ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Σχολή: Τεχνολογιών Πληροφορικής και Επικοινωνιών
Τμήμα: Πληροφορικής
Μάθημα: Προγραμματισμός στο Διαδίκτυο και τον Παγκόσμιο Ιστό

Τεκμηρίωση Τελικής Εργασίας

Στοιχεία φοιτητών ομάδας:

Σπύρος Γιαννικάκης	Π17210
Ελένη Κατσικαντάμη	Π17197
Αθηνά Παπαφιλιππποπούλου	Π15112

ΣΕΠΤΕΜΒΡΙΟΣ 2020

Εκφώνηση **4**

Γενική περιγραφή της λύσης **4**

Εκτέλεση Προγράμματος **5**

Login Page 5

Σελίδες χρηστών 5

Admin 5

Αρχική Σελίδα 5

Register a new Seller 6

Create a new Program 7

Modify an existing Program 8

Seller 9

Αρχική Σελίδα 9

View all Programs 9

Add a new Client 10

Assign a Program to a Client 11

Billing 13

Client 14

Αρχική Σελίδα 14

Bill Payment 15

Κώδικας Προγράμματος **16**

AdministratorDao 16

Μέθοδος Validate() 16

Μέθοδος save() - Εισαγωγή πωλητή 16

Μέθοδος save2() - Εισαγωγή νέου προγράμματος στη βάση 17

Μέθοδος save3() - Αναθεση Προγράμματος σε πελάτη 17

ClientDao 18

Μέθοδος Validate() 18

Μέθοδος showCallHistory() 18

ProgramDao 19

Μέθοδος save() 19

Μέθοδος getAllRecords() - Εμφανίζει όλα τα προγράμματα 20

Μέθοδος AssignProgramToClient() 20

SellerDao 21

Μέθοδος Validate() 21

Μέθοδος ShowClientBill 21

Κλάση ViewProgramName 22

Κλάση SaltedHashed 23

AdminServlet 23

AdminServlet2 25

ClientHistoryServlet 26

SellerLogin 27
SellerLogout 28
SellerServlet 28
SellerShowBill 30

Εκφώνηση

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.

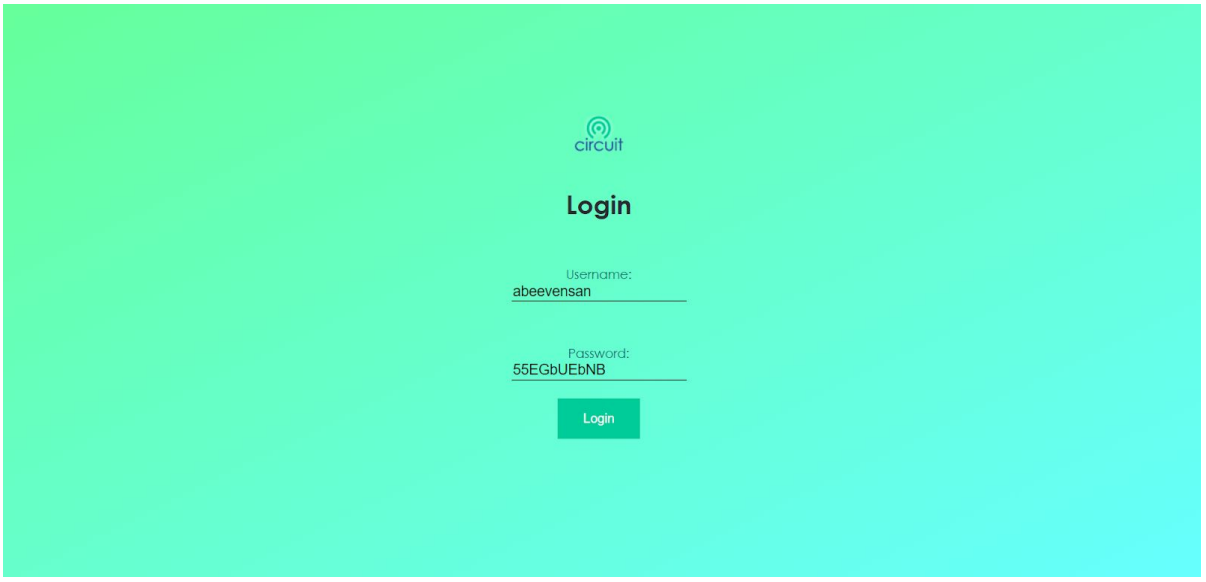
Γενική περιγραφή της λύσης

Στην παρούσα εργασία δημιουργήσαμε ένα Dynamic Web Project χρησιμοποιώντας java, html, css, servlets, database.Την υλοποιήσαμε στο Eclipse IDE for Java Developers. Ο server είναι Tomcat 9.0. Για τη βάση χρησιμοποιήσαμε PostgreSQL. Επεκτείναμε τη λειτουργικότητα του project της Άσκησης 2 καθώς υλοποιήθηκαν οι ζητούμενες λειτουργίες για όλες τις κατηγορίες χρηστών (Admins, Sellers, Clients).

Εκτέλεση Προγράμματος

Login Page

Στην αρχική σελίδα Login ο χρήστης συμπληρώνει στα αντίστοιχα πεδία, το username και το password του.



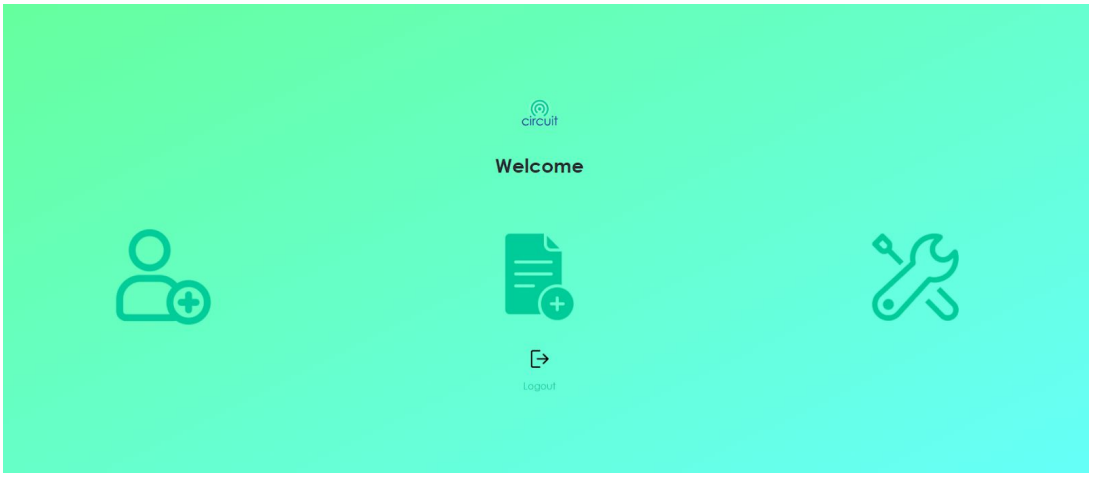
Σελίδες χρηστών

Admin

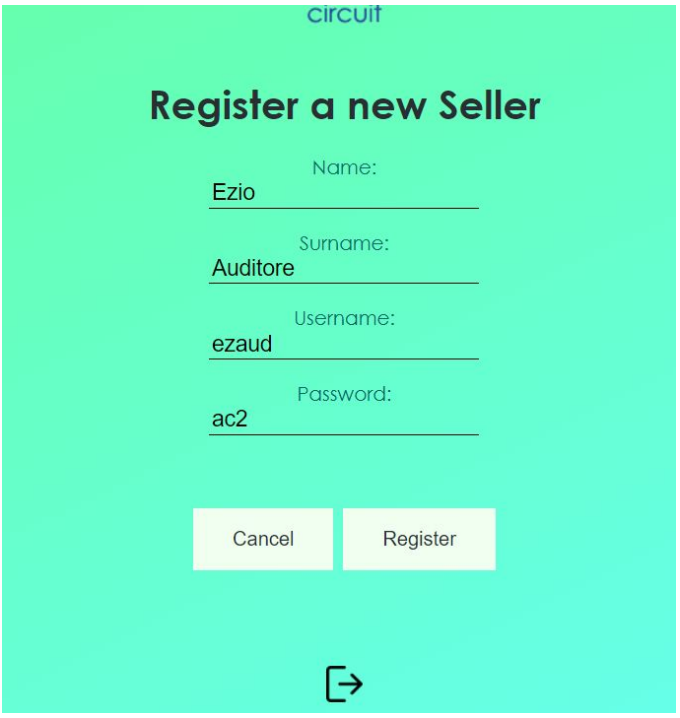
Αρχική Σελίδα

Οι διαχειριστές της σελίδας έχουν τις παρακάτω δυνατότητες:

- Καταχώρηση/ Εγγραφή νέου πωλητή (Register a new Seller)
- Δημιουργία νέου προγράμματος (Create a new Program)
- Τροποποίηση ήδη υπάρχοντος προγράμματος (Modify an existing Program)



Register a new Seller



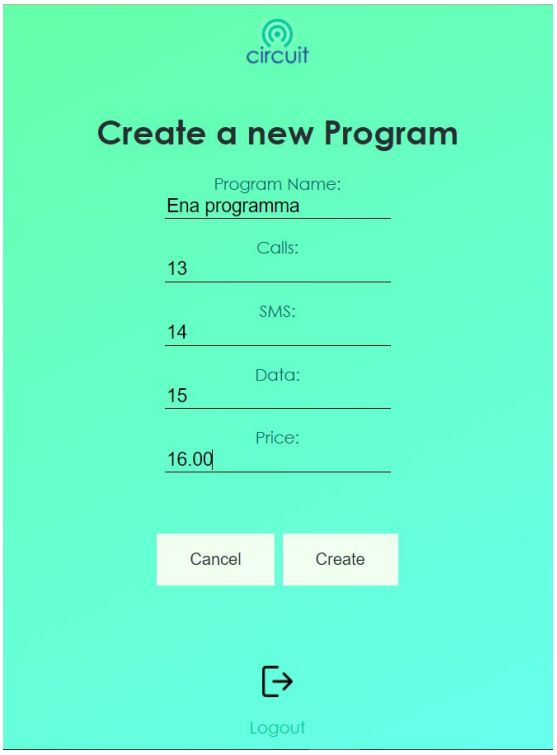
Ο συνδεδεμένος διαχειριστής μπορεί να καταχωρήσει έναν νέο πωλητή συμπληρώνοντας τα πεδία που φαίνονται στην παραπάνω εικόνα (Name, Surname, Username, Password). Πατώντας το κουμπί Register γίνεται η καταχώρηση του πωλητή ενώ πατώντας το κουμπί Cancel ακυρώνεται η διαδικασία.

