **else** **if** (request.getParameter("choices").equals("update")){
62. PreparedStatement preparedStatement = con.prepareStatement(((Admin)LoginServlet.user).editSeller());
63. preparedStatement.setString(1, request.getParameter("user"));
64. preparedStatement.setString(2, request.getParameter("name"));
65. preparedStatement.setString(3, request.getParameter("surname"));
66. preparedStatement.setString(4, username);
68. preparedStatement.executeUpdate();
69. preparedStatement.close();
70. }
72. con.close();
74. request.getRequestDispatcher("/admin.jsp").forward(request, response);
75. } **catch** (Exception e) {
76. out.println("Database connection problem");
77. out.println("<br>");
78. out.println(e.getLocalizedMessage());
79. }
80. }
81. **else**
82. out.println("Session timeout!");
83. }
85. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {
86. response.setContentType("text/html; charset=UTF-8");
87. response.setCharacterEncoding("UTF-8");
88. request.setCharacterEncoding("UTF-8");
89. PrintWriter out = response.getWriter();
90. out.println("<html>");
91. out.println("<head><title>Παράδειγμα Servlet με Βάση</title></head>");
92. out.println("<body>");
94. **if** (LoginServlet.session != **null**) {
95. **try** {
96. Connection con = datasource.getConnection();
98. String username = request.getParameter("username");
100. PreparedStatement ps = con.prepareStatement(((Admin)LoginServlet.user).showSellerInfo());
101. ps.setString(1, username);
103. ResultSet rs = ps.executeQuery();
105. String user = "";
106. String name = "";
107. String surname = "";
108. **while**(rs.next()){
109. user = rs.getString("username");
110. name = rs.getString("name");
111. surname = rs.getString("surname");
112. }
114. request.setAttribute("username", user);
115. request.setAttribute("name", name);
116. request.setAttribute("surname", surname);
118. rs.close();
119. ps.close();
120. con.close();
122. request.getRequestDispatcher("/edit\_seller.jsp").forward(request, response);
123. } **catch** (Exception e) {
124. out.println("Database connection problem");
125. out.println("<br>");
126. out.println(e.getLocalizedMessage());
127. }
128. }
129. **else**
130. out.println("Session timeout!");
131. }
132. }

5.3.14 Κώδικας NewProgramServlet:

1. **package** netprog.servlets;
3. **import** netprog.classes.users.Admin;
5. **import** javax.naming.InitialContext;
6. **import** javax.servlet.ServletException;
7. **import** javax.servlet.annotation.WebServlet;
8. **import** javax.servlet.http.HttpServlet;
9. **import** javax.servlet.http.HttpServletRequest;
10. **import** javax.servlet.http.HttpServletResponse;
11. **import** javax.sql.DataSource;
12. **import** java.io.IOException;
13. **import** java.io.PrintWriter;
14. **import** java.sql.Connection;
15. **import** java.sql.PreparedStatement;
16. **import** java.sql.ResultSet;
17. **import** java.sql.Statement;
19. @WebServlet("/newprogramservlet")
20. **public** **class** NewProgramServlet **extends** HttpServlet {
21. **private** **static** **final** **long** serialVersionUID = 1L;
22. **private** DataSource datasource = **null**;
24. /\*\*
25. \* Establishes the Datasource required for the servlet
26. \* @throws ServletException
27. \*/
28. **public** **void** init() **throws** ServletException{
29. **try** {
31. InitialContext ctx = **new** InitialContext();
32. datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/LiveDataSource");
33. } **catch**(Exception e) {
34. **throw** **new** ServletException(e.toString());
35. }
36. }
38. **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException{
39. response.setContentType("text/html; charset=UTF-8");
40. response.setCharacterEncoding("UTF-8");
41. request.setCharacterEncoding("UTF-8");
42. PrintWriter out = response.getWriter();
44. **if** (LoginServlet.session!=**null**) {
45. **try** {
46. Connection con = datasource.getConnection();
48. String program\_name = request.getParameter("pr\_name");
49. **int** fixed\_charge = Integer.parseInt(request.getParameter("fixed\_charge"));
50. **double** minutes\_charge = Double.parseDouble(request.getParameter("min\_charge"));
51. **double** messages\_charge = Double.parseDouble(request.getParameter("mes\_charge"));
52. **int** minutes = Integer.parseInt(request.getParameter("minutes"));
53. **int** messages = Integer.parseInt(request.getParameter("messages"));
55. PreparedStatement preparedStatement = con.prepareStatement(((Admin) LoginServlet.user).addPrograms());
56. preparedStatement.setString(1, program\_name);
57. preparedStatement.setInt(2, fixed\_charge);
58. preparedStatement.setDouble(3, minutes\_charge);
59. preparedStatement.setDouble(4, messages\_charge);
60. preparedStatement.setInt(5, minutes);
61. preparedStatement.setInt(6, messages);
63. preparedStatement.executeUpdate();
64. request.getRequestDispatcher("/admin.jsp").forward(request, response);
66. preparedStatement.close();
67. con.close();
68. } **catch** (Exception e) {
69. out.println("Database connection problem");
70. //out.println("<br>");
71. //out.println(e.getLocalizedMessage());
72. }
73. }
74. **else**
75. out.println("Session Timeout!");
76. }
77. }

***5.4)*** ***Αρχεία για το front-end***

Στoν φάκελο **askisi2-2020(web)/web** υπάρχουν τα αρχεία τύπου .html, .xml, .jsp, .css, τα οποία χρησιμεύουν εμφανισιακά.

**6.ΒΙΒΛΙΟΓΡΑΦΙΑ-ΠΗΓΕΣ**

* Marty Hall, Larry Brown, Servlets και Σελίδες Διακομιστή JAVA: Τεχνολογίες Πυρήνα (2η Αμερικανική Έκδοση 2007), Εκδόσεις Κλειδάριθμος ΕΠΕ, ISBN 978-960-461-001-3
* W3Schools Online Web Tutorials, <https://www.w3schools.com>
* Stack Overflow <https://stackoverflow.com>
* Coreservlets Java EE, JavaScript, jQuery, and Android Training, Tutorials, Consulting, Books, & Resources, <http://www.coreservlets.com>
* MySQL, <https://www.mysql.com>
* Apache Tomcat, <http://tomcat.apache.org>
* MDN Web Docs, <https://developer.mozilla.org>
* Tutorialspoint-Simply Learning, <https://www.tutorialspoint.com/index.htm>