ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος «Προγραμματισμός στο διαδίκτυο και στον παγκόσμιο ιστό»

ΤΕΛΙΚΗ ΕΡΓΑΣΙΑ	MobileAccountManagement
Όνομα φοιτητή –Αρ. Μητρώου	ΓΕΩΡΓΙΟΣ ΜΠΟΥΤΣΙΚΟΣ – Π20141
	ΣΤΥΛΙΑΝΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ ΒΑΡΥΜΠΟΜΠΙΩΤΗΣ - 20028
	ΓΕΩΡΓΙΟΣ ΧΡΙΣΤΟΠΟΥΛΟΣ – Π20206
	ΣΤΑΥΡΟΣ ΚΟΛΟΥΑΣ – Π18077
Ημερομηνία παράδοσης	19/07/24

Εκφώνηση της άσκησης

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΤΟ ΔΙΑΔΙΚΤΥΟ ΚΑΙ ΣΤΟΝ ΠΑΓΚΟΣΜΙΟ ΙΣΤΟ

ΕΡΓΑΣΙΑ ΜΑΘΗΜΑΤΟΣ 2023-24

ΤΕΛΙΚΟ ΠΑΡΑΔΟΤΕΟ ΜΑΘΗΜΑΤΟΣ

Ολοκλήρωση 3-tier εφαρμογής για την *έκδοση λογαριασμών κινητής τηλεφωνίας*

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

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

Αναλυτικά Βήματα:

1. **Επέκταση web project προηγούμενης άσκησης**

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

2. **Ολοκλήρωση λειτουργιών όλων των κατηγοριών χρηστών (servlet)**

   2.1. Λειτουργίες που αφορούν όλες τις κατηγορίες χρηστών:

   2.1.1. Σύνδεση (login): Κάθε χρήστης θα πρέπει να συνδέεται με το μοναδικό username-password. Το password θα διατηρείται στη βάση salted+hashed. (Μπορείτε να δείτε ενδεικτικά βοηθητικές συναρτήσεις για τη διαδικασία παραγωγής salt και για τη χρήση συναρτήσεων hash, στο παράδειγμα 06-b στη σελίδα του μαθήματος).

   2.1.2. Παρακολούθηση συνόδου (session management): Εάν η σύνδεση είναι επιτυχής, θα πρέπει η εφαρμογή σας να διατηρεί πληροφορία που αφορά τη σύνοδο του συγκεκριμένου χρήστη , από τη στιγμή της σύνδεσης μέχρι την αποσύνδεση του χρήστη. Πληροφορία που μπορεί να διατηρείται στο session είναι ενδεικτικά το username και ο ρόλος του εκάστοτε χρήστη. Η πληροφορία συνόδου θα πρέπει να "ακολουθεί" τον χρήστη κατά την πλοήγησή του στην εφαρμογή, μέχρι την αποσύνδεσή του.

   2.1.3. Αποσύνδεση (logout): Κάθε χρήστης θα πρέπει να αποσυνδέεται με ασφάλεια από την εφαρμογή. Κατά την αποσύνδεση να υλοποιήσετε τον καθαρισμό της cache και την ακύρωση του session (invalidate session).

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

   2.2.1. Λειτουργίες Πωλητών (Seller): Οι Πωλητές θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: καταχώρηση νέου πελάτη, έκδοση λογαριασμού πελατών, αλλαγή προγράμματος πελάτη (Σημείωση: μέρος των λειτουργιών έχει υλοποιηθεί από την προηγούμενη εργασία)

   2.2.2. Λειτουργίες Πελατών (Client): Οι Πελάτες θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: μέθοδοι όπως προβολή λογαριασμού, προβολή ιστορικού κλήσεων, εξόφληση λογαριασμού.

2.2.3. Λειτουργίες Διαχειριστή (Administrator). Οι Διαχειριστές θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: δημιουργία νέου πωλητή, δημιουργία νέων προγραμμάτων, αλλαγή χαρακτηριστικών προγράμματος (π.χ. χρέωση παγίου)

3. Ολοκλήρωση επιπέδου Δεδομένων

3.1. Μπορείτε να προβείτε σε όποιες τροποποιήσεις θεωρείτε απαραίτητες στη βάση δεδομένων που έχετε δημιουργήσει από την προηγούμενη άσκηση. 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.

Οδηγίες:

- Ισχύουν οι ίδιες ομάδες και οι ίδιες οδηγίες με τις προηγούμενες εργασίες.
- Το συνολικό παραδοτέο θα περιλαμβάνει σε ένα συμπιεσμένο αρχείο: (α) το project, (β) τη βάση δεδομένων (.sql ή .mwb αρχείο εάν χρησιμοποιείτε το Workbench) και (γ) την τεκμηρίωση της εργασίας.