Data Output Explain Messages Notifications

	username [PK] text	firstname text	lastname text	salt text	hashed_password text
5	ezaud	Ezio	Auditore	C101A0...	F4C70971A41B00A2B30...
6	jbricket4	Jeniece	Bricket	CA1223...	AF78E1E46978960650D...
7	mcutford9	Modesty	Cutford	077E35...	52F107AB7A4A0C202C...
8	mdevitt0	Marlowe	Devitt	E1D23F...	20145E26ABDF380A225...
9	tcollins3	Thomasina	Collins	ECFA9B...	14DC351C1A2FFA1E885...
10	ttesche8	Tannie	Tesche	37BA10...	3BC7DB844BEAB33BD0...
11	vmacewan2	Viole	Macewan	F74145...	D272F1B4CAE0F29FDCE...
12	yamar6	Yvor	Amar	2BF78F...	AACA44D1059FE8AF992...

Όπως φαίνεται και στο παραπάνω στιγμιότυπο ενώ ο κωδικός που συμπλήρωσε ο χρήστης για τη δημιουργία νέου πωλητή ήταν ο ac2, στη βάση διατηρείται salted & hashed.

Create a new Program

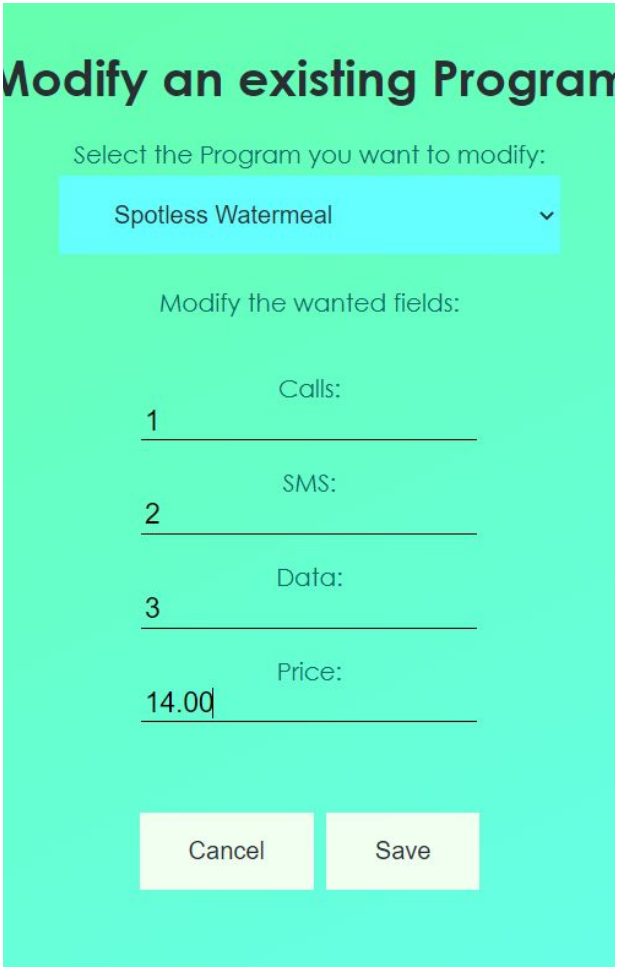


Ο συνδεδεμένος διαχειριστής μπορεί να δημιουργήσει ένα νέο πρόγραμμα συμπληρώνοντας τα πεδία που φαίνονται στην παραπάνω εικόνα (Program Name, Calls, SMS, Data, Price). Πατώντας το κουμπί Create γίνεται η δημιουργία του νέου προγράμματος ενώ πατώντας το κουμπί Cancel ακυρώνεται η διαδικασία.

Data Output | Explain | Messages | Notifications

	pname [PK] text	data integer	sms integer	calls integer	fee numeric
8	Common Eve...	669	195	1081	18.75
9	Deckert's Pin...	1346	194	1828	13.57
10	Dipogon	1796	291	1230	36.77
11	Ena program...	15	14	13	16
12	Finmark's Lec...	1156	112	829	31.05
13	Florida Cacalia	717	193	848	6.35
14	Fuchsia Bego...	1021	116	1526	6.48
15	Golden Prickl...	745	162	1299	24.29

Modify an existing Program



Ο συνδεδεμένος διαχειριστής μπορεί να τροποποιήσει ένα ήδη υπάρχον πρόγραμμα συμπληρώνοντας τα αντίστοιχα πεδία που φαίνονται στην παραπάνω εικόνα. Επιλέγει από

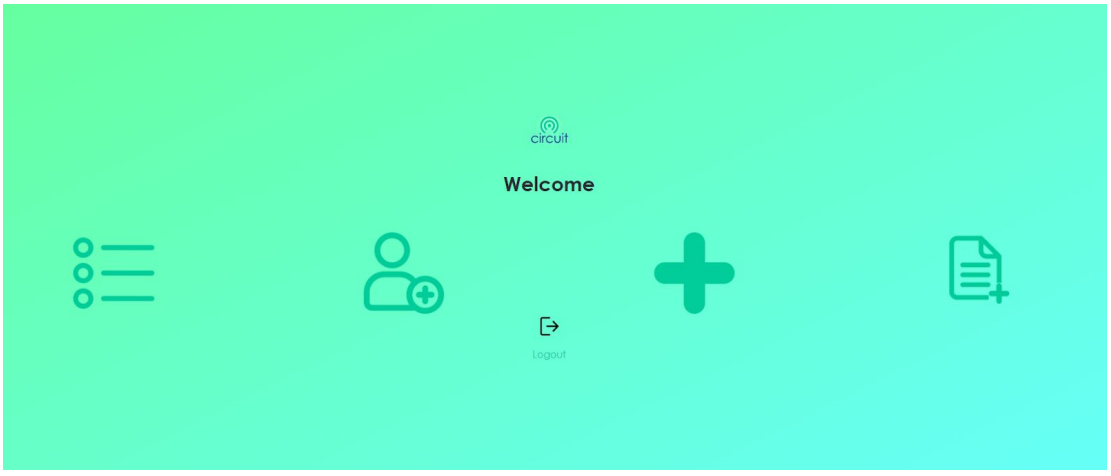
το Dropdown Button το ήδη υπάρχον πρόγραμμα που θέλει να επεξεργαστεί και έπειτα καταχωρεί τα επιθυμητά πεδία (Calls, SMS, Data, Price).
Πατώντας το κουμπί Save γίνεται η αποθήκευση του προγράμματος με τις νέες τροποποιήσεις ενώ πατώντας το κουμπί Cancel ακυρώνεται η διαδικασία.

Seller

Αρχική Σελίδα

Οι πωλητές έχουν τις λειτουργίες που είχαμε υλοποιήσει στην δεύτερη εργασία του μαθήματος και μία ακόμη καινούρια δυνατότητα.
Οι δυνατότητες αυτές είναι οι παρακάτω:

- Προβολή όλων των προγραμμάτων (View all Programs)
- Προσθήκη νέου πελάτη (Add a new Client)
- Ανάθεση προγράμματος σε πελάτη (Assign a Program to a Client)
- Έκδοση λογαριασμού (Billing)



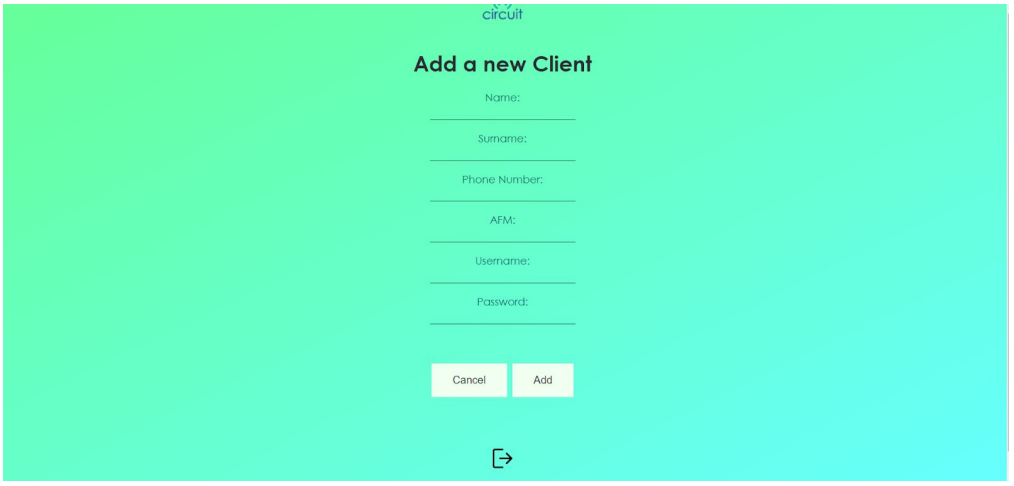
View all Programs

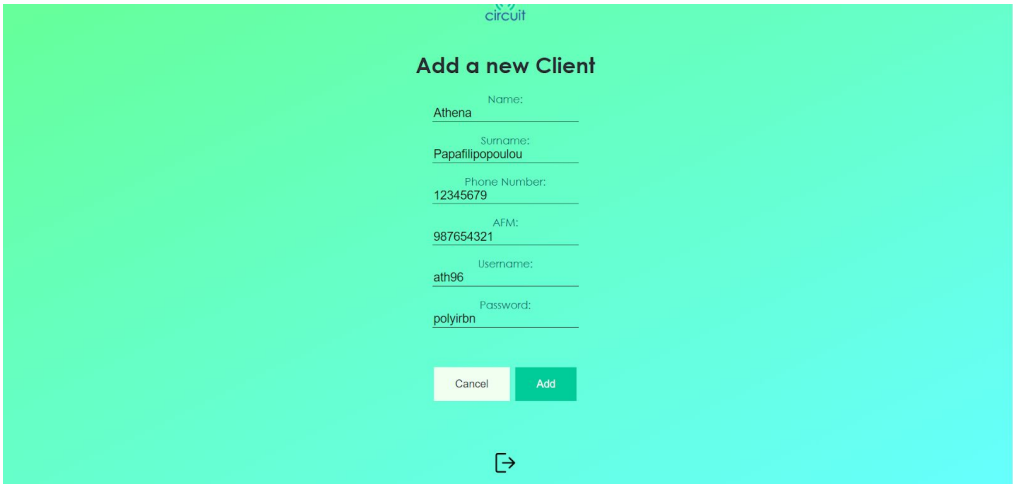
Maclura	1097	163	1046.0	9.05
Cartilage Lichen	736	217	1171.0	35.85
Pandanus	1658	178	1351.0	39.56
Erigeron Pussytoes	1253	211	1756.0	32.44
Texas Holdback	1835	271	1453.0	18.25
Dotted Duckmeat	1835	175	1530.0	39.02
Ballhead Ipomopsis	1532	253	835.0	29.71
Tadpole Buttercup	1281	277	1551.0	23.99
Wideleaf Desmatodon Moss	1516	202	1574.0	20.02
Rosendahl's Golden Saxifrage	1312	168	1902.0	18.67
Ithaca Blackberry	922	228	1736.0	37.13
Sevier Canyon Stickleaf	1448	177	1016.0	23.15
Pancratium	1177	291	1694.0	39.27
Pseudoleskea Moss	1939	298	1903.0	20.94
Wheel Milkweed	1037	124	965.0	21.22
Sevendevils Onion	1468	130	1472.0	15.48
Cup Lichen	1359	216	1605.0	36.05
African Clover	1369	298	1917.0	28.48
Coralbean	1300	241	1838.0	35.24
Heartleaf Mock Orange	1792	245	1082.0	33.04

Όταν ο χρήστης επιλέξει την πρώτη λειτουργία για προβολή όλων των προγραμμάτων, θα μπορεί να δει τα διαθέσιμα προγράμματα διατεταγμένα σε πίνακα όπως φαίνεται στο

παραπάνω στιγμιότυπο. Οι στήλες περιέχουν το όνομα του προγράμματος, τα λεπτά ομιλίας που προσφέρει, τα γραπτά μηνύματα που προσφέρει, τα δεδομένα για χρήση διαδικτύου και την τιμή του αντίστοιχου προγράμματος. Πατώντας το μαύρο εικονίδιο, ο χρήστης κάνει Logout από την εφαρμογή.

