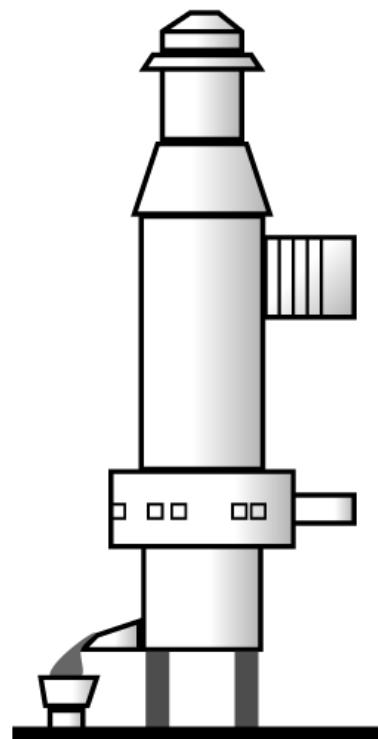


Istituto Tecnico Industriale
“G.B. Bosco Lucarelli”
82100 Benevento
Viale San Lorenzo



Benevento: 20/01/2022

Gagliarde Stefano _____
Giordano Luca _____
Gagliarde Giorgio Giuseppe _____

Traccia:

Creare una web-app mediante servlet Java per la gestione di un sito web dinamico su un tema a scelta. In particolare, prevedere:

- Una pagina di login con accesso controllato mediante autenticazione da parte dell'utente
- Una home page con contenuti personalizzati per ciascun utente o categoria di utenti che hanno accesso consentito al sito
- Una o più servizi a disposizione dell'utente (chat, sondaggi, form di registrazione, etc...) con altrettante pagine web dinamiche
- La connessione ad un database che consenta l'inserimento, la cancellazione, la modifica dei dati degli utenti, quali dati di accesso, generalità, numero di accessi, ecc..

Realizzare una possibile soluzione e argomentare la strategia risolutiva, le scelte implementative e progettuali con una relazione dettaglia.

LA NOSTRA WEB-APP:

La nostra web-app, ha la funzione di una rubrica telefonica online. Dopo essersi registrati, l'utente visualizzerà una rubrica, ovviamente, vuota dove potrà aggiungere dei contatti.

Successivamente l'utente potrà accedere alla propria rubrica tramite il “login”.

Abbiamo anche una parte Admin, con email = admin@gmail.com e password = admin123#, dove visualizzerà tutti gli utenti registrati, tutti i contatti inseriti e tutte le associazioni tra utente e contatti.

IMPLEMENTAZIONE:

Inizialmente abbiamo creato un database con 3 tabelle:

- Utenti: Per inserire tutti gli utenti che si sono registrati;
- Contatti: Per inserire tutti i contatti che ogni utente ha inserito nella propria rubrica;
- Link: Per inserire l'email dell'utente associato al numero del contatto aggiunto;

Per creare questa web-app abbiamo creato varie servlet, file .jsp e .html, classi java.

Per prima cosa abbiamo creato un *index.html* dove l'utente ha la scelta di registrarsi o loggarsi.

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="UTF-8">
    <title>Scelta</title>

    <link rel="stylesheet" href="style.css">
</head>
<body>

<div class="box">
    <h1>Ciao</h1>

    <a href=".reg-servlet">
        <button class="primo">Registrati</button>
    </a>

    <a href=".Login-servlet">
        <button class="secondo">Login</button>
    </a>
</div>
</body>
</html>
```



Ovviamente se per l'utente è la prima volta che accede al sito deve registrarsi.

Registrazione

Una volta cliccato il pulsante “Registrati” entra in gioco la prima servlet *Registrazione*

```
1 package com.example.appweb;
2
3 import java.io.*;
4 import java.sql.SQLException;
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.http.*;
8 import javax.servlet.annotation.*;
9
10
11 @WebServlet(name = "registrazione", value = "/reg-servlet")
12 public class Registrazione extends HttpServlet {
13
14     public void init() {}
15
16     @Override
17     public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
18
19         request.getRequestDispatcher("WEB-INF/registrazione.jsp").forward(request, response);
20     }
21 }
```

Dove utilizzo, inizialmente, il metodo doGet per rindirizzare l'utente nel form di *registrazione.jsp* tramite un dispatcher.

```

8  <%@ page contentType="text/html; charset=UTF-8" language="java" %>
9  <html>
10 <head>
11   <title>Registrazione</title>
12   <link rel="stylesheet" href=".//style.css">
13 </head>
14 <body>
15
16
17 <form class="box" action=".//reg-servlet" method="post">
18   <h1>Registrazione</h1>
19
20   <input type="text" name="Nome" placeholder="Nome *">
21   <input type="text" name="Cognome" placeholder="Cognome *">
22   <input type="text" name="Email" placeholder="Email *">
23   <input type="password" name="Password" placeholder="Password *">
24
25   <input type="submit" name="" value="Registrati">
26   <input type="reset" name="" value="Reset">

```

Faccio inserire Nome, Cognome, Email e Password e una volta cliccato il tasto “Registrati” mi ricollego alla servlet *Registrazione* e vado a fare vari controlli.

Utilizzando il metodo *doPost*, vado a salvare i dati inseriti nel form in 4 Stringhe (nome, cognome, email, pass), vado a crearmi una variabile contatore che vado a incrementare ogni volta che l’utente non rispetti i formati richiesti.

Poi vado a utilizzare la prima classe java, ovvero *UtenteDAO*, queste classi le vado a creare per richiamare i metodi che mi permettono di collegarmi al database e interagire con esso.

I primi due metodi che vado a richiamare sono *firstNameValidation* e *secondNameValidation* dove passo come parametro nome e cognome inserito nel form.

Se il nome e il cognome non rispettano il formato vado a creare due messaggi:

“messaggioNome” e “messaggioCognome”, dove devono iniziare e finire con una lettera e non devono superare i 25 caratteri e incremento il contatore.

```
22 @Override
23     public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
24         response.setContentType("text/html");
25         PrintWriter out = response.getWriter();
26
27         String nome = request.getParameter("Nome");
28         String cognome = request.getParameter("Cognome");
29         String email = request.getParameter("Email");
30         String pass = request.getParameter("Password");
31
32         int contatore=0;
33         UtenteDAO utenteDAO = new UtenteDAO();
34
35         if (UtenteDAO.firstNameValidation(nome)){
36             //
37         }else{
38             request.setAttribute("messaggioNome", "Inserisci un nome; *Deve Iniziare e Finire con un carattere " +
39             "\n*NON Deve superare i 25 caratteri");
40             contatore++;
41
42         if (UtenteDAO.secondNameValidation(cognome)){
43             //
44         }else{
45             request.setAttribute("messaggioCognome", "Inserisci un cognome; *Deve Iniziare e Finire con un carattere" +
46             "\n*NON Deve superare i 25 caratteri");
47             contatore++;
48         }
49     }

```

```
36
37     public static boolean firstNameValidation(String nome){
38
39         boolean valid = nome.matches(regex: "(?i)(^([a-z]+)[a-z .,-]((?! .,-)$){1,25}$");
40         return valid;
41     }
42
43     public static boolean secondNameValidation(String cognome){
44
45         boolean valid = cognome.matches(regex: "(?i)(^([a-z]+)[a-z .,-]((?! .,-)$){1,25}$");
46         return valid;
47     }

```

Poi vado a controllare anche email e password utilizzando sempre dei metodi della classe java *UtenteDAO*

Vado a controllare se l'email contiene "@" e se la password rispetta il formato

```
48     public static final Pattern VALID_PASS =
49         Pattern.compile( regex: "^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&+=])(?=\\s+$)", Pattern.CASE_INSENSITIVE);
50
51     public static boolean validatePass(String passStr){
52         Matcher matcher = VALID_PASS.matcher(passStr);
53         return matcher.find();
54     }
55
56
57     public static final Pattern VALID_EMAIL =
58         Pattern.compile( regex: "^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.\w{2,6}$", Pattern.CASE_INSENSITIVE);
59
60     public static boolean validate(String emailStr) {
61         Matcher matcher = VALID_EMAIL.matcher(emailStr);
62         return matcher.find();
63     }
64
65 }
```

Se email e password non sono corretti, nella servlet, vado a crearmi i messaggi e incremento il contatore.

```
49         if (UtenteDAO.validate(email)){
50             //
51         }else{
52             request.setAttribute( s: "messaggioEmail", o: "Email non valida !!!");
53             contatore++;
54         }
55
56
57         if (UtenteDAO.validatePass(pass)){
58             //
59         }else{
60             request.setAttribute(
61                 s: "messaggioPass", o: "Password non valida, riprova!! " +
62                 "\n"+ "* Deve contenere almeno un carattere minuscolo [a - z]" +
63                 "\n"+ "* Deve contenere almeno una cifra [0 - 9]" +
64                 "\n"+ "* Deve contenere almeno un carattere speciale [@#$%^&+=]" +
65                 "\n"+ "* Deve contenere almeno 8 caratteri" +
66                 "\n"+ "* NON deve contenere spazi bianchi"
67             );
68
69             contatore++;
70         }
71 }
```

Poi vado a controllare anche se l'email inserita nel form è già stata registrata nel database oppure non ancora, grazie al metodo *existEmail*.

Il metodo *existEmail* è il primo metodo che interagisce con il database.

Vado a connettermi tramite Connection e setto i Driver necessari:

L'URL del database, USER e PASS (password)

```
10  public class UtenteDAO {  
11  
12      private final String URL = "jdbc:mysql://localhost:3306/webapp";  
13      private final String USER = "root";  
14      private final String PASS = "gagliarde";  
15  
16  
17  }
```

Vado a crearmi lo statement dove gli passo la QUERY per interrogare il database: vado a selezionare tutti i campi della tabella *utenti* con email = all'email inserita nel form.

Setto la stringa con *statement.setString*

Con il ResultSet vado a salvarmi il risultato che mi restituisce la QUERY

Se il resulset mi ritorna qualcosa vuol dire che l'email è già stata registrata altrimenti l'email non esiste ancora.

```
80  
81  public boolean existEmail (String email) throws SQLException {  
82  
83      Connection connection = DriverManager.getConnection(URL, USER, PASS);  
84  
85      PreparedStatement statement = connection.prepareStatement(sql: "SELECT * FROM utenti WHERE email = ?");  
86  
87      statement.setString(parameterIndex: 1, email);  
88  
89      ResultSet resultSet = statement.executeQuery();  
90  
91      if (resultSet.next()){  
92          return true;  
93      }else {  
94          return false;  
95      }  
96  }
```

Quindi nella servlet mi creo una variabile di tipo booleano e vado a richiamare il metodo sopra elencato.

Se questa variabile assume valore: true, vuol dire che l'email è stata già registrata nel database, quindi, vado a creare un messaggio e incremento il contatore

```

71
72     try {
73
74         boolean ritorno = utenteDAO.existEmail(email);
75
76         if (ritorno){
77             request.setAttribute("messaggioExist", "Email già registrata");
78             contatore++;
79         }else{ }
80

```

Infine una volta finiti tutti i controlli per l'inserimento dei dati, vado a fare un ultimo controllo

```

82
83     if (contatore>0){
84
85         RequestDispatcher rd =request.getRequestDispatcher("WEB-INF/registrazione.jsp");
86         rd.forward(request, response);
87
88     }else {
89         utenteDAO.insert(nome, cognome, email, pass);
90
91         HttpSession sessione = request.getSession(true);
92
93         sessione.setAttribute("nome1", nome);
94         sessione.setAttribute("cognome1", cognome);
95         sessione.setAttribute("email1", email);
96         sessione.setAttribute("passwd1", pass);
97
98
99         request.setAttribute("messaggioReg", "Registrazione andata a buon fine !!");
100        RequestDispatcher rd2 = request.getRequestDispatcher("home-servlet");
101        rd2.forward(request, response);
102    }
103
104 } catch (SQLException e) {
105     e.printStackTrace();
106 }

```

Se il contatore assume valore maggiore di 0, vuol dire che l'utente ha inserito dei dati che non rispettano il formato richiesto, che abbiamo visto prima, quindi l'utente non può registrarsi.

Lo rindirizzo al form *registrazione.jsp*, tramite il dispatcher, e gli faccio visualizzare i messaggi che corrispondono ai dati non inseriti correttamente.

ES:

REGISTRAZIONE

Nome *

Cognome *

Email *

Password *

Registrati

Reset

This image shows a registration form on a black background. It consists of four input fields: 'Nome *', 'Cognome *', 'Email *', and 'Password *'. Below these fields are two buttons: a green rounded rectangle labeled 'Registrati' and a white rounded rectangle labeled 'Reset'.

Questo è come si presenta il form di registrazione

Se clicco “Registrati” e non vado a inserire nessun campo, il risultato sarà questo:

REGISTRAZIONE

Nome *

Cognome *

Email *

Password *

Registrati

Reset

Inserisci un nome; *Deve Iniziare e Finire con un carattere
*NON Deve superare i 25 caratteri

Inserisci un cognome; *Deve Iniziare e Finire con un carattere
*NON Deve superare i 25 caratteri

Email non valida !!

Password non valida, riproval! * Deve contenere almeno un carattere minuscolo [a - z] * Deve contenere almeno una cifra [0 - 9] * Deve contenere almeno un carattere speciale [#@%\$%^&+=] * Deve contenere almeno 8 caratteri * NON deve contenere spazi bianchi

This image shows the same registration form as above, but with validation errors displayed in red text below the buttons. The errors are: 'Inserisci un nome; *Deve Iniziare e Finire con un carattere *NON Deve superare i 25 caratteri', 'Inserisci un cognome; *Deve Iniziare e Finire con un carattere *NON Deve superare i 25 caratteri', 'Email non valida !!', and 'Password non valida, riproval! * Deve contenere almeno un carattere minuscolo [a - z] * Deve contenere almeno una cifra [0 - 9] * Deve contenere almeno un carattere speciale [#@%\$%^&+=] * Deve contenere almeno 8 caratteri * NON deve contenere spazi bianchi'.

ES 2:

Se invece compilo correttamente solo i due campi e clicco “Registrati”

REGISTRAZIONE

Stefano

Gagliarde

Email *

Password *

Registrati

Reset

Il risultato sarà questo:

REGISTRAZIONE

Nome *

Cognome *

Email *

Password *

Registrati

Reset

Email non valida !!

Password non valida, riprova!! * Deve contenere almeno un carattere minuscolo [a - z] * Deve contenere almeno una cifra [0 - 9] * Deve contenere almeno un carattere speciale [@#\$%^&+=] * Deve contenere almeno 8 caratteri * NON deve contenere spazi bianchi

Questi messaggi li gestisco in *registrazione.jsp*

```
27
28     <% if (request.getAttribute("messaggioNome")!=null){ %>
29         <div class="mess"> <%= request.getAttribute("messaggioNome")%> </div>
30     <% } %>
31
32
33     <% if (request.getAttribute("messaggioCognome")!=null){ %>
34         <div class="mess"> <%= request.getAttribute("messaggioCognome")%> </div>
35     <% } %>
36
37     <% if (request.getAttribute("messaggioEmail")!=null){ %>
38         <div class="mess"> <%= request.getAttribute("messaggioEmail")%> </div>
39     <% } %>
40
41     <% if (request.getAttribute("messaggioPass")!=null){ %>
42         <div class="mess"> <%= request.getAttribute("messaggioPass")%> </div>
43     <% } %>
44
45
46     <% if (request.getAttribute("messaggioExist")!=null){ %>
47         <div class="mess"> <%= request.getAttribute("messaggioExist")%> </div>
48     <% } %>
```

Se il messaggio è diverso da null, vuol dire che l'utente ha sbagliato in quel campo associato al messaggio, quindi, vado a stampare quel messaggio.

Questo metodo lo vado a utilizzare per ogni messaggio

Ritornando all'ultimo controllo

```
82
83     if (contatore>0){
84
85         RequestDispatcher rd =request.getRequestDispatcher( s: "WEB-INF/registrazione.jsp")
86         rd.forward(request, response);
87
88     }else {
89         utenteDAO.insert(nome, cognome, email, pass);
90
91
92         HttpSession sessione = request.getSession( b: true);
93
94         sessione.setAttribute( s: "nome1", nome);
95         sessione.setAttribute( s: "cognome1", cognome);
96         sessione.setAttribute( s: "email1", email);
97         sessione.setAttribute( s: "passwd1", pass);
98
99
100        request.setAttribute( s: "messaggioReg", o: "Registrazione andata a buon fine !!");
101        RequestDispatcher rd2 = request.getRequestDispatcher( s: "home-servlet");
102        rd2.forward(request, response);
103
104    }
105
106 } catch (SQLException e) {
107     e.printStackTrace();
108 }
```

Se, invece, il contatore NON è maggiore di 0 vuol dire che i campi inseriti sono corretti.

Vado a chiamare il metodo *insert* che mi permette di inserire l'utente nel database, passandogli come parametri nome, cognome, email, pass:

```
66
67     public void insert (String nome, String cognome, String email, String password) throws SQLException {
68
69         Connection connection = DriverManager.getConnection(URL, USER, PASS);
70         PreparedStatement statement = connection.prepareStatement( sql: "INSERT INTO utenti VALUE (?, ?, ?, ?)");
71
72         statement.setString( parameterIndex: 1, nome);
73         statement.setString( parameterIndex: 2, cognome);
74         statement.setString( parameterIndex: 3, email);
75         statement.setString( parameterIndex: 4, password);
76
77         statement.executeUpdate();
78
79 }
```

Mi creo la sessione e vado a salvare questi campi, creo il messaggio di “Registrazione andata a buon fine” e eseguo un dispatcher verso la servlet *Home*, che vedremo in seguito dopo aver visto l’implementazione per la login.

```
3
4
5         RequestDispatcher rd = request.getRequestDispatcher( <= "WEB-INF/registrazione.jsp"
6             rd.forward(request, response);
7
8     } else {
9         utenteDAO.insert(nome, cognome, email, pass);
10
11
12         HttpSession sessione = request.getSession( b: true);
13
14         sessione.setAttribute( <= "nome1", nome);
15         sessione.setAttribute( <= "cognome1", cognome);
16         sessione.setAttribute( <= "email1", email);
17         sessione.setAttribute( <= "passwd1", pass);
18
19
20         request.setAttribute( <= "messaggioReg", o: "Registrazione andata a buon fine !!");
21         RequestDispatcher rd2 = request.getRequestDispatcher( <= "home-servlet");
22         rd2.forward(request, response);
23     }
24
25 } catch (SQLException e) {
26     e.printStackTrace();
27 }
```

Login

Una volta registratosi, le prossime volte che l'utente accede al sito deve ovviamente loggarsi.

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="UTF-8">
    <title>Scelta</title>

    <link rel="stylesheet" href="style.css">
</head>
<body>

<div class="box">
    <h1>Ciao</h1>

    <a href=".//reg-servlet">
        <button class="primo">Registrati</button>
    </a>

    <a href=".//login-servlet">
        <button class="secondo">Login</button>
    </a>
</div>

</body>
</html>
```



Cliccando il tasto “Login” vado a chiamare la servlet *Login*

```
1  package com.example.appweb;
2
3  import java.io.*;
4  import java.sql.SQLException;
5  import javax.servlet.RequestDispatcher;
6  import javax.servlet.ServletException;
7  import javax.servlet.http.*;
8  import javax.servlet.annotation.*;
9
10 @WebServlet(name = "login", value = "/login-servlet")
11 public class Login extends HttpServlet {
12
13     public void init() {}
14
15     @Override
16     public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
17
18         request.getRequestDispatcher("WEB-INF/login.jsp").forward(request, response);
19     }
}
```

Con il metodo doGet rindirizzo l'utente in login.jsp tramite un dispatcher

```
7  <%@ page contentType="text/html; charset=UTF-8" language="java" %>
8  <html>
9   <head>
10    <title>Login</title>
11
12    <link rel="stylesheet" href="./style.css">
13
14  </head>
15  <body>
16
17
18    <form class="box" action="./login-servlet" method="post">
19      <h1>Login</h1>
20
21      <input type="email" name="Email" placeholder="Email">
22      <input type="password" name="Password" placeholder="Password">
23
24      <a href="./CambiaPasswd">Vuoi cambiare la password?</a>
25
26      <input type="submit" name="" value="Login">
27
```

Faccio inserire email e password. L'utente avrà la possibilità di loggarsi o di cambiare la propria password.

Se viene cliccato il pulsante “Login” vado a richiamare la servlet *Login*

```
20
21  public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {
22
23    response.setContentType("text/html");
24    PrintWriter out = response.getWriter();
25
26    String email = request.getParameter("Email");
27    String pass = request.getParameter("Password");
28
29    UtenteDAO utenteDAO = new UtenteDAO();
30
31    String emailAdmin = "admin@gmail.com";
32    String passwdAdmin = "admin123#";
```

Con il metodo doPost vado a salvare i dati inseriti nel form in due stringhe (email, pass), creo un oggetto di tipo UtenteDAO sempre per utilizzare alcuni metodi, che abbiamo visto già nella registrazione, per gestire i dati e per interagire con il database.

Vado a creare due Stringhe contenenti email e password per entrare nella parte admin, che vedremo in seguito.

```

34
35     if (email.equals(emailAdmin) && pass.equals(passwdAdmin)){
36
37         RequestDispatcher dispatcher = request.getRequestDispatcher("homeAdmin-servlet");
38         dispatcher.forward(request, response);
39
40     }else{
41
42         boolean ritorno = utenteDAO.exist(email,pass);
43
44         if (ritorno){
45
46             HttpSession sessione = request.getSession(true);
47             sessione.setAttribute("passwd1",pass);
48             sessione.setAttribute("email1",email);
49             RequestDispatcher dispatcher = request.getRequestDispatcher("home-servlet");
50             dispatcher.forward(request, response);
51
52         }else{
53             request.setAttribute("messaggioLogin", "Email o Password errati !!");
54             RequestDispatcher rd2 = request.getRequestDispatcher("WEB-INF/login.jsp");
55             rd2.forward(request, response);
56         }
57
58     }
59
60 }
61 } catch (SQLException e) {
62     e.printStackTrace();

```

Nel primo *if* vado a controllare se l'email e la password inserite nel form sono dell'admin o dell'utente

Se sono quelle dell'admin allora vado a chiamare la servlet *Admin*.

Altrimenti vado a creare una variabile di tipo booleana che mi salva il risultato del metodo *exist*.

Il metodo *exist* mi permette di verificare se l'email e la password inserita nel form di login combaciano e se sono presenti nel database

```

16
17     public boolean exist (String email, String password) throws SQLException {
18
19         Connection connection = DriverManager.getConnection(URL, USER, PASS);
20
21         PreparedStatement statement = connection.prepareStatement("SELECT * FROM utenti WHERE email = ? AND passwd = ?");
22
23         statement.setString(parameterIndex: 1, email);
24         statement.setString(parameterIndex: 2, password);
25
26         ResultSet resultSet= statement.executeQuery();
27
28         if (resultSet.next()){
29
30             return true;
31
32         }else {
33             return false;
34
35     }

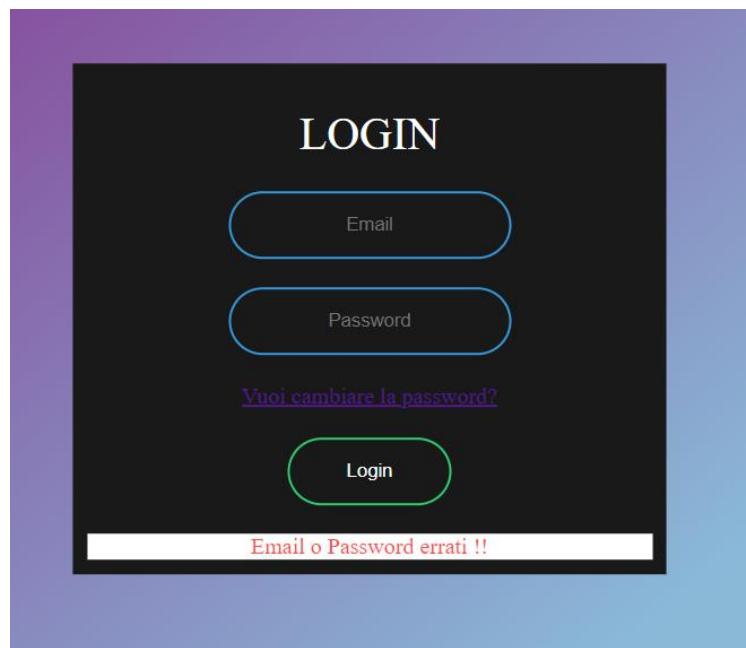
```

Mi connetto al database, creo lo statement con la query, setto le stringhe, eseguo la query, se mi ritorna: true vuol dire che l'email e la password combaciano altrimenti no.

Quindi se l'email e la password sono giuste vado a creare la sessione, salvo nella sessione sia l'email che la password e vado a chiamare la servlet *Home* tramite un dispatcher.

```
34
35         if (email.equals(emailAdmin) && pass.equals(passwdAdmin)){
36
37             RequestDispatcher dispatcher = request.getRequestDispatcher("homeAdmin-servlet");
38             dispatcher.forward(request, response);
39
40         }else{
41
42             boolean ritorno = utenteDAO.exist(email,pass);
43
44             if (ritorno){
45
46                 HttpSession sessione = request.getSession(true);
47                 sessione.setAttribute("passwd1",pass);
48                 sessione.setAttribute("email1",email);
49                 RequestDispatcher dispatcher = request.getRequestDispatcher("home-servlet");
50                 dispatcher.forward(request, response);
51
52             }else{
53                 request.setAttribute("messaggioLogin", "Email o Password errati !!");
54                 RequestDispatcher rd2 = request.getRequestDispatcher("WEB-INF/Login.jsp");
55                 rd2.forward(request, response);
56             }
57         }
58     }
```

Altrimenti creo il messaggio “Email o Password errati” e rindirizzo l'utente sul form di login con il messaggio



che vado a gestire sempre nel file login.jsp

```
27
28
29      <% if (request.getAttribute("messaggioLogin")!=null){%>
30          <div class="mess"> <%= request.getAttribute("messaggioLogin")%> </div>
31      <%}%>
32
33      <% if (request.getAttribute("cambioPasswd")!=null){%>
34          <div class="mess messCheck"> <%= request.getAttribute("cambioPasswd")%> </div>
35      <%}%>
36  </form>
```

L'Utente potrà anche cambiare la propria password cliccando sul link “Vuoi cambiare la password?”

```
7  <!-->
8  <%@ page contentType="text/html;charset=UTF-8" language="java" %>
9  <html>
10 <head>
11     <title>Login</title>
12
13     <link rel="stylesheet" href="./style.css">
14 </head>
15 <body>
16
17
18     <form class="box" action=".//login-servlet" method="post">
19         <h1>Login</h1>
20
21         <input type="email" name="Email" placeholder="Email">
22         <input type="password" name="Password" placeholder="Password">
23
24         <a href=".//CambiaPasswd">Vuoi cambiare la password?</a>
25
26         <input type="submit" name="" value="Login">
```

Se viene selezionata questa scelta vado a chiamare la servlet *CambiaPasswd*

```
1  package com.example.appweb;
2
3  import ...
4
5
6  @WebServlet(name = "CambiaPasswd", value = "/CambiaPasswd")
7  public class CambiaPasswd extends HttpServlet {
8
9
10     public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException{
11
12         request.getRequestDispatcher( s: "WEB-INF/ChangePass.jsp").forward(request, response);
13
14     }
15
16 }
```

Anche qui vado a utilizzare un dispatcher che porta l'utente al form per cambiare la propria password

```

8   <%@ page contentType="text/html; charset=UTF-8" language="java" %>
9   <html>
10  <head>
11    <title>Cambia Password</title>
12    <link rel="stylesheet" href=".//style.css">
13  </head>
14  <body>
15
16
17  <form class="box" action="CambiaPasswd" method="post">
18    <h1>Cambia Password</h1>
19
20    <input type="email" name="email" placeholder="Email">
21    <input type="password" name="newPasswd" placeholder="Nuova Password">
22    <input type="password" name="checkPasswd" placeholder="Conferma Password">
23
24    <input type="submit" name="" value="Cambia Password">
25

```

Faccio inserire l'email dell'utente, la nuova password e la conferma. Una volta cliccato il pulsante “Cambia Password” vado a richiamare la servlet *CambiaPasswd*

Dove vado a salvare i dati inseriti nelle Stringhe, creo un oggetto di tipo UtenteDAO

```

18 ⏵ @ protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
19
20   String email = request.getParameter("email");
21   String newPass = request.getParameter("newPasswd");
22   String checkPass = request.getParameter("checkPasswd");
23
24   UtenteDAO utenteDAO = new UtenteDAO();
25

```

Anche qui vado a controllare se l'email e la nuova password rispettano il formato, utilizzando i metodi che abbiamo utilizzato per la registrazione.

```

26     try {
27         int contatore = 0;
28
29         if (UtenteDAO.validate(email)){
30             //
31         }else {
32             request.setAttribute("messaggioEmail", "Email non valida !!");
33             contatore++;
34         }
35
36
37         if (UtenteDAO.validatePass(newPass)){
38             //
39         }else{
40             request.setAttribute(
41                 "messaggioPass", "Password non valida, riprova!! " +
42                 "\n"+ "* Deve contenere almeno un carattere minuscolo [a - z]" +
43                 "\n"+ "* Deve contenere almeno una cifra [0 - 9]" +
44                 "\n"+ "* Deve contenere almeno un carattere speciale [#@${%^&+=}]" +
45                 "\n"+ "* Deve contenere almeno 8 caratteri" +
46                 "\n"+ "* NON deve contenere spazi bianchi"
47             );
48
49             contatore++;
50         }

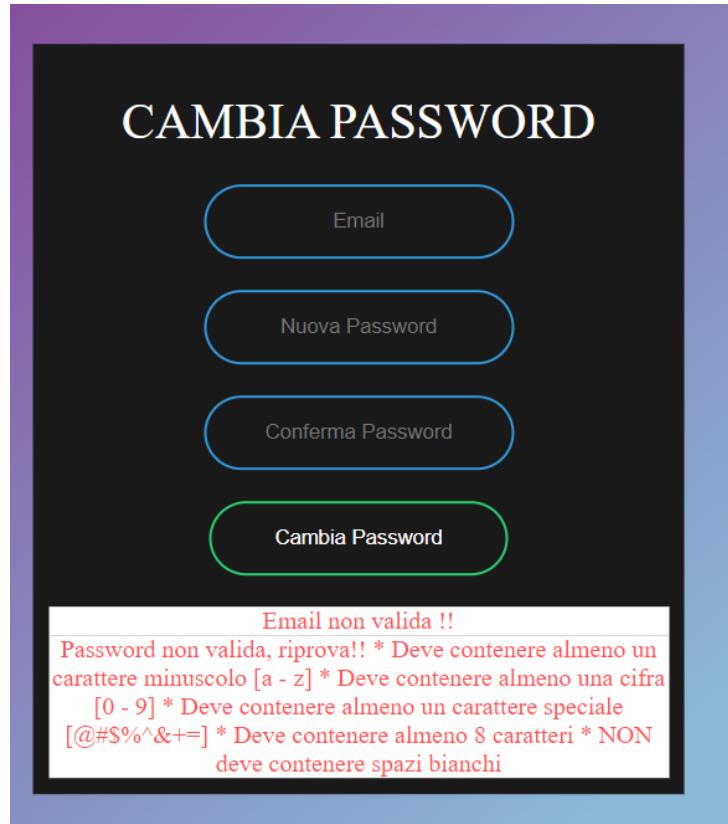
```

Anche qui vado a utilizzare il contatore. Se il contatore è maggiore di 0 vuol dire che l'email e la nuova password non rispettano il formato e lo rindirizzo al form ChangePass.jsp con i messaggi dell'eventuale errore.

```

52     if (contatore > 0){
53
54         RequestDispatcher rd =request.getRequestDispatcher("WEB-INF/ChangePass.jsp");
55         rd.forward(request, response);
56
57     }else {
58
59         boolean ritorno = utenteDAO.existEmail(email);
60         if (ritorno) {
61
62             if (checkPass.equals(newPass)) {
63
64                 utenteDAO.changePass(newPass, email);
65
66                 request.setAttribute("cambioPasswd", "Password cambiata");
67                 RequestDispatcher dispatcher = request.getRequestDispatcher("WEB-INF/Login.jsp");
68                 dispatcher.forward(request, response);
69
70             } else {
71
72                 request.setAttribute("checkPasswd", "Le password non combaciano");
73                 RequestDispatcher dispatcher = request.getRequestDispatcher("WEB-INF/ChangePass.jsp");
74                 dispatcher.forward(request, response);
75
76             }
77
78         }else {
79
80             request.setAttribute("existEmail", "Email non trovata");
81             RequestDispatcher dispatcher = request.getRequestDispatcher("WEB-INF/ChangePass.jsp");
82             dispatcher.forward(request, response);

```



Se non ci sono errori allora vado a fare altri controlli.

```

1 if (contatore > 0){
2
3     RequestDispatcher rd = request.getRequestDispatcher( "/WEB-INF/ChangePass.jsp");
4     rd.forward(request, response);
5
6 } else {
7
8     boolean ritorno = utenteDAO.existEmail(email);
9     if (ritorno) {
10
11         if (checkPass.equals(newPass)) {
12
13             utenteDAO.changePass(newPass, email);
14
15             request.setAttribute( "cambioPassato", true );
16             RequestDispatcher dispatcher = request.getRequestDispatcher( "/WEB-INF/login.jsp");
17             dispatcher.forward(request, response);
18
19         } else {
20
21             request.setAttribute( "checkPassed", false );
22             RequestDispatcher dispatcher = request.getRequestDispatcher( "/WEB-INF/ChangePass.jsp");
23             dispatcher.forward(request, response);
24
25         }
26
27     } else {
28
29         request.setAttribute( "existEmail", false );
30         RequestDispatcher dispatcher = request.getRequestDispatcher( "/WEB-INF/ChangePass.jsp");
31         dispatcher.forward(request, response);
32
33     }
34
35 }
```

Vado a richiamare il metodo *existEmail* (mi va a controllare se l'Email inserita nel form è presente nel database oppure no), se l'email esiste vado a controllare se la nuova password e la conferma password sono identiche: se sono identiche vado a chiamare il metodo *changePass* (va a sostituire la password nel database)

```

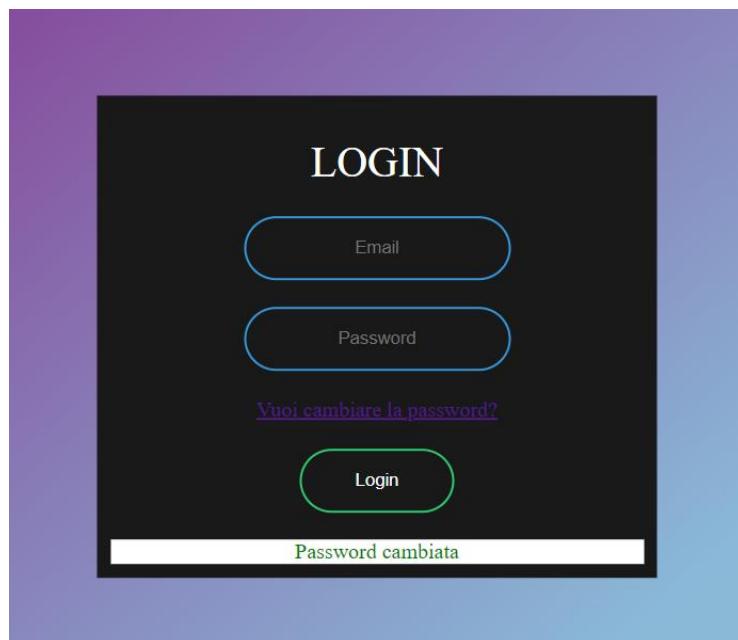
public void changePass (String NuovaPasswd, String email) throws SQLException{
    Connection connection = DriverManager.getConnection(URL, USER, PASS);
    PreparedStatement statement = connection.prepareStatement( sql: "UPDATE utenti SET passwd = ? WHERE email = ?");

    statement.setString( parameterIndex: 1, NuovaPasswd);
    statement.setString( parameterIndex: 2, email);