Table of Contents

Table of Images

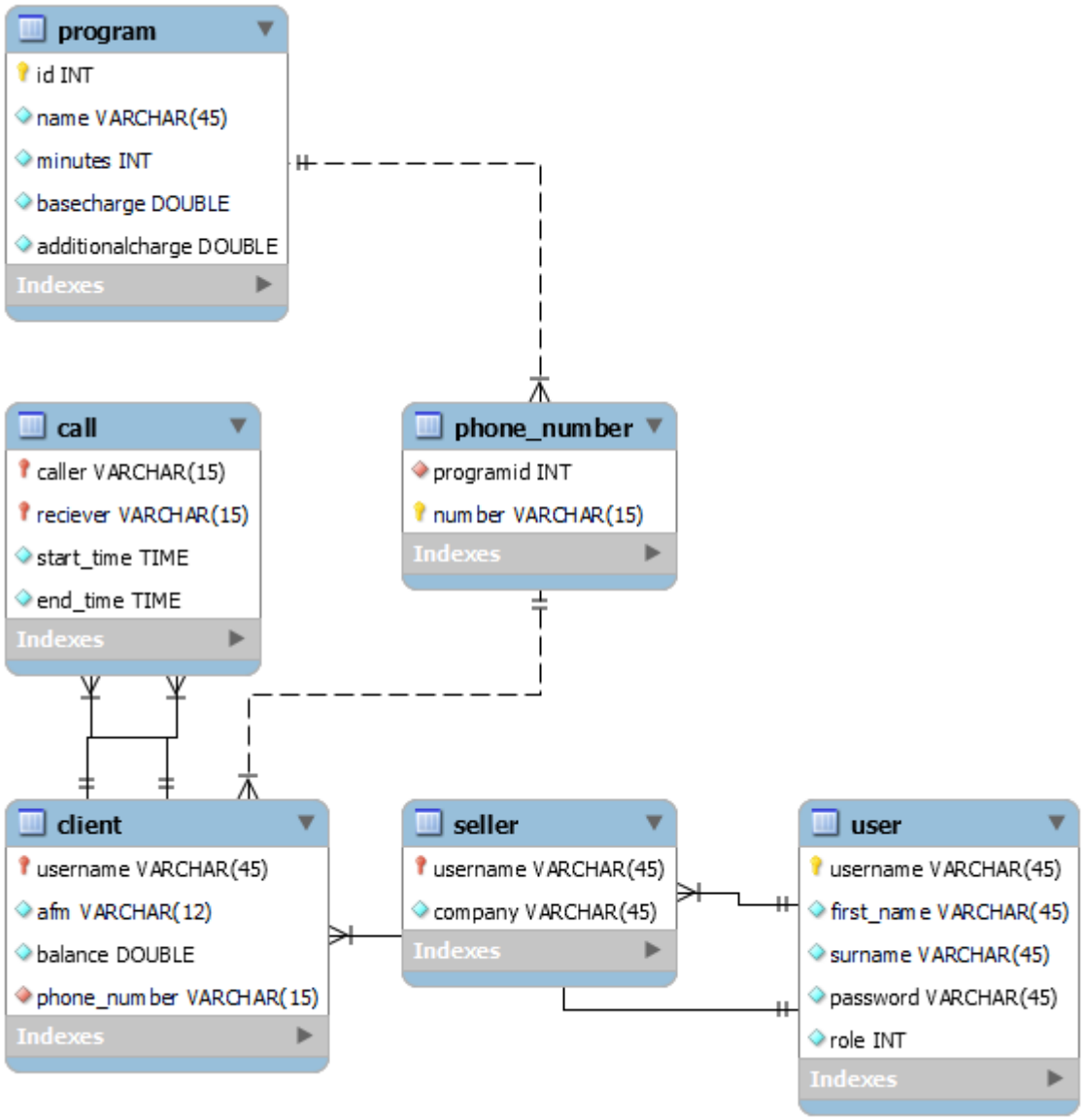
Γενική Περιγραφή της λύσης

1.1 Αρχιτεκτονική Εφαρμογής

1.1.1 Περιγραφή 3-tier Αρχιτεκτονικής

- Επίπεδο Παρουσίασης (Presentation Layer):
  - **Ρόλος**: Αυτό το επίπεδο είναι υπεύθυνο για την αλληλεπίδραση με τον χρήστη. Παρέχει την διεπαφή χρήστη (UI) και συλλέγει τα δεδομένα εισόδου από τον χρήστη.
  - **Τεχνολογίες**: CSS, Java, JSP (JavaServer Pages).
  - **Περιγραφή**:
    - Οι σελίδες JSP λαμβάνουν τα δεδομένα από τον χρήστη μέσω φορμών και τις στέλνουν στα servlets για επεξεργασία.
    - Τα δεδομένα που επιστρέφουν από τα servlets παρουσιάζονται στον χρήστη μέσω δυναμικών JSP σελίδων.
    - Επίπεδο Λογικής (Logic Layer)
- Επίπεδο Λογικής (Logic Layer):
  - **Ρόλος**: Αυτό το επίπεδο περιλαμβάνει την επιχειρηματική λογική της εφαρμογής. Επεξεργάζεται τα δεδομένα που λαμβάνει από το επίπεδο παρουσίασης και διαχειρίζεται την αλληλεπίδραση με το επίπεδο δεδομένων.
  - **Τεχνολογίες**: Java Servlets, Ενδιάμεσες κλάσεις λογικής.
  - **Περιγραφή**:
    - Τα Servlets δέχονται αιτήσεις από τις JSP σελίδες, επεξεργάζονται τα δεδομένα και αλληλεπιδρούν με τη βάση δεδομένων μέσω JDBC.
    - Οι ενδιάμεσες κλάσεις λογικής (business logic classes) υλοποιούν τις λειτουργίες που απαιτούνται από την εφαρμογή (π.χ., έλεγχος πιστοποίησης χρηστών, διαχείριση λογαριασμών, κλπ).
- Επίπεδο Δεδομένων (Data Layer):
  - **Ρόλος**: Αυτό το επίπεδο είναι υπεύθυνο για την αποθήκευση και ανάκτηση των δεδομένων. Περιλαμβάνει τη βάση δεδομένων και τη διαχείριση της αλληλεπίδρασης με αυτήν.
  - **Τεχνολογίες**: MySQL, JDBC (Java Database Connectivity), apache-tomcat-9.0.90, Eclipse.
  - **Περιγραφή**:
    - Η βάση δεδομένων MySQL αποθηκεύει τα δεδομένα της εφαρμογής (π.χ., στοιχεία χρηστών, λογαριασμούς, κλήσεις, κλπ).
    - Τα δεδομένα που επιστρέφουν από τα servlets παρουσιάζονται στον χρήστη μέσω δυναμικών JSP σελίδων.
    - Το JDBC χρησιμοποιείται για την αλληλεπίδραση με τη βάση δεδομένων από τα Servlets και τις ενδιάμεσες κλάσεις λογικής)

1.2 Βάση Δεδομένων(MySQL Workbench)



Εικόνα 1.Το Σχήμα της Βάσης

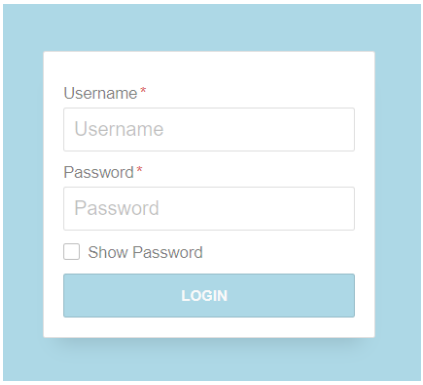
Βλέπε αρχεία Database1.mwb & mobilemanagementdb.sql

Υλοποίηση Λειτουργιών

2.1 Κοινές Λειτουργίες

2.1.1 Σύνδεση (login).

- Εδώ μπορεί να συνδεθεί ο Χρήστης και ανάλογα με τον ρόλο του θα έχει τα αντίστοιχα δικαιώματα, χάρη στην στήλη role του πίνακα user (1=Admin, 2=Client, 3=Seller)



- Χρησιμοποιούμε την login(HttpServletRequest request, HttpServletResponse response), όπου διαχειρίζεται το αίτημα σύνδεσης, καλεί την UserDao(login(String username, String password)) για την επαλήθευση των στοιχείων και ανακατευθύνει τον χρήστη στη σωστή σελίδα ανάλογα με τον ρόλο του.Παρακάτω φαινεται για παράδειγμα η συνδεση του client και επειτα η αντίστοιχη ανακατευθυνση.

```
HttpSession session = request.getSession();
session.setAttribute("username", username);
RequestDispatcher dispatcher=
request.getRequestDispatcher("ClientMain.jsp");
```

2.1.2 Παρακολούθηση Συνόδου (session management)

Κατά την είσοδο του χρήστη, το username αποθηκεύεται στο session:

```
HttpSession session = request.getSession();
session.setAttribute("username", username);
```

Μπορεί να κληθεί με τον παρακάτω τρόπο για αποθηκευση δεδομένων απο την JSP

```
String username = (String) session.getAttribute("username");
```

2.1.3 Αποσύνδεση (logout).

Αν υπάρχει session τότε το ακυρώνει:

```
HttpSession session = request.getSession(false);
              if (session != null) {
                  session.invalidate(); // Invalidate the session
              }
```

Και ανακατευθύνει:

```
RequestDispatcher dispatcher = request.getRequestDispatcher("index.jsp");
dispatcher.forward(request, response);
```

2.2 Λειτουργίες Ανά Κατηγορία Χρήστη

2.2.1 Πωλητές (Seller)

Πρώτα απ'ολα η εγγραφή νέου πωλητή

```
private void register(HttpServletRequest request, HttpServletResponse response)
throws SQLException, IOException, ServletException {
            String username = request.getParameter("username");
            String name = request.getParameter("name");
            String surname = request.getParameter("surname");
            String password = request.getParameter("password");
            String company = request.getParameter("company");
            int role = 3;
            Seller newSeller = new Seller(username, name, surname, password,
role, company);
            sellerDao.insertSeller(newSeller);
            RequestDispatcher dispatcher =
request.getRequestDispatcher("index.jsp");
          dispatcher.forward(request, response);
      }
```

Οι πωλητές μπορούν να εισάγουν νέους πελάτες στο σύστημα:

```
private void insertNewClient(HttpServletRequest request, HttpServletResponse
response) throws SQLException, IOException, ServletException {
            String username = request.getParameter("username");
            String name = request.getParameter("first_name"); // Correct
parameter name
            String surname = request.getParameter("surname");
            String password = request.getParameter("password");
            String AFM = request.getParameter("afm");
            Double balance = 0.0; // Initialize balance to 0.0 as it's not
provided in the form
            int role = 2;
            String phone_number = request.getParameter("phone_number");

            Client newClient = new Client(username, name, surname, password,
role, AFM, balance, phone_number);

            clientDao.insertClient(newClient);
            RequestDispatcher dispatcher =
request.getRequestDispatcher("SellerMain.jsp");
          dispatcher.forward(request, response);
      }
```

Οι πωλητές μπορούν να προβάλουν τα διαθέσιμα προγράμματα κινητής τηλεφωνίας που προσφέρει η εταιρεία:

```
private void display_programs(HttpServletRequest request, HttpServletResponse
response) throws SQLException, IOException, ServletException {
          ArrayList<Program> programs = programDao.getPrograms();
          request.setAttribute("programs", programs);
          request.getRequestDispatcher("/ShowPrograms.jsp").forward(request,
response);
      }
```

Οι πωλητές μπορούν να αντιστοιχίσουν έναν πελάτη σε ένα συγκεκριμένο πρόγραμμα κινητής τηλεφωνίας. Η λειτουργία αυτή επιτρέπει την αλλαγή του προγράμματος ενός πελάτη:

```
private void matchClientToProgram(HttpServletRequest request, HttpServletResponse
response)
                    throws SQLException, IOException, ServletException {
                String clientAFM = request.getParameter("client");
                int programId = Integer.parseInt(request.getParameter("program"));

                // Change the program for the client
                clientDao.changeProgram(clientAFM, programId);

                // Redirect back to a success page or the form page with a success
message
                response.sendRedirect("SellerMain.jsp");
            }
```

Οι πωλητές μπορούν να εκδώσουν τιμολόγια για τους πελάτες τους, παρέχοντας μια λίστα με όλα τα τιμολόγια ενός συγκεκριμένου πελάτη:

```
private void generateInvoice(HttpServletRequest request, HttpServletResponse
response)
                    throws SQLException, IOException, ServletException {
                String username = request.getParameter("username");
                List<Bill> bills = billDao.getCustomerBills(username);
                request.setAttribute("bills", bills);
                RequestDispatcher dispatcher =
request.getRequestDispatcher("Invoice.jsp");
                dispatcher.forward(request, response);
            }
```

Οι πωλητές μπορούν να δουν μια λίστα με όλους τους πελάτες και τα προγράμματα που είναι διαθέσιμα:

```
private void listClientsAndPrograms(HttpServletRequest request,
HttpServletResponse response)
                    throws SQLException, IOException, ServletException {
                List<Client> clients = clientDao.getClients();
                List<Program> programs = programDao.getPrograms();
                request.setAttribute("clients", clients);
                request.setAttribute("programs", programs);
                RequestDispatcher dispatcher =
request.getRequestDispatcher("MatchClient.jsp");
                dispatcher.forward(request, response);
            }
```

2.2.2 Πελάτες (Client)

Η λειτουργία αυτή επιτρέπει την εγγραφή ενός νέου πελάτη στο σύστημα:

```
private void registerClient(HttpServletRequest request, HttpServletResponse
response) throws SQLException, IOException, ServletException {
                String username = request.getParameter("username");
                String name = request.getParameter("first_name"); // Correct
parameter name
                String surname = request.getParameter("surname");
                String password = request.getParameter("password");
                String AFM = request.getParameter("afm");
                Double balance = 0.0; // Initialize balance to 0.0 as it's not
provided in the form
                PhoneNumber phoneNumber = new
PhoneNumber(request.getParameter("phone_number"), null); // Correct parameter name
                int role = 2;
                Client newClient = new Client(username, name, surname, password,
role, AFM, balance, phoneNumber);
                clientDao.insertClient(newClient);
```

```
            RequestDispatcher dispatcher =
request.getRequestDispatcher("index.jsp");
          dispatcher.forward(request, response);
      }
```

Οι πελάτες μπορούν να δουν τον λογαριασμό τους:

```
private void viewAccount(HttpServletRequest request, HttpServletResponse response)
                  throws SQLException, IOException, ServletException {
            HttpSession session = request.getSession(false);
            if (session == null || session.getAttribute("username") == null) {
                  response.sendRedirect("login.jsp");
                  return;
            }
            String username = (String) session.getAttribute("username");
            Bill bill = billDao.selectBillByUsername(username); // Fetch the
bill based on the username
            request.setAttribute("bill", bill);
            RequestDispatcher dispatcher =
request.getRequestDispatcher("Account.jsp"); // Forward to account.jsp
            dispatcher.forward(request, response);
      }
```

Οι πελάτες μπορούν να πληρώσουν τον λογαριασμό τους μέσω της εφαρμογής. Μετά την πληρωμή, ο λογαριασμός αφαιρείται από τη βάση δεδομένων:

```
private void payBill(HttpServletRequest request, HttpServletResponse response)
                  throws SQLException, IOException, ServletException {
            String billId = request.getParameter("bill_id");
            billDao.payBill(billId);
            billDao.deleteBill(billId); // Remove the bill after payment
            response.sendRedirect("Account.jsp");
      }
```

Οι πελάτες μπορούν να δουν το ιστορικό των κλήσεών τους. Η λειτουργία αυτή προωθεί τον πελάτη σε μια σελίδα που εμφανίζει το ιστορικό των κλήσεων:

```
private void display_call_history(HttpServletRequest request, HttpServletResponse
response) throws SQLException, IOException, ServletException{
            RequestDispatcher dispatcher =
request.getRequestDispatcher("CallHistory.jsp");
            dispatcher.forward(request, response);
      }
```

Οι πελάτες μπορούν να δουν το τρέχον υπόλοιπο του λογαριασμού τους και να το ενημερώσουν:

```
private void display_balance(HttpServletRequest request, HttpServletResponse
response) throws SQLException, IOException, ServletException {
            HttpSession session = request.getSession(false);
            if (session == null || session.getAttribute("username") == null) {
                  response.sendRedirect("login.jsp");
                  return;
            }
            String username = (String) session.getAttribute("username");
            request.setAttribute("balance", clientDao.getBallance(username));
            RequestDispatcher dispatcher =
request.getRequestDispatcher("balance.jsp");
            dispatcher.forward(request, response);
      }

      private void set_balance(HttpServletRequest request, HttpServletResponse
response) throws SQLException, IOException, ServletException {
            HttpSession session = request.getSession(false);
```

```
            if (session == null || session.getAttribute("username") == null) {
                  response.sendRedirect("login.jsp");
                  return;
            }
            String username = (String) session.getAttribute("username");
            clientDao.setBalance(username,Double.parseDouble((String)
request.getAttribute("balance")));
            RequestDispatcher dispatcher =
request.getRequestDispatcher("balance.jsp");
            dispatcher.forward(request, response);
      }
```

2.2.3 Διαχειριστής (Administrator)

Η λειτουργία αυτή επιτρέπει την προσθήκη νέων διαχειριστών στο σύστημα:

```
private void insertAdmin(HttpServletRequest request, HttpServletResponse response)
                  throws SQLException, IOException {
            String username = request.getParameter("name");
            String name = request.getParameter("name");
            String surname = request.getParameter("name");
            String password = request.getParameter("name");
            int role = 1;
            Admin newAdmin = new Admin(username, name, surname, password, role);
            adminDao.insertAdmin(newAdmin);
            response.sendRedirect("AdminPage.jsp");
      }
```

Η λειτουργία αυτή επιτρέπει την προσθήκη νέων πελατών στο σύστημα:

```
private void insertClient(HttpServletRequest request, HttpServletResponse
response)
                  throws SQLException, IOException {
            String username = request.getParameter("name");
            String name = request.getParameter("name");
            String surname = request.getParameter("name");
            String password = request.getParameter("name");
            int role = 2;
            String AFM = request.getParameter("AFM");
            Double balance =
Double.parseDouble(request.getParameter("balance"));
            PhoneNumber PhoneNumber = new
PhoneNumber(request.getParameter("phonenumber"), null);
            Client newClient = new Client(username, name, surname, password,
role, AFM, balance, PhoneNumber);
            clientDao.insertClient(newClient);
            response.sendRedirect("AdminPage.jsp");
      }
```

Η λειτουργία αυτή επιτρέπει την προσθήκη νέων πωλητών στο σύστημα:

```
private void insertSeller(HttpServletRequest request, HttpServletResponse
response)
                  throws SQLException, IOException {
            String username = request.getParameter("name");
            String name = request.getParameter("name");
            String surname = request.getParameter("name");
            String password = request.getParameter("name");
            int role = 3;
            String company = request.getParameter("company");
            Seller newSeller = new Seller(username, name, surname, password,
role, company);
```

```
            sellerDao.insertSeller(newSeller);
            response.sendRedirect("AdminPage.jsp");
        }
```

Η λειτουργία αυτή επιτρέπει την διαγραφή χρηστών από το σύστημα:

```
private void deleteUser(HttpServletRequest request, HttpServletResponse response)
                throws SQLException, IOException {
            String username = request.getParameter("username");
            userDao.deleteUser(username);
            response.sendRedirect("ShowUsers.jsp");
        }
```

Η λειτουργία αυτή επιτρέπει την προσθήκη νέων αριθμών τηλεφώνου στο σύστημα:

```
private void insertPhoneNumber(HttpServletRequest request, HttpServletResponse
response)
                throws SQLException, IOException {
            String phoneNumber = request.getParameter("phonenumber");
            int programId = Integer.parseInt(request.getParameter("programId"));
            Program program = programDao.getProgramById(programId);
            phoneNumberDao.insertNumber(phoneNumber, program);
            response.sendRedirect("AdminPage.jsp");
        }
```

Η λειτουργία αυτή επιτρέπει την διαγραφή αριθμών τηλεφώνου από το σύστημα:

```
private void deletePhoneNumber(HttpServletRequest request, HttpServletResponse
response)
                throws SQLException, IOException {
            String phoneNumber = request.getParameter("phonenumber");
            phoneNumberDao.deleteNumber(phoneNumber);
            response.sendRedirect("AdminPage.jsp");
        }
```

Η λειτουργία αυτή επιτρέπει την προσθήκη νέων προγραμμάτων κινητής τηλεφωνίας στο σύστημα:

```
private void insertProgram(HttpServletRequest request, HttpServletResponse
response)
                throws SQLException, IOException  {
            int id = Integer.parseInt(request.getParameter("id"));
            String name = request.getParameter("name");
            int minutes = Integer.parseInt(request.getParameter("minutes"));
            double baseCharge =
Double.parseDouble(request.getParameter("baseCharge"));
            double additionalCharge =
Double.parseDouble(request.getParameter("additionalCharge"));
            Program newProgram = new Program(id, name, minutes, baseCharge,
additionalCharge);
            programDao.insertProgram(newProgram);
            response.sendRedirect("AdminPage.jsp");
        }
```

Η λειτουργία αυτή επιτρέπει την επεξεργασία των στοιχείων ενός υπάρχοντος προγράμματος κινητής τηλεφωνίας:

```
private void editProgram(HttpServletRequest request, HttpServletResponse response)
                throws SQLException, IOException {
            int id = Integer.parseInt(request.getParameter("id"));
          double baseCharge =
Double.parseDouble(request.getParameter("baseCharge"));
          // Fetch the existing program
          Program program = programDao.getProgramById(id);
          // Update the base charge
          program.setBaseCharge(baseCharge);
```

```
            // Persist the updated program
            programDao.editProgram(program);
            response.sendRedirect("EditPrograms.jsp");
        }
```

Κώδικας Προγράμματος

3.1 Userdao.java

Η κλάση αυτή αποτελεί ένα βασικό μέρος της εφαρμογής μας, είναι υπεύθυνη για την πρόσβαση και διαχείριση των δεδομένων των χρηστών στη βάση δεδομένων. Χρησιμοποιεί τη βιβλιοθήκη JDBC (Java Database Connectivity) για να δημιουργήσει μια σύνδεση με τη βάση δεδομένων MySQL και να εκτελέσει εντολές SQL. Η κλάση περιλαμβάνει τη μέθοδο getConnection(), η οποία δημιουργεί και επιστρέφει μια σύνδεση στη βάση δεδομένων.

Επιπλέον, η κλάση UserDao περιέχει τη μέθοδο login(), η οποία επαληθεύει τα στοιχεία σύνδεσης του χρήστη. Αυτή η μέθοδος χρησιμοποιεί δύο PreparedStatement για να εκτελέσει SQL εντολές που ελέγχουν αν το όνομα χρήστη και ο κωδικός πρόσβασης είναι έγκυρα και ανήκουν σε έναν εγγεγραμμένο χρήστη. Αν ο χρήστης βρεθεί στη βάση δεδομένων, επιστρέφει τον ρόλο του χρήστη, ο οποίος καθορίζει τα δικαιώματα και τις δυνατότητές του στην εφαρμογή. Σε περίπτωση που τα στοιχεία είναι λανθασμένα, επιστρέφει κατάλληλες τιμές που δείχνουν το είδος του σφάλματος. Με αυτόν τον τρόπο, η UserDao διασφαλίζει την σωστή διαχείριση των δεδομένων των χρηστών.

3.2 Sellerdao.java

Η κλάση αυτή είναι υπεύθυνη για τη διαχείριση των δεδομένων των πωλητών. Αξιοποιώντας το JDBC, η κλάση αυτή παρέχει λειτουργικότητα για την εισαγωγή νέων πωλητών στη βάση δεδομένων, καθώς και για την ανάκτηση των πληροφοριών τους. Η μέθοδος getConnection() δημιουργεί και επιστρέφει μια σύνδεση στη βάση δεδομένων MySQL. Η μέθοδος insertSeller(Seller seller) χρησιμοποιεί SQL εντολές για την εισαγωγή ενός νέου πωλητή τόσο στον πίνακα user όσο και στον πίνακα seller, εξασφαλίζοντας ότι τα δεδομένα του χρήστη και της εταιρείας του εισάγονται σωστά.

Επιπλέον, η μέθοδος setSeller(String username) μπορεί να ανακτησει δεδομένα ενός πωλητή με βάση το όνομα χρήστη του. Η μέθοδος αυτή εκτελεί δύο ξεχωριστές SQL ερωτήσεις για να ανακτήσει αρχικά τα βασικά στοιχεία του χρήστη από τον πίνακα user και στη συνέχεια τα στοιχεία της εταιρείας από τον πίνακα seller. Αν βρεθούν αντίστοιχες εγγραφές, δημιουργείται και επιστρέφεται ένα αντικείμενο Seller με όλες τις σχετικές πληροφορίες. Εάν δεν βρεθεί ο χρήστης ή ο πωλητής, η μέθοδος επιστρέφει null και καταγράφει το αντίστοιχο μήνυμα στο σύστημα.

3.3 Clientdao.java

Η κλάση αυτή είναι υπεύθυνη για τη διαχείριση των δεδομένων των πελατών. Περιλαμβάνει λειτουργίες για την εισαγωγή νέων πελατών, την ανάκτηση πληροφοριών

πελατών με βάση το όνομα χρήστη και την απόκτηση όλων των πελατών από τη βάση δεδομένων. Η μέθοδος insertClient(Client client) είναι υπεύθυνη για την εισαγωγή ενός νέου πελάτη στη βάση δεδομένων. Χρησιμοποιεί πολλαπλές SQL εντολές για να εισάγει τα δεδομένα του χρήστη, του τηλεφωνικού αριθμού και του πελάτη στους αντίστοιχους πίνακες, εξασφαλίζοντας ότι οι σχέσεις μεταξύ των δεδομένων είναι σωστά ορισμένες και ότι τα προγράμματα κινητής τηλεφωνίας υπάρχουν πριν την εισαγωγή των δεδομένων του πελάτη.

Η μέθοδος setClient(String username) ανακτά τα δεδομένα ενός πελάτη με βάση το όνομα χρήστη του. Η μέθοδος εκτελεί πολλαπλές SQL ερωτήσεις για να ανακτήσει τις πληροφορίες του χρήστη, του πελάτη, του τηλεφωνικού αριθμού και του προγράμματος. Αν βρεθούν αντίστοιχες εγγραφές, δημιουργείται και επιστρέφεται ένα αντικείμενο Client με όλες τις σχετικές πληροφορίες. Εάν δεν βρεθεί ο χρήστης ή ο πελάτης, η μέθοδος επιστρέφει null και καταγράφει το αντίστοιχο μήνυμα στο σύστημα. Επιπλέον, η κλάση περιλαμβάνει τη μέθοδο getClients() που επιστρέφει μια λίστα όλων των πελατών από τη βάση δεδομένων, προσφέροντας μια συνολική εικόνα των πελατών που διαχειρίζεται η εφαρμογή.

3.4 Admindao.java

Η κλάση αυτή είναι υπεύθυνη για τη διαχείριση των δεδομένων των διαχειριστών . Χρησιμοποιεί τη βιβλιοθήκη JDBC για να συνδέεται με τη βάση δεδομένων MySQL και να εκτελεί SQL εντολές. Η μέθοδος getConnection() δημιουργεί και επιστρέφει μια σύνδεση στη βάση δεδομένων, φροντίζοντας να φορτώσει τον κατάλληλο driver και να χειριστεί τυχόν εξαιρέσεις που μπορεί να προκύψουν στη διαδικασία σύνδεσης. Η κύρια μέθοδος της κλάσης, insertAdmin(Admin admin), είναι υπεύθυνη για την εισαγωγή ενός νέου διαχειριστή στη βάση δεδομένων. Χρησιμοποιεί την PreparedStatement για να εκτελέσει την εντολή SQL που εισάγει τα δεδομένα του διαχειριστή στον πίνακα users.

Η insertAdmin(Admin admin) εξασφαλίζει ότι τα δεδομένα του διαχειριστή -όπως το όνομα χρήστη, ο κωδικός πρόσβασης, το όνομα, το επώνυμο και ο ρόλος- εισάγονται σωστά στη βάση δεδομένων. Χρησιμοποιεί τη δομή try-catch για να διαχειριστεί τη σύνδεση και το statement, εξασφαλίζοντας ότι θα κλείσουν αυτόματα μετά την εκτέλεση της εντολής. Σε περίπτωση που προκύψει κάποιο σφάλμα κατά την εκτέλεση της SQL εντολής, το σφάλμα καταγράφεται και εμφανίζεται μέσω της κλήσης e.getStackTrace().

3.5 UserServlet.java

Η μέθοδος login στην κλάση UserServlet είναι υπεύθυνη για τη διαχείριση της διαδικασίας σύνδεσης των χρηστών. Κατά την εκτέλεση της μεθόδου, τα στοιχεία σύνδεσης του χρήστη (username και password)λαμβάνονται από το αίτημα HTTP. Στη συνέχεια, η μέθοδος χρησιμοποιεί την UserDao για να επαληθεύσει τα στοιχεία αυτά και να προσδιορίσει τον ρόλο του χρήστη. Ανάλογα με τον ρόλο που επιστρέφεται από την UserDao, η μέθοδος δημιουργεί ένα HttpSession και ανακατευθύνει τον χρήστη στη

σωστή σελίδα: ClientMain.jsp για πελάτες, SellerMain.jsp για πωλητές και AdminMain.jsp για διαχειριστές. Αν ο ρόλος είναι άκυρος ή ο κωδικός πρόσβασης δεν είναι σωστός, η μέθοδος ενημερώνει το session με ένα κατάλληλο μήνυμα σφάλματος και ανακατευθύνει τον χρήστη στη σελίδα σύνδεσης (LoginPage.jsp).

Η χρήση του RequestDispatcher επιτρέπει την μεταβίβαση μεταξύ των σελίδων της εφαρμογής, ενώ η διαχείριση των sessions εξασφαλίζει ότι οι χρήστες παραμένουν συνδεδεμένοι καθ' όλη τη διάρκεια της χρήσης της εφαρμογής. Ετσι διασφαλίζουμε ότι μόνο εγγεγραμένοι χρήστες μπορούν να αποκτήσουν πρόσβαση στις αντίστοιχες σελίδες και λειτουργίες της εφαρμογής.

3.6 SellerServlet.java

Η κλάση αυτή είναι υπεύθυνη για τη διαχείριση των αιτημάτων που σχετίζονται με τους πωλητές στην εφαρμογή . Η κλάση αυτή εκτελεί μια σειρά από ενέργειες ανάλογα με την παράμετρο action που λαμβάνει από το αίτημα HTTP. Οι κύριες λειτουργίες περιλαμβάνουν την εγγραφή νέων πωλητών, την εισαγωγή νέων πελατών, την αντιστοίχιση πελατών με προγράμματα κινητής τηλεφωνίας και την εμφάνιση διαθέσιμων προγραμμάτων. Για παράδειγμα, η μέθοδος insertNewClient δημιουργεί ένα νέο αντικείμενο Client με τα δεδομένα που παρέχονται από το αίτημα και το εισάγει στη βάση δεδομένων χρησιμοποιώντας το ClientDao, ενώ η μέθοδος matchClient μπορεί και ανακτά λίστες πελατών και προγραμμάτων και τις προωθεί στην προβολή για εμφάνιση.

Επιπλέον, η κλάση SellerServlet χρησιμοποιεί ξανά μεθόδους σύνδεσης και αποσύνδεσης από το UserServlet για να διαχειρίζεται τα αιτήματα σύνδεσης και αποσύνδεσης των χρηστών. Η μέθοδος register επιτρέπει την εγγραφή νέων πωλητών, δημιουργώντας ένα νέο αντικείμενο Seller και αποθηκεύοντάς το στη βάση δεδομένων μέσω του SellerDao. Η μέθοδος changeProgram επιτρέπει την αλλαγή του προγράμματος ενός πελάτη, ενημερώνοντας τα δεδομένα του στη βάση δεδομένων

3.7 ClientServlet.java

Η κλάση αυτή διαχειρίζεται τα αιτήματα που σχετίζονται με τους πελάτες. Η κλάση αυτή αρχικοποιεί τα DAO αντικείμενα (ClientDao, BillDao, CallDao, PhoneNumberDao, ProgramDao) στην μέθοδο init(), εξασφαλίζοντας τον σωστό χειρισμό των δεδομένων των πελατών, των κλήσεων, των αριθμών τηλεφώνου και των προγραμμάτων. Στην μέθοδο doPost(), η κλάση χειρίζεται διάφορες ενέργειες όπως η registerClient και η displayCallHistory. Η registerClient δημιουργεί ένα νέο αντικείμενο Client με τα δεδομένα από το αίτημα και το αποθηκεύει στη βάση δεδομένων, ενώ η displayCallHistory ανακτά το ιστορικό κλήσεων ενός πελάτη και το προωθεί στην κατάλληλη σελίδα προβολής.

Επιπλέον, η doGet() χειρίζεται αιτήματα που σχετίζονται με την προβολή της λίστας χρηστών (listUser) και άλλες ενέργειες που μπορεί να χρειαστούν προσθήκη. Κάθε μέθοδος μέσα στην κλάση ClientServlet έχει συγκεκριμένο σκοπό, όπως η προβολή

προγραμμάτων (display_programs), η αλλαγή προγραμμάτων (change_program), η προβολή υπολοίπου (display_balance), και η προβολή λογαριασμών (display_account). Οι μέθοδοι αυτές χρησιμοποιούν το RequestDispatcher για να προωθήσουν τα αιτήματα στις αντίστοιχες σελίδες JSP.

3.8 AdminServlet.java

Η κλάση αυτή είναι υπεύθυνη για τη διαχείριση των αιτημάτων που σχετίζονται με τις λειτουργίες διαχειριστή. Μέσω της μεθόδου doGet, η κλάση χειρίζεται διάφορες ενέργειες όπως την εισαγωγή διαχειριστών, πελατών και πωλητών, την εισαγωγή και διαγραφή τηλεφωνικών αριθμών, καθώς και την εισαγωγή και επεξεργασία προγραμμάτων κινητής τηλεφωνίας. Κάθε ενέργεια καθορίζεται από την παράμετρο action που λαμβάνεται από το αίτημα HTTP. Οι μέθοδοι όπως insertAdmin, insertClient, και insertSeller δημιουργούν τα αντίστοιχα objects (Admin, Client, Seller) με τα δεδομένα που παρέχονται από το αίτημα και τα αποθηκεύουν στη βάση δεδομένων χρησιμοποιώντας τα DAO αντικείμενα.

Επιπλέον, η κλάση AdminServlet περιλαμβάνει μεθόδους για την εισαγωγή, διαγραφή και επεξεργασία τηλεφωνικών αριθμών και προγραμμάτων κινητής τηλεφωνίας. Για παράδειγμα, η μέθοδος insertPhoneNumber εισάγει έναν νέο phonenumber και τον συσχετίζει με ένα πρόγραμμα, ενώ η μέθοδος deletePhoneNumber διαγράφει έναν υπάρχοντα τηλεφωνικό αριθμό από τη βάση δεδομένων. Η μέθοδος insertProgram δημιουργεί ένα νέο πρόγραμμα με τα καθορισμένα χαρακτηριστικά και το αποθηκεύει στη βάση δεδομένων, ενώ η editProgram επιτρέπει την ενημέρωση του βασικού κόστους ενός υπάρχοντος προγράμματος.

3.9 User.java, Seller.java, Client.java, Admin.java

Η κλάσεις αυτές ορίζου τις βασικές ιδιότητες και μεθόδους για έναν χρήστη στην εφαρμογή, όπως το όνομα χρήστη, το όνομα, το επώνυμο, τον κωδικό πρόσβασης και τον ρόλο (client, seller, administrator). Περιλαμβάνουν, όποτε χρειάζονται, getter και setter μεθόδους για κάθε ιδιότητα.

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΠΗΓΕΣ

1. https://www.youtube.com/playlist?list=PLGRDMO4rOGcOjhFoLV2xOfRQqYjpSVhxt
2. https://www.javatpoint.com/servlet-tutorial