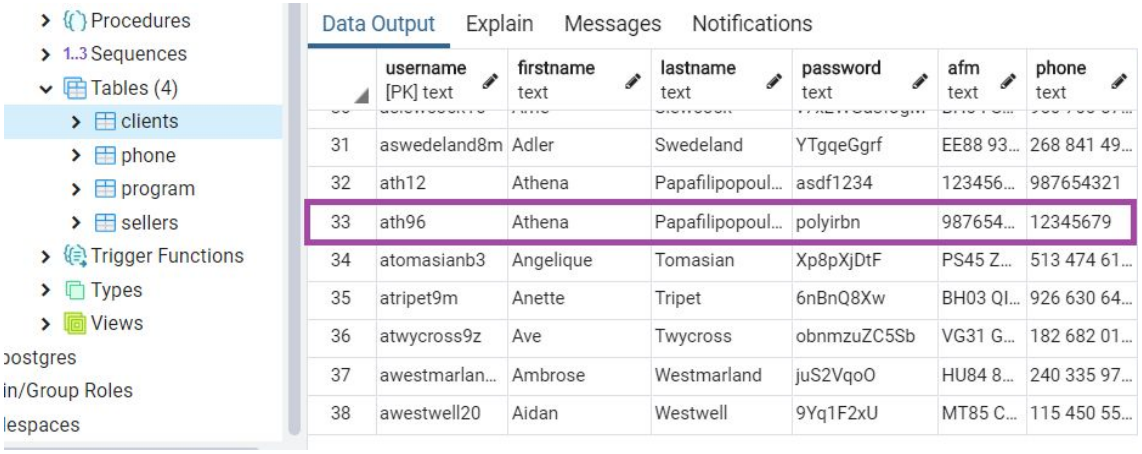
Add a new Client





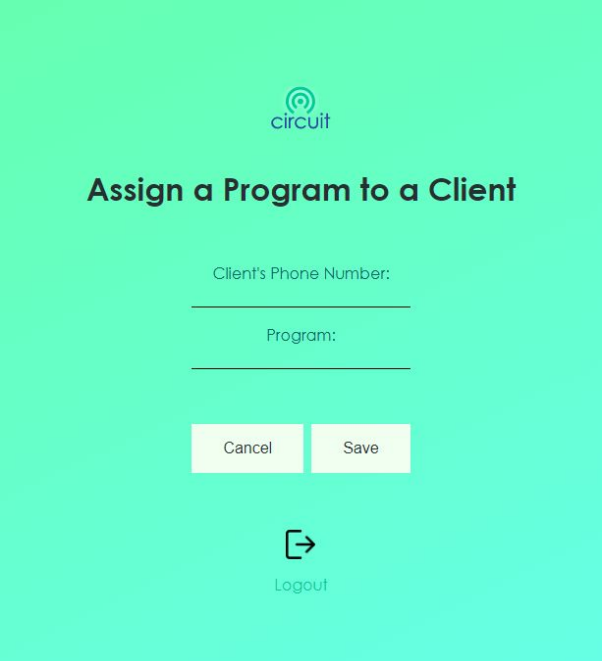
Ο χρήστης συμπληρώνοντας όλα τα πεδία της φόρμας μπορεί να εισάγει έναν νέο πελάτη στη βάση.

Για παράδειγμα, προσθέσαμε τον πελάτη με username ath96 και κωδικό polyirbn. Όπως φαίνεται στο παρακάτω στιγμιότυπο της βάσης, ο νέος πελάτης έχει καταχωρηθεί επιτυχώς.

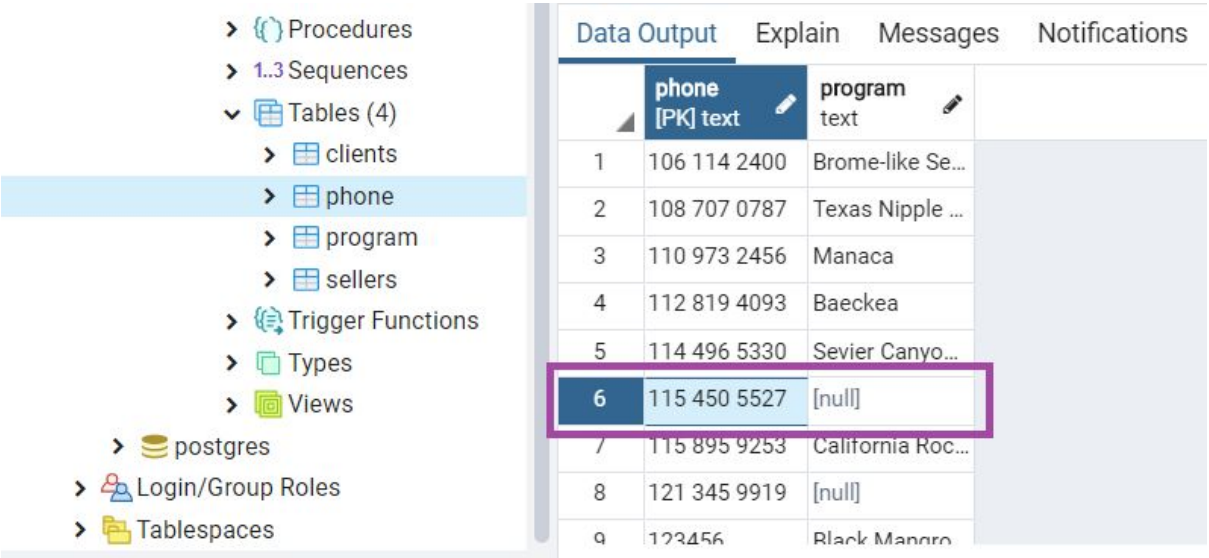


	username [PK] text	firstname text	lastname text	password text	afm text	phone text
31	aswedeland8m	Adler	Swedeland	YTgqeGgrf	EE88 93...	268 841 49...
32	ath12	Athena	Papafilipopoul...	asdf1234	123456...	987654321
33	ath96	Athena	Papafilipopoul...	polyirbn	987654...	12345679
34	atomasianb3	Angelique	Tomasian	Xp8pXjDtF	PS45 Z...	513 474 61...
35	atripet9m	Anette	Tripet	6nBnQ8Xw	BH03 QI...	926 630 64...
36	atwycross9z	Ave	Twycross	obnmzuZC5Sb	VG31 G...	182 682 01...
37	awestmarlan...	Ambrose	Westmarland	juS2VqoO	HU84 8...	240 335 97...
38	awestwell20	Aidan	Westwell	9Yq1F2xU	MT85 C...	115 450 55...

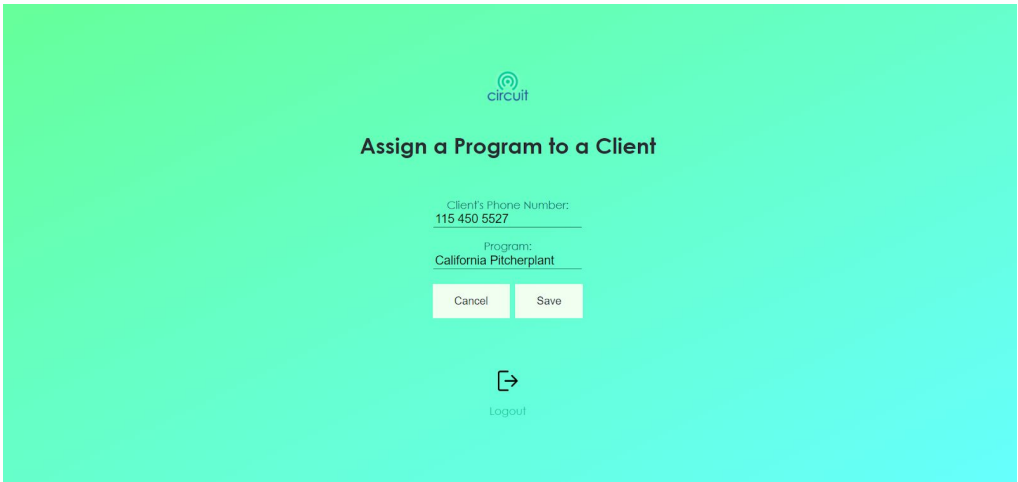
Assign a Program to a Client



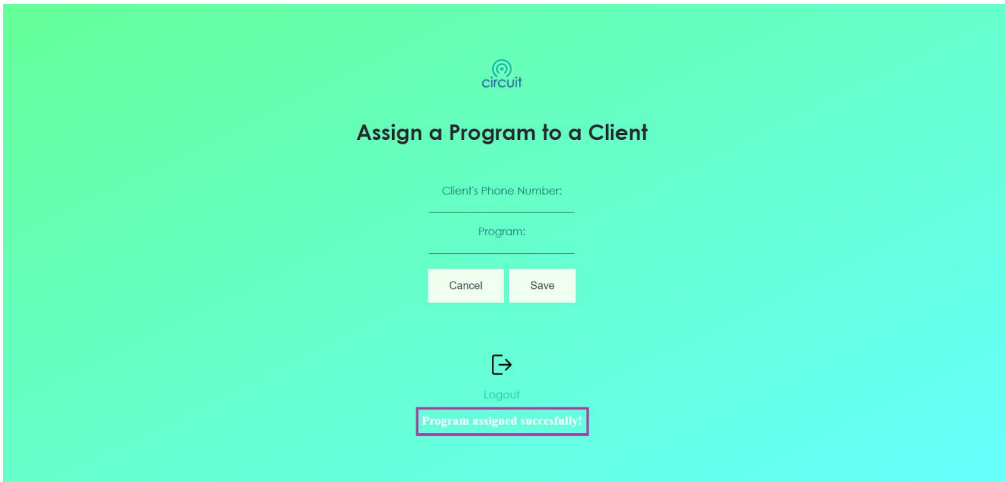
Ο χρήστης συμπληρώνοντας τα απαραίτητα πεδία, μπορεί να αναθέσει συγκεκριμένο πρόγραμμα σε πελάτη.
Μπορούμε να δούμε στο παρακάτω στιγμιότυπο ότι στη βάση υπάρχει ο πελάτης με αριθμό τηλεφώνου 1154505527 στον οποίο όμως δεν έχει ανατεθεί κάποιο πρόγραμμα, και για αυτό το πεδίο program είναι null.



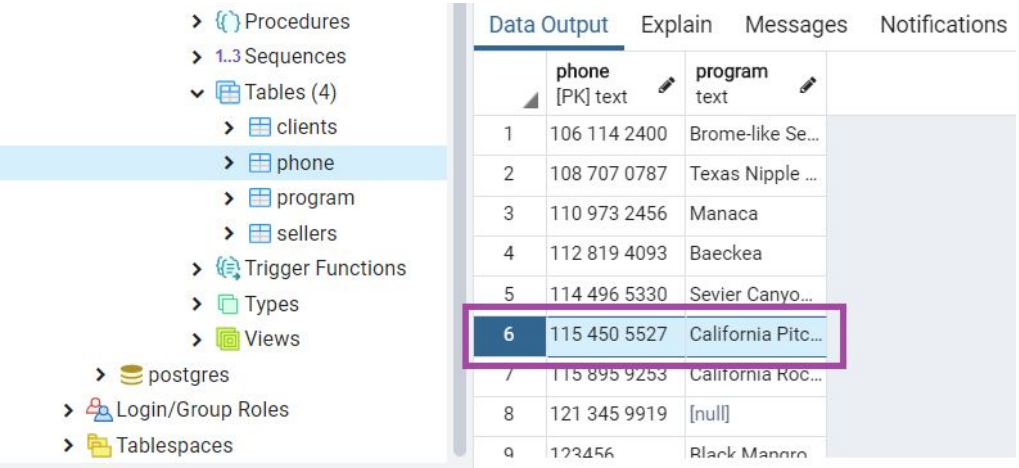
Ο χρήστης αναθέτει στο συγκεκριμένο πελάτη ένα από τα διαθέσιμα προγράμματα.



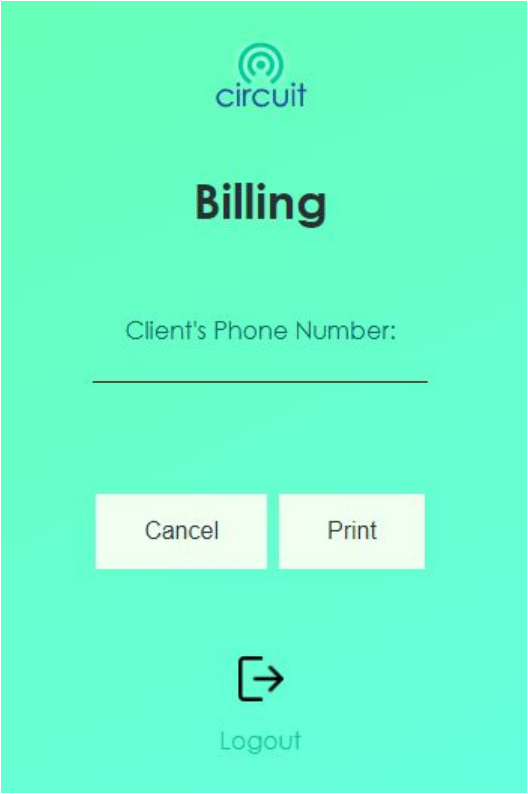
Το πρόγραμμα εμφανίζει μήνυμα επιτυχίας εφόσον η διαδικασία ολοκληρωθεί επιτυχώς και ο χρήστης αποθηκεύσει τις αλλαγές που έκανε.



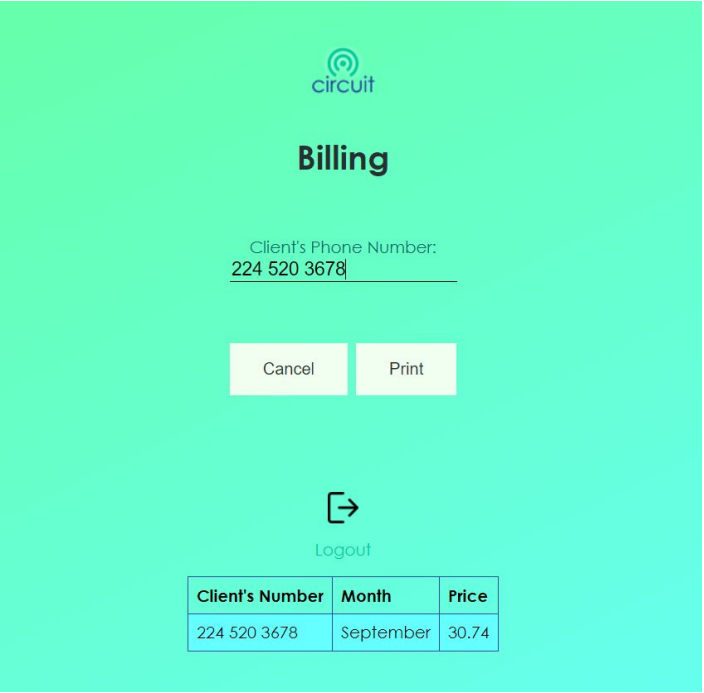
Στο παρακάτω στιγμιότυπο φαίνεται η βάση μετά την επιτυχή ανάθεση προγράμματος σε πελάτη.



Billing



Συμπληρώνοντας τον αριθμό τηλεφώνου του πελάτη ο συνδεδεμένος πωλητής μπορεί να κάνει την έκδοση του αντίστοιχου λογαριασμού.

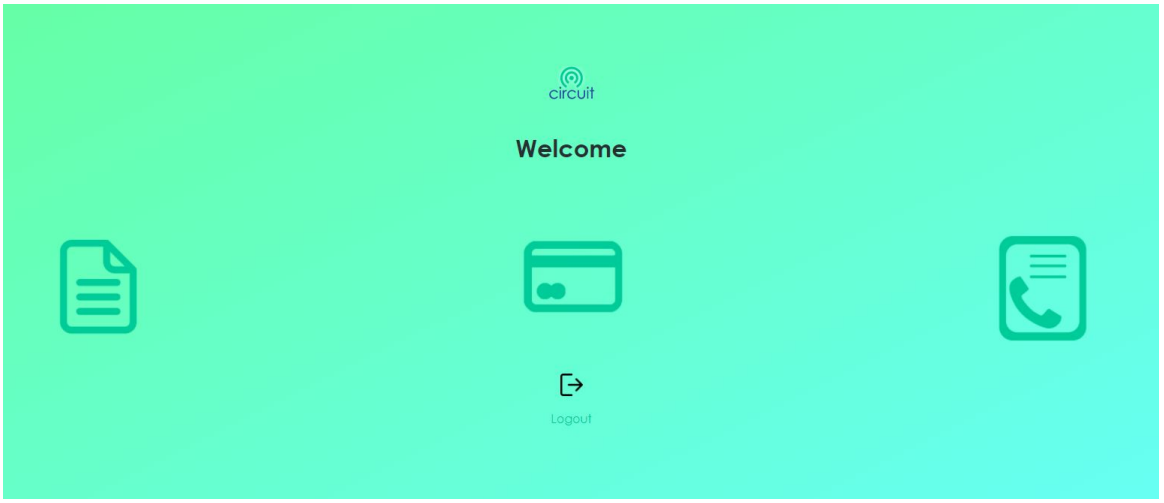


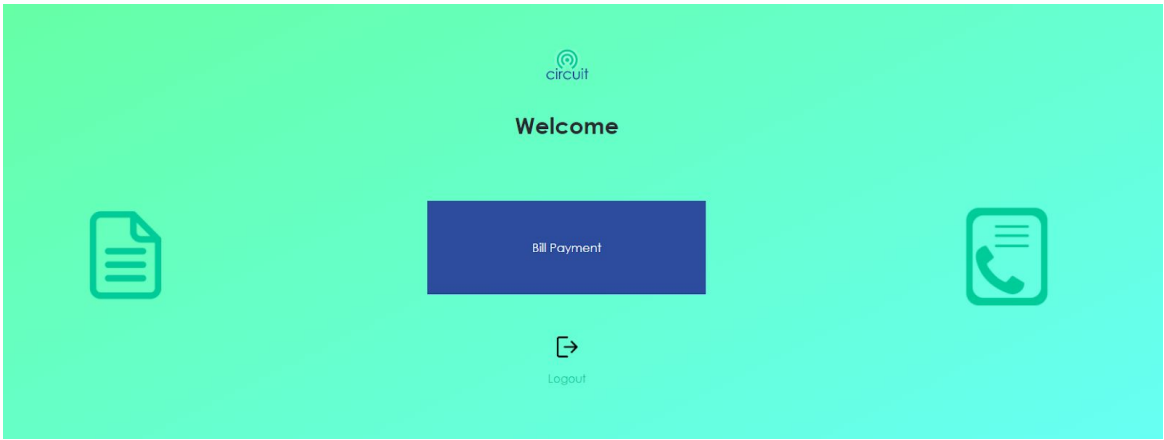
Client

Αρχική Σελίδα

Οι χρήστες που είναι καταχωρημένοι ως πελάτες έχουν τις παρακάτω δυνατότητες:

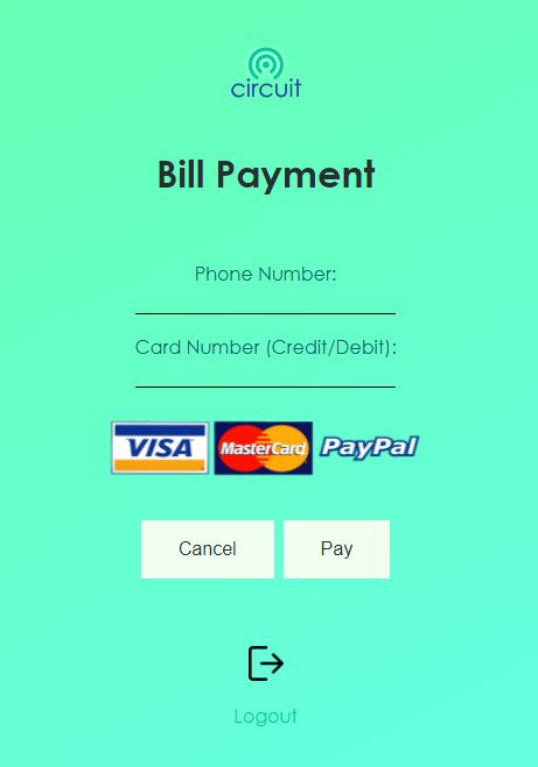
- Προβολή του τρέχοντα λογαριασμού τους (My Bill)
- Πληρωμή του τρέχοντα λογαριασμού (Bill Payment)
- Προβολή του ιστορικού κλήσεων (Call History)





Όπως φαίνεται και στην παραπάνω εικόνα ο εκάστοτε χρήστης τοποθετώντας το ποντίκι του σε κάποια από τις διαθέσιμες επιλογές (εικονίδια) μπορεί να δει μία σύντομη περιγραφή της κάθε λειτουργίας. Συγκεκριμένα, στο παραπάνω στιγμιότυπο όταν ο χρήστης τοποθετεί τον κέρσορα του στο εικονίδιο της πιστωτικής κάρτας μπορεί να δει ότι η συγκεκριμένη δυνατότητα είναι αυτή της πληρωμής του λογαριασμού του.

Bill Payment



Συμπληρώνοντας τον αριθμό του τηλεφώνου και τον αριθμό της κάρτας (χρεωστικής/ πιστωτικής) ο πελάτης μπορεί να πραγματοποιήσει την πληρωμή του τρέχοντα λογαριασμού του.

Κώδικας Προγράμματος

AdministratorDao

Μέθοδος Validate()

```
1  package dao;
2
3  import java.io.PrintWriter;
15
16 public class AdministratorDao {
17     public static boolean validate(String username, String hashed, String salt){
18
19         boolean status = false;
20         try{
21             InitialContext ctx = new InitialContext();
22             DataSource datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/postgres");
23             Connection con = datasource.getConnection();
24             PreparedStatement ps = con.prepareStatement("select * from administrators where username =? and hashed_password =? and salt =?");
25             ps.setString(1,username);
26             ps.setString(2,hashed);
27             ps.setString(3,salt);
28             ResultSet rs = ps.executeQuery();
29             status = rs.next();
30             con.close();
31         }catch(Exception ex){System.out.println(ex);}
32         return status;
33     }
```

Η μέθοδος validate() χρησιμοποιείται για την “κατηγοριοποίηση” των χρηστών, στη συγκεκριμένη περίπτωση ως Administrators. Αρχικοποιούμε μια μεταβλητή τύπου boolean με την τιμή false. (γρ. 21-23) Γίνεται η σύνδεση με τη βάση, για την οποία χρησιμοποιείται η αρχιτεκτονική 3-Tier.
Περιέχει το prepareStatement το οποίο έχει το query σε SQL και αυτό ελέγχει αν παραβιάζεται κάποιος κανόνας από τη θεωρία των βάσεων. Από τον πίνακα administrators της βάσης πηγαίνουμε στη γραμμή που έχει το username και password που έχουν συμπληρωθεί στη φόρμα και τα καταχωρούμε αντίστοιχα.Όσο δεν βρίσκεται εγγραφή που να ταιριάζει πηγαίνει στην επόμενη. Εκτελείται το query. Τέλος, επιστρέφεται το status.

Μέθοδος save() - Εισαγωγή πωλητή

```
34
35     public static int save(Seller s) {
36         int status = 0;
37         try{
38             InitialContext ctx = new InitialContext();
39             DataSource datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/postgres");
40             Connection con = datasource.getConnection();
41
42             SecureRandom random = new SecureRandom();
43             byte bytes[] = s.getPassword().getBytes();
44             random.nextBytes(bytes);
45
46             PreparedStatement ps = con.prepareStatement("insert into sellers(username,firstname,lastname, salt, hashed_password) values(?,?,?,?,?)");
47             ps.setString(1,s.getUsername());
48             ps.setString(2,s.getName());
49             ps.setString(3,s.getSurname());
50             ps.setString(4, SaltedHashed.getHashMD5(s.getPassword() ,random.toString()));
51             ps.setString(5,SaltedHashed.getHashMD5(s.getPassword()));
52             status = ps.executeUpdate();
53             con.close();
54         }catch(Exception e){System.out.println(e);}
55         return status;
56     }
57
```

Με τη μέθοδο save() γίνεται εισαγωγή νέου πωλητή. (γρ. 38-40) Γίνεται η σύνδεση με τη βάση με 3-Tier αρχιτεκτονική. Με το query που περιέχει το PreparedStatement τα στοιχεία που καταχωρεί ο χρήστης στη φόρμα εισάγονται στη βάση. (γρ. 50-51) Καλείται η μέθοδος

getHashMD5 που χρησιμοποιείται για την κρυπτογράφηση του password. Επιστρέφεται το status.

Μέθοδος save2() - Εισαγωγή νέου προγράμματος στη βάση

```
58    public static int save2(Program p) {
59        int status = 0;
60        try{
61            InitialContext ctx = new InitialContext();
62            DataSource datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/postgres");
63            Connection con = datasource.getConnection();
64
65            PreparedStatement ps = con.prepareStatement("insert into program(pname, data,sms, calls, fee) values(?,?,?,?,?)");
66            ps.setString(1,p.getProgramName());
67            ps.setInt(2,p.getData());
68            ps.setInt(3,p.getSms());
69            ps.setInt(4,p.getCalls());
70            ps.setFloat(5,p.getBillingFee());
71            status = ps.executeUpdate();
72            con.close();
73        }catch(Exception e){System.out.println(e);}
74        return status;
75    }
```

Στην μέθοδο save2() εισάγεται νέο πρόγραμμα στη βάση. Πιο συγκεκριμένα, γίνεται η σύνδεση με τη βάση με 3-Tier αρχιτεκτονική (γρ. 61-63), έπειτα εκτελείται το query το οποίο εισάγει στον πίνακα program τα στοιχεία που θα συμπληρώσει ο χρήστης στη φόρμα καλώντας τις αντίστοιχες μεθόδους. Επιστρέφει το status.

Μέθοδος save3() - Αναθεση Προγράμματος σε πελάτη

```
77    public static int save3(Program p) {
78        int status = 0;
79        try{
80            InitialContext ctx = new InitialContext();
81            DataSource datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/postgres");
82            Connection con = datasource.getConnection();
83
84            PreparedStatement ps = con.prepareStatement("update program set pname = ?, data = ?, sms = ?, calls = ?, fee = ? where pname = ?");
85            ps.setString(1,p.getProgramName());
86            ps.setInt(2,p.getData());
87            ps.setInt(3,p.getSms());
88            ps.setInt(4,p.getCalls());
89            ps.setFloat(5,p.getBillingFee());
90            ps.setString(6,p.getProgramName());
91            status = ps.executeUpdate();
92            con.close();
93        }catch(Exception e){System.out.println(e);}
94        return status;
95    }
```

Η μέθοδος save3() χρησιμοποιείται για την ανάθεση προγράμματος σε πελάτη. (γρ. 80-82) Γίνεται η σύνδεση με τη βάση με 3-Tier αρχιτεκτονική. Έπειτα εκτελείται το query το οποίο για το δοσμένο από τον χρήστη όνομα προγράμματος πηγαίνει και καταχωρεί τα πεδία που συμπληρώνει ο χρήστης στη φόρμα. Επιστρέφεται το status.

ClientDao

Μέθοδος Validate()

```
1  package dao;
2
3  import java.sql.Connection;
18
19 public class ClientDao {
20     public static boolean validate(String username, String hashed, String salt){
21
22         boolean status = false;
23         try{
24             InitialContext ctx = new InitialContext();
25             DataSource datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/postgres");
26             Connection con = datasource.getConnection();
27             PreparedStatement ps = con.prepareStatement("select * from clients where username =? and hashed_password =? and salt =?");
28             ps.setString(1,username);
29             ps.setString(2,hashed);
30             ps.setString(3,salt);
31             ResultSet rs = ps.executeQuery();
32             status = rs.next();
33             con.close();
34         }catch(Exception ex){System.out.println(ex);}
35         return status;
36     }
37
```

Η μέθοδος validate() χρησιμοποιείται για την “κατηγοριοποίηση” των χρηστών, στη συγκεκριμένη περίπτωση ως Clients. Αρχικοποιούμε μια μεταβλητή τύπου boolean με την τιμή false. (γρ. 24-26) Γίνεται η σύνδεση με τη βάση, για την οποία χρησιμοποιείται η αρχιτεκτονική 3-Tier.
Περιέχει το prepareStatement με το query το οποίο από τον πίνακα administrators της βάσης πηγαίνει στη γραμμή που έχει το username και password που έχουν συμπληρωθεί στη φόρμα και τα καταχωρεί αντίστοιχα. Όσο δεν βρίσκεται εγγραφή που να ταιριάζει πηγαίνει στην επόμενη. Εκτελείται το query. Τέλος, επιστρέφεται το status.

Μέθοδος showCallHistory()

```
    public static List<Call> showCallHistory(String username){

        List<Call> list = new ArrayList<Call>();
        String phone = null;
        try{
            InitialContext ctx = new InitialContext();
            DataSource datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/postgres");
            Connection con = datasource.getConnection();

            PreparedStatement ps2 = con.prepareStatement("select phone from clients where username = ?");
            ps2.setString(1, username);
            ResultSet rs = ps2.executeQuery();
            while(rs.next()){
                phone = rs.getString(1);
            }

            PreparedStatement ps = con.prepareStatement("select * from call where phone = ?");
            ps.setString(1, phone);
            ResultSet rs2 = ps.executeQuery();
            while(rs2.next()){
                Call c = new Call();
                c.setReceiver(rs.getString(1));
                c.setCallDate(rs.getString(2));
                c.setCallTime(rs.getString(3));
                c.setCallDuration(rs.getString(4));
                c.setCallType(rs.getString(5));
                list.add(c);
            }
            con.close();
        }catch(Exception e){System.out.println(e);}
        return list;
```

Η μέθοδος showCallHistory() χρησιμοποιείται για την προβολή του ιστορικού κλήσεων από τους πελάτες. (γρ. 40) Δημιουργείται μία λίστα με τις κλήσεις. (γρ. 41) Αρχικοποιείται μία string μεταβλητή phone ως null. (γρ. 43-45) Γίνεται η σύνδεση με τη βάση.
(γρ. 47-51)Έπειτα με βάση το username που έχει καταχωρήσει ο χρήστης κατά τη διαδικασία Login δημιουργείται το query στο preparedStatement το οποίο επιλέγει το τηλέφωνο από τον πίνακα clients της εγγραφής που έχει το username που καταχωρήθηκε κατά την είσοδο χρήστη. Εκτελείται το query. Όσο δεν βρίσκει την εγγραφή που αναζητά πηγαίνει στην επόμενη. Όταν βρεθεί η εγγραφή τότε αποθηκεύεται στην μεταβλητή phone.
(γρ. 54-68) Στο preparedStatement υπάρχει το query το οποίο κάνει μία επιλογή από τον πίνακα call με βάση το τηλέφωνο. (γρ. 58) Δημιουργείται ένα αντικείμενο c τύπου Call και έπειτα καταχωρούνται σε αυτό οι τιμές των αντίστοιχων χαρακτηριστικών. Τέλος, επιστρέφεται η λίστα.

ProgramDao

Μέθοδος save()

```
17    public static int save(Client c){
18
19        int status = 0;
20        try{
21            InitialContext ctx = new InitialContext();
22            DataSource datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/postgres");
23            Connection con = datasource.getConnection();
24
25            PreparedStatement ps = con.prepareStatement("insert into phone(phone, program) values (?, ?)");
26            ps.setString(1, c.getPhone());
27            ps.setString(2, null);
28            status = ps.executeUpdate();
29
30            SecureRandom random = new SecureRandom();
31            byte bytes[] = c.getPassword().getBytes();
32            random.nextBytes(bytes);
33
34            PreparedStatement ps2 = con.prepareStatement("insert into clients(username,firstname,lastname,afm, salt, hashed_password, phone) values(?,?,?,?,?,?,?)");
35            ps2.setString(1,c.getUsername());
36            ps2.setString(2,c.getName());
37            ps2.setString(3,c.getSurname());
38            ps2.setString(4, c.getAfm());
39            ps2.setString(5, SaltedHashed.getHashMD5(c.getPassword() ,random.toString()));
40            ps2.setString(6,SaltedHashed.getHashMD5(c.getPassword()));
41            ps2.setString(7, c.getPhone());
42            status = ps2.executeUpdate();
43            con.close();
44        }catch(Exception e){System.out.println(e);}
45        return status;
46    }
```

Η μέθοδος save() χρησιμοποιείται για την καταχώρηση νέου πελάτη.
(γρ. 21-23) Γίνεται η σύνδεση με τη βάση.
(γρ. 25-28) Στο preparedStatement βρίσκεται το query το οποίο εισάγει στον πίνακα phone τις τιμές που θα βάλει ο χρήστης στη φόρμα για τα αντίστοιχα πεδία του τηλεφώνου και του προγράμματος. Εκτελείται το query.
(γρ. 34-45) Στο preparedStatement βρίσκεται το query το οποίο εισάγει στον πίνακα clients τις τιμές που συμπληρώνει ο χρήστης στη φόρμα για τα αντίστοιχα πεδία: username, firstname, lastname, afm, salt, hashed_password, phone. Εκτελείται το query και επιστρέφεται το status.

Μέθοδος getAllRecords() - Εμφανίζει όλα τα προγράμματα

```
48    public static List<Program> getAllRecords(){
49
50        List<Program> list = new ArrayList<Program>();
51        try{
52            InitialContext ctx = new InitialContext();
53            DataSource datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/postgres");
54            Connection con = datasource.getConnection();
55            PreparedStatement ps = con.prepareStatement("select * from program");
56            ResultSet rs = ps.executeQuery();
57            while(rs.next()){
58                Program p = new Program();
59                p.setProgramName(rs.getString(1));
60                p.setData(rs.getInt(2));
61                p.setSms(rs.getInt(3));
62                p.setCalls(rs.getInt(4));
63                p.setBillingFee(rs.getFloat(5));
64                list.add(p);
65            }
66            con.close();
67        }catch(Exception e){System.out.println(e);}
68        return list;
69    }
70
```

Η μέθοδος getAllRecords() χρησιμοποιείται για την εμφάνιση όλων των προγραμμάτων.
(γρ. 50) Δημιουργείται μία λίστα.
(γρ. 53-55) Γίνεται η σύνδεση με τη βάση.
(γρ.55-64) Στο preparedStatement βρίσκεται το query το οποίο επιλέγει όλες τις εγγραφές από τον πίνακα program.
(γρ. 57-64) Μέσα σε μία δομή επανάληψης while, όσο βρίσκονται εγγραφές στον πίνακα, δημιουργείται αντικείμενο τύπου Program και καταχωρούνται οι τιμές στα αντίστοιχα χαρακτηριστικά του. Το αντικείμενο προστίθεται στη λίστα.
(γρ. 68) Επιστρέφει τη λίστα.

Μέθοδος AssignProgramToClient()

```
public static int assignProgramToClient(String clientPhone, String clientProgram) {
    int status = 0;
    try{
        InitialContext ctx = new InitialContext();
        DataSource datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/postgres");
        Connection con = datasource.getConnection();

        PreparedStatement ps = con.prepareStatement("update phone set phone=?, program=? where phone=?");
        ps.setString(1,clientPhone);
        ps.setString(2,clientProgram);
        ps.setString(3,clientPhone);
        status = ps.executeUpdate();

        con.close();

    }catch(Exception e){System.out.println(e);}
    return status;
}
```

Δίνεται η τιμή 0 (μηδέν) στην ακέραια μεταβλητή status. Γίνεται η σύνδεση με τη βάση. Στο PreparedStatement υπάρχει το query το οποίο στον πίνακα phone ψάχνει την εγγραφή στην οποίο το phone είναι αυτό που έχει εισάγει ο χρήστης στη φόρμα και αφότου βρεθεί η εγγραφή καταχωρείται το νέο πρόγραμμα που θα εισάγει ο χρήστης στη φόρμα στο αντίστοιχο πεδίο. Επιστρέφεται το status.

SellerDao

Μέθοδος Validate()

```
1  package dao;
2
3  import java.security.SecureRandom;
20
21 public class SellerDao {
22     public static boolean validate(String username, String hashed, String salt){
23
24         boolean status = false;
25         try{
26             InitialContext ctx = new InitialContext();
27             DataSource datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/postgres");
28             Connection con = datasource.getConnection();
29             PreparedStatement ps = con.prepareStatement("select * from sellers where username =? and hashed_password =? and salt =?");
30             ps.setString(1,username);
31             ps.setString(2,hashed);
32             ps.setString(3,salt);
33             ResultSet rs = ps.executeQuery();
34             status = rs.next();
35             con.close();
36         }catch(Exception ex){System.out.println(ex);}
37         return status;
38     }
39
```

Η μέθοδος validate() χρησιμοποιείται για την “κατηγοριοποίηση” των χρηστών, στη συγκεκριμένη περίπτωση ως Sellers. Αρχικοποιούμε μια μεταβλητή τύπου boolean με την τιμή false. (γρ. 24-26) Γίνεται η σύνδεση με τη βάση, για την οποία χρησιμοποιείται η αρχιτεκτονική 3-Tier.

Περιέχει το prepareStatement με το query το οποίο από τον πίνακα sellers της βάσης πηγαίνει στη γραμμή που έχει το username και password που έχουν συμπληρωθεί στη φόρμα και τα καταχωρεί αντίστοιχα. Όσο δεν βρίσκεται εγγραφή που να ταιριάζει πηγαίνει στην επόμενη. Εκτελείται το query. Τέλος, επιστρέφεται το status.

Μέθοδος ShowClientBill

```
40     public static List<Bill> showClientBill(String clientPhone) {
41         List<Bill> list = new ArrayList<Bill>();
42         try{
43             InitialContext ctx = new InitialContext();
44             DataSource datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/postgres");
45             Connection con = datasource.getConnection();
46
47             YearMonth thisMonth    = YearMonth.now();
48             DateTimeFormatter monthYearFormatter = DateTimeFormatter.ofPattern("MMMM");
49             //System.out.printf("Today: %s\n", thisMonth.format(monthYearFormatter));
50
51             //String month = thisMonth.format(monthYearFormatter);
52             String month = "September";
53             PreparedStatement ps = con.prepareStatement("select * from bill where month = ?");
54             ps.setString(1, month);
55             ResultSet rs = ps.executeQuery();
56             while(rs.next()){
57                 Bill b = new Bill();
58                 b.setPhoneNumber(rs.getString(1));
59                 b.setBillingMonth(rs.getString(2));
60                 b.setPrice(rs.getString(3));
61                 list.add(b);
62             }
63             con.close();
64         }catch(Exception e){System.out.println(e);}
65         return list;
66
67     }
```

Η μέθοδος ShowClientsBill χρησιμοποιείται για την έκδοση λογαριασμού του πελάτη.
(γρ. 41) Δημιουργείται μία λίστα Bill.
(γρ. 43-45) Γίνεται η σύνδεση με τη βάση.

(γρ. 47-49) Εύρεση του τρέχοντα μήνα.
(γρ. 53-65) Στο PreparedStatement υπάρχει το query το οποίο επιλέγει από τον πίνακα bill όλες τις εγγραφές όπου ο μήνας είναι ο τρέχοντας. Με μία δομή επανάληψης while, όσο υπάρχουν επόμενες εγγραφές, δημιουργείται αντικείμενο b τύπου Bill το οποίο παίρνει τιμές στα αντίστοιχα πεδία του και προστίθεται στη λίστα.
(γρ. 65) Επιστρέφεται η λίστα.

Κλάση ViewProgramName

```
3  import askhsh1.Program;
13
14 @WebServlet("/ViewProgramName")
15 public class ViewProgramName extends HttpServlet {
16
17     private static final long serialVersionUID = 1L;
18
19     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
20         response.setContentType("text/html");
21         PrintWriter out = response.getWriter();
22
23         out.println("<title>View Programs</title>");
24
25         request.getRequestDispatcher("viewPrograms.jsp").include(request, response);
26
27         //out.println("<h1 style=\"font-family:century gothic; color:#292F33;\">View Programs</h1>");
28         List<Program> list = ProgramDao.getAllRecords();
29         out.print("<table>");
30         out.print("<tr><th>"+"Program Name"+"</th><th>"+"MB"+"</th><th>"+"SMS"+"</th><th>"+"Talk Time"+"</th><th>"+"Price"+"</th></tr>");
31         for(Program p:list){
32             out.println("<tr><td>"+p.getProgramName()+"</td><td>"+p.getData()+"</td><td>"+p.getSms()+"</td><td>"+p.getCalls()+"</td><td>"+p.getBillingFee()+"</td>");
33
34             out.println("</tr>");
35         }
36         out.println("</table>");
37         out.println("<center>");
38         out.println("<a href=\"index.jsp\">");
39         out.println("</br></br></br></br>");
40         out.println("<img src=\"logout.png\" alt=\"Logout\" width=\"30\"; height=\"30\";>");
41         out.println("</a>");
42         out.println("<p>Logout</p>");
43         out.close();
44     }
45
46 }
```

Η ViewProgramName είναι servlet. Η μέθοδος doGet χρησιμοποιείται για την προβολή των προγραμμάτων.
(γρ. 20) Ο τύπος που θα εμφανίσει είναι html.
(γρ. 25) Γίνεται η σύνδεση με τη jsp σελίδα viewPrograms.
(γρ. 28) Δημιουργείται μία λίστα με τα προγράμματα και καλείται από την κλάση ProgramDao η μέθοδος getAllRecords.
(γρ. 29-42) Γίνεται η εμφάνιση των επιθυμητών στοιχείων για κάθε αντικείμενο p τύπου Program που υπάρχει στη λίστα.

Κλάση SaltedHashed

```java
package mainpackage;

import java.math.BigInteger;

public class SaltedHashed {

    public static String getHashMD5(String unhashed) {
            return getHashMD5(unhashed, "");
        }

        /**
         * Computes the hash of the given string salted. Return the hash uppercase.
         * @param unhashed the string to hash
         * @param salt the salt to use
         * @return the hash of the given string salted and uppercase.
         * @throws NoSuchAlgorithmException
         */
        public static String getHashMD5(String unhashed, String salt) {
            // Hash the password.
            final String toHash = salt + unhashed + salt;
            MessageDigest messageDigest = null;
            try {
                messageDigest = MessageDigest.getInstance("MD5");
            } catch (NoSuchAlgorithmException ex) {
                return "00000000000000000000000000000000";
            }
            messageDigest.update(toHash.getBytes(), 0, toHash.length());
            String hashed = new BigInteger(1, messageDigest.digest()).toString(16);
            if (hashed.length() < 32) {
                hashed = "0" + hashed;
            }
            return hashed.toUpperCase();
        }

}
```

Η κλάση SaltedHashed βασίζεται στον αλγόριθμο MD5 για την κρυπτογράφηση του κωδικού πρόσβασης των χρηστών.

AdminServlet

```java
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        //response.getWriter().append("Served at: ").append(request.getContextPath());
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        request.getRequestDispatcher("registerSeller.jsp").include(request, response);


        String sellerFirstName = request.getParameter("name");
        String sellerLastName = request.getParameter("surname");
        String sellerUsername = request.getParameter("username");
        String sellerPassword = request.getParameter("password");
        Seller s = new Seller(sellerUsername, sellerFirstName, sellerLastName, sellerPassword);

        if(sellerUsername == "" || sellerFirstName == "" || sellerLastName == "" || sellerPassword == "")
            out.print("<h3 style= \"font-family:century gothic; color:#292F33;\">Unable to add client!</h3>");
        else {
            int status = AdministratorDao.save(s);
            if(status>0)
                out.print("<h3 style= \"font-family:century gothic; color:#292F33;\">Seller added successfully</h3>");
            else
                out.print("<h3 style= \"font-family:century gothic; color:#292F33;\">Unable to add seller!</h3>");
        }
        out.close();
    }
```

Η AdminServlet είναι servlet.

Η μέθοδος doGet χρησιμοποιείται για την εγγραφή νέου πωλητή.
(γρ. 20) Ο τύπος που θα εμφανίσει είναι html.
(γρ. 40) Γίνεται η σύνδεση με την jsp σελίδα registerSeller για την εγγραφή νέου πωλητή.
(γρ. 43-46) Παίρνει τα στοιχεία που δίνει ο χρήστης στη φόρμα.
(γρ. 47) Δημιουργείται ένα αντικείμενο s τύπου Seller.
(γρ. 49-56) Χρησιμοποιώντας δομή επιλογής, σε περίπτωση που κάποιο από τα πεδία της φόρμας δεν συμπληρώθηκε από τον χρήστη, εμφανίζεται μήνυμα αποτυχίας εγγραφής πωλητή. Αλλιώς, καλείται η save() από την AdministratorDao η οποία επιστρέφει την τιμή του status. Αν η τιμή της μεταβλητής status είναι μεγαλύτερη του μηδενός τότε γίνεται επιτυχής καταχώρηση του νέου πωλητή και εμφανίζεται μήνυμα επιτυχίας στον χρήστη, αλλιώς εμφανίζεται μήνυμα αποτυχίας και δεν γίνεται η εγγραφή του πωλητή.

```
64    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
65        // TODO Auto-generated method stub
66        //doGet(request, response);
67
68        response.setContentType("text/html");
69        PrintWriter out = response.getWriter();
70
71        request.getRequestDispatcher("newProgram.jsp").include(request, response);
72
73        String pname = request.getParameter("pname");
74        int calls = Integer.parseInt(request.getParameter("calls"));
75        int sms = Integer.parseInt(request.getParameter("sms"));
76        int data = Integer.parseInt(request.getParameter("data"));
77        float price = Float.parseFloat(request.getParameter("price"));
78        Program p = new Program(pname, data, sms, calls, price);
79
80        if(pname == "" )
81            out.print("<h3 style= \"font-family:century gothic; color:#292F33;\">Unable to add program!</h3>");
82        else {
83            int status = AdministratorDao.save2(p);
84            if(status>0)
85                out.print("<h3 style= \"font-family:century gothic; color:#292F33;\">Program added successfully</h3>");
86            else
87                out.print("<h3 style= \"font-family:century gothic; color:#292F33;\">Unable to add program!</h3>");
88        }
89        out.close();
90
91
92    }
```

Η μέθοδος doPost χρησιμοποιείται για την καταχώρηση νέου προγράμματος.
(γρ. 71) Γίνεται η σύνδεση με την jsp σελίδα newProgram.
(γρ. 73-77) Η request.getParameter παίρνει από τη φόρμα τα στοιχεία που εισάγει ο χρήστης.
(γρ. 78) Δημιουργείται το αντικείμενο p τύπου Program.
(γρ. 80-88) Χρησιμοποιώντας δομή επιλογής, σε περίπτωση που κάποιο από τα πεδία της φόρμας δεν συμπληρώθηκε από τον χρήστη, εμφανίζεται μήνυμα αποτυχίας προσθήκης νέου προγράμματος. Αλλιώς, καλείται η save2() από την AdministratorDao η οποία επιστρέφει την τιμή του status. Αν η τιμή της μεταβλητής status είναι μεγαλύτερη του μηδενός τότε γίνεται επιτυχής εισαγωγή νέου προγράμματος και εμφανίζεται μήνυμα επιτυχίας στον χρήστη, αλλιώς εμφανίζεται μήνυμα αποτυχίας και δεν γίνεται η εγγραφή του νέου προγράμματος.

AdminServlet2

```
35    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
36        // TODO Auto-generated method stub
37        //response.getWriter().append("Served at: ").append(request.getContextPath());
38
39        //doGet(request, response);
40
41        response.setContentType("text/html");
42        PrintWriter out = response.getWriter();
43
44        request.getRequestDispatcher("programModify.jsp").include(request, response);
45
46        List<Program> list = ProgramDao.getAllRecords();
47        out.println("<head>\r\n" +
48                "<center><img src=\"logo3.png\" alt=\"logo\">\r\n" +
49                "   </br></br>\r\n" +
50                "   <center><h1>Modify an existing Program</h1>\r\n" +
51                "</head>");
52        out.println("<body>\r\n" +
53                "<form method=\"get\" action=\"AdminServlet2\">");
54        out.println("<label for=\"program\">Select the Program you want to modify:</label><br>");
55        out.println("<select id=\"program\" name=\"pname\"></br></br>");
56        for(Program p:list){
57
58            out.println("<option value=\"pname\" name=\"pname\">" +p.getProgramName() + "</option>");
59        }
60        out.println("</select><br/><br/>");
61        out.println("<label for=\"program\">Modify the wanted fields:</label><br>\r\n" +
62                "<br/><br/>\r\n" +
63                "<label for=\"calls\">Calls:</label>\r\n" +
64                "   </br>\r\n" +
65                "   <input type=\"text\" id=\"calls\" name=\"calls\">\r\n" +
66                "   </br></br>\r\n" +
67                "   <label for=\"sms\">SMS:</label>\r\n" +
68                "   </br>\r\n" +
69                "   <input type=\"text\" id=\"sms\" name=\"sms\">\r\n" +
70                "   </br></br>\r\n" +
```

```
71                "   <label for=\"data\">Data:</label>\r\n" +
72                "   </br>\r\n" +
73                "   <input type=\"text\" id=\"data\" name=\"data\">\r\n" +
74                "   </br></br>\r\n" +
75                "   <label for=\"price\">Price:</label>\r\n" +
76                "   </br>\r\n" +
77                "   <input type=\"text\" id=\"price\" name=\"price\">\r\n" +
78                "   </br></br></br></br>");
79        out.println("<a href=\"index.jsp\"><button  class=\"button button2\">Cancel</button></a>\r\n" +
80                "   <button class=\"button button1\">Save</button>\r\n" +
81                "   </br></br>\r\n");
82        out.println("</form>\r\n" +
83                "</br></br></br>\r\n" +
84                "<center><a href=\"index.jsp\">\r\n" +
85                "  <img src=\"logout.png\" alt=\"Logout\" width=\"30\"; height=\"30\";>\r\n" +
86                "  </a>\r\n" +
87                "  <p>Logout</p>\r\n" +
88                "</body>");
89        out.close();
90
91        String pname = request.getParameter("pname");
92        int calls = Integer.parseInt(request.getParameter("calls"));
93        int sms = Integer.parseInt(request.getParameter("sms"));
94        int data = Integer.parseInt(request.getParameter("data"));
95        float price = Float.parseFloat(request.getParameter("price"));
96        Program p = new Program(pname, data, sms, calls, price);
97
98        if(pname == "" )
99            out.print("<h3 style= \"font-family:century gothic; color:#292F33;\">Unable to add program!</h3>");
100       else {
101           int status = AdministratorDao.save3(p);
102           if(status>0)
103               out.print("<h3 style= \"font-family:century gothic; color:#292F33;\">Program added successfully</h3>");
104           else
105               out.print("<h3 style= \"font-family:century gothic; color:#292F33;\">Unable to add program!</h3>");
106       }
107       out.close();
```

Η AdminServlet2 είναι servlet.

Η μέθοδος doGet χρησιμοποιείται για την τροποποίηση ήδη υπάρχοντος προγράμματος. (γρ. 44) Γίνεται η σύνδεση με την jsp σελίδα programModify.

(γρ. 46) Δημιουργείται μία λίστα και καλείται η getAllRecords από την ProgramDao.
(γρ. 47-89) Εμφανίζονται τα διαθέσιμα προς τροποποίηση προγράμματα.
(γρ. 91-95) Παίρνει τα στοιχεία που καταχωρεί ο χρήστης στη φόρμα.
(γρ. 96) Δημιουργείται ένα αντικείμενο p τύπου Program.
(γρ. 98-105) Χρησιμοποιώντας δομή επιλογής, σε περίπτωση που κάποιο από τα πεδία της φόρμας δεν συμπληρώθηκε από τον χρήστη, εμφανίζεται μήνυμα αποτυχίας προσθήκης νέου προγράμματος. Αλλιώς, καλείται η save3() από την AdministratorDao η οποία επιστρέφει την τιμή του status. Αν η τιμή της μεταβλητής status είναι μεγαλύτερη του μηδενός τότε η διαδικασία ολοκληρώνεται επιτυχώς και εμφανίζεται μήνυμα επιτυχίας στον χρήστη, αλλιώς δεν ολοκληρώνεται η διαδικασία και εμφανίζεται στον χρήστη αντίστοιχο μήνυμα αποτυχίας.

ClientHistoryServlet

```
38    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
39        // TODO Auto-generated method stub
40        //response.getWriter().append("Served at: ").append(request.getContextPath());
41        //doGet(request, response);
42        response.setContentType("text/html");
43        PrintWriter out = response.getWriter();
44
45        request.getRequestDispatcher("historyCall.jsp").include(request, response);
46
47        HttpSession session = request.getSession();
48        String username = (String) session.getAttribute("username");
49
50        List<Call> list = ClientDao.showCallHistory(username);
51        out.print("<table>");
52        out.print("<tr><th>"+"Call with"+"</th><th>"+"Date"+"</th><th>"+"Time"+"</th><th>"+"Duration"+"</th><th>"+"Type"+"</th></tr>");
53        for(Call c:list){
54            out.println("<tr><td>"+c.getReceiver()+"</td><td>"+c.getCallDate()+"</td><td>"+c.getCallTime()+"</td><td>"+c.getCallDuration()+"</td><td>"+c.getCallType()+"
55
56            out.println("</tr>");
57        }
58        out.println("</table>");
59        out.println("<center>");
60        out.println("<a href=\"index.jsp\">");
61        out.println("</br></br></br></br>");
62        out.println("<img src=\"logout.png\" alt=\"Logout\" width=\"30\"; height=\"30\";>");
63        out.println("</a>");
64        out.println("<p>Logout</p>");
65        out.close();
66    }
```

Η ClientHistoryServlet είναι servlet.
Η μέθοδος doGet χρησιμοποιείται για την προβολή του ιστορικού του πελάτη.
(γρ. 45) Γίνεται η σύνδεση με την jsp σελίδα historyCall.
(γρ. 50) Δημιουργείται μία λίστα Call και καλείται η showCallHistory από την ClientDao.
(γρ. 51-64) Γίνεται εμφάνιση των επιθυμητών στοιχείων.

SellerLogin

```
36    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
37
38        response.setContentType("text/html");
39        PrintWriter out = response.getWriter();
40        response.setContentType("text/html");
41
42        String username = request.getParameter("username");
43        String password = request.getParameter("password");
44
45        try {
46            SecureRandom random = new SecureRandom();
47            byte bytes[] = password.getBytes();
48            random.nextBytes(bytes);
49            String hashed = SaltedHashed.getHashMD5(password);
50            String salt = SaltedHashed.getHashMD5(password ,random.toString());
51
52            boolean status = SellerDao.validate(username, hashed, salt);
53            boolean status1 = AdministratorDao.validate(username, hashed, salt);
54            boolean status2 = ClientDao.validate(username, hashed, salt);
55
56            if(status){
57                HttpSession session = request.getSession();
58                session.setAttribute("username",username);
59                response.sendRedirect("startPage.jsp");
60            }
61            else if(status1) {
62                HttpSession session = request.getSession();
63                session.setAttribute("username",username);
64                response.sendRedirect("startPageAdmin.jsp");
65
66            }
67            else if(status2) {
68                HttpSession session = request.getSession();
69                session.setAttribute("username",username);
70                response.sendRedirect("startPageClients.jsp");
71            }
```

```
72            else{
73                out.print("<h3 style= \"font-family:century gothic; color:#292F33;\">Incorrect username or password. Please try again</h3>");
74            }
75
76            }catch (ProviderException e) {
77
78                System.out.println("Exception thrown : " + e);
79            }
80    }
```

Η SellerLogin είναι servlet και παρά το όνομά της, αποτελεί την αρχική LoginPage για όλους τους χρήστες.

(γρ. 42-43) Παίρνει τα στοιχεία που εισάγει ο χρήστης στη φόρμα (username, password).

(γρ. 46-50) Γίνεται η διαδικασία αντιστοίχισης της κρυπτογράφησης του password. Πιο συγκεκριμένα, καλείται η getHashMD5 από την SaltedHashed.

(γρ. 52-54) Έχουμε τις τρεις μεταβλητές:

- status : καλεί την validate από την SellerDao
- status1: καλεί την validate από την AdministratorDao
- status2: καλεί την validate από την ClientDao

(γρ. 45-78) Ανάλογα με το status που επιστρέφεται, με μία δομή επιλογής, γίνεται ο διαχωρισμός της κατηγορίας στην οποία ανήκει ο χρήστης. Στις περιπτώσεις επιτυχούς ταυτοποίησης εμφανίζεται στον χρήστη το αντίστοιχο μενού επιλογής αναλόγως με το αν είναι admin, client ή seller. Αν αποτύχει η ταυτοποίηση τότε εμφανίζεται αντίστοιχο μήνυμα στον χρήστη.

SellerLogout

```java
package mainpackage;

import java.io.IOException;

/**
 * Servlet implementation class ContentAdminLogout
 */
@WebServlet("/SellerLogout")
public class SellerLogout extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Logout Admin</title>");
        out.println("<link rel='stylesheet' href='resources/bootstrap.min.css'/>");
        out.println("</head>");
        out.println("<body>");
        request.getRequestDispatcher("index.jsp").include(request, response);

        out.println("<h1>You are successfully logged out</h1>");

        HttpSession session = request.getSession();
        session.removeAttribute("username");
        session.invalidate();
        response.sendRedirect("index.jsp");

        out.close();
    }
}
```

Η SellerLogout είναι η σελίδα αποσύνδεσης όλων των χρηστών και είναι servlet.
Όταν ο χρήστης κάνει αποσύνδεση το session τερματίζει (γρ. 6-9), τότε ο χρήστης γίνεται redirect στην αρχική σελίδα για να κάνει εκ νεου login εφόσον το επιθυμεί.

SellerServlet

```java
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        request.getRequestDispatcher("addClient.jsp").include(request, response);


        String clientUsername = request.getParameter("username");
        String clientFirstName = request.getParameter("name");
        String clientLastName = request.getParameter("surname");
        String clientPassword = request.getParameter("password");
        String clientAfm = request.getParameter("afm");
        String clientPhone = request.getParameter("phone");

        Client c = new Client(clientUsername, clientFirstName, clientLastName, clientPassword, clientAfm, clientPhone);

        if(clientUsername == "" || clientFirstName == "" || clientLastName == "" || clientPassword == "")
            out.print("<h3 style= \"font-family:century gothic; color:#292F33;\">Unable to add client!</h3>");
        else {
            int status = ProgramDao.save(c);
            if(status>0)
                out.print("<h3 style= \"font-family:century gothic; color:#292F33;\">Client added successfully</h3>");
            else
                out.print("<h3 style= \"font-family:century gothic; color:#292F33;\">Unable to add client!</h3>");
        }
        out.close();

    }
```

Η SellerServlet είναι servlet.
Η μέθοδος doPost χρησιμοποιείται για την προσθήκη νέου πελάτη από κάποιον πωλητή.

(γρ. 25) Γίνεται η σύνδεση με την jsp σελίδα addClient.
(γρ. 28-33) Παίρνει τα απαραίτητα στοιχεία από τη φόρμα που συμπληρώνει ο χρήστης.
(γρ. 35) Δημιουργείται αντικείμενο c τύπου Client.
(γρ.37-38) Δομή επιλογής, αν κάποιο από τα πεδία που συμπλήρωσε ο χρήστης είναι κενό τότε εμφανίζεται αντίστοιχο μήνυμα αποτυχίας. Αλλιώς, (γρ. 39-44) καλείται η save από την ProgramDao η οποία επιστρέφει το status. Αν το status είναι θετικό, γίνεται επιτυχώς η εγγραφή του νέου πελάτη και εμφανίζεται μήνυμα επιτυχίας στον χρήστη. Διαφορετικά, η διαδικασία αποτυγχάνει και εμφανίζεται μήνυμα αποτυχίας στον χρήστη.

```
50    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
51
52        response.setContentType("text/html");
53        PrintWriter out = response.getWriter();
54
55        request.getRequestDispatcher("assignProgramToClient.jsp").include(request, response);
56
57        String clientPhone = request.getParameter("phone");
58        String clientProgram = request.getParameter("program");
59
60        if(clientPhone == "" || clientProgram== "") {
61
62            out.print("<h3 style= \"color:white;\">Unable to assign program!</h3>");
63
64        }else {
65
66            int status = ProgramDao.assignProgramToClient(clientPhone, clientProgram);
67
68            if(status>0){
69
70                out.print("<h3 style= \"color:white;\">Program assigned succesfully!</h3>");
71
72            }else{
73                out.print("<h3 style= \"color:white;\">Unable to assign program!</h3>");
74            }
75
76        }
77
78
79        out.close();
80    }
```

Η μέθοδος doGet χρησιμοποιείται για την ανάθεση προγράμματος σε πελάτη.
(γρ. 55) Γίνεται η σύνδεση με την jsp σελίδα assignProgramToClient.
(γρ. 57-58) Παίρνει τα στοιχεία που συμπληρώνει ο χρήστης στη φόρμα.
(γρ. 60) Αν τα πεδία που έχει συμπληρώσει ο χρήστης είναι κενά τότε εμφανίζεται μήνυμα αποτυχίας. Αλλιώς, (γρ.64-66) καλείται η assignProgramToClient από την ProgramDao η οποία επιστρέφει το status. (γρ. 68-76) Αν το status είναι θετικό η ανάθεση προγράμματος ολοκληρώνεται επιτυχώς και εμφανίζεται αντίστοιχο μήνυμα επιτυχίας στον χρήστη. Αλλιώς, η διαδικασία αποτυγχάνει και εμφανίζεται μήνυμα αποτυχίας στον χρήστη.

SellerShowBill

```
38●     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
39          // TODO Auto-generated method stub
40          //response.getWriter().append("Served at: ").append(request.getContextPath());
41          response.setContentType("text/html");
42          PrintWriter out = response.getWriter();
43
44          out.println("<title>View Film</title>");
45
46          request.getRequestDispatcher("viewPrograms.jsp").include(request, response);
47
48          //out.println("<h1 style=\"color:orange;\">View All Programs</h1>");
49          List<Program> list = ProgramDao.getAllRecords();
50          out.print("<center><img src=\"logo3.png\" alt=\"logo\"> <br/><br/> <center> <h1>View all Programs</h1>");
51          out.print("<table>");
52          for(Program p:list){
53              out.println("<tr><td>"+p.getProgramName()+"</td><td>"+p.getCalls()+"</td><td>"+p.getSms()+"</td><td>"+p.getData() + "</td><td>"+p.getBillingFee()+"</td>");
54
55              out.println("</tr>");
56          }
57          out.println("</table>");
58          out.close();
59      }
60
```

Η SellerShowBill είναι servlet.
Η μέθοδος doGet χρησιμοποιείται για την προβολή των προγραμμάτων.
(γρ. 49) Δημιουργείται μία λίστα με τα προγράμματα αφού κληθεί η getAllRecords από την ProgramDao.
Έπειτα εμφανίζονται τα αποτελέσματα.

```
64●     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
65          // TODO Auto-generated method stub
66          //doGet(request, response);
67          response.setContentType("text/html");
68          PrintWriter out = response.getWriter();
69
70          out.println("<title>View Programs</title>");
71
72          request.getRequestDispatcher("billing.jsp").include(request, response);
73
74          //out.println("<h1 style=\"font-family:century gothic; color:#292F33;\">View Programs</h1>");
75          String phone = request.getParameter("phone");
76          List<Bill> list = SellerDao.showClientBill(phone);
77          out.print("<table>");
78          out.print("<tr><th>"+"Client's Number"+"</th><th>"+"Month"+"</th><th>"+"Price"+"</th></tr>");
79          for(Bill b:list){
80              out.println("<tr><td>"+b.getPhoneNumber()+"</td><td>"+b.getBillingMonth()+"</td><td>"+b.getPrice()+"</td>");
81
82              out.println("</tr>");
83          }
84          out.println("</table>");
85          out.close();
86      }
87  }
```

Η μέθοδος doPost χρησιμοποιείται για την έκδοση λογαριασμού του πελάτη.
(γρ. 72) Γίνεται η σύνδεση με την jsp σελίδα billing.
(γρ. 75) Παίρνει τα στοιχεία που εισάγει ο χρήστης στη βάση.
(γρ. 76) Δημιουργείται μία λίστα και καλείται η showClientBill από την SellerDao.
(γρ. 77-84) Εμφανίζονται τα επιθυμητά αποτελέσματα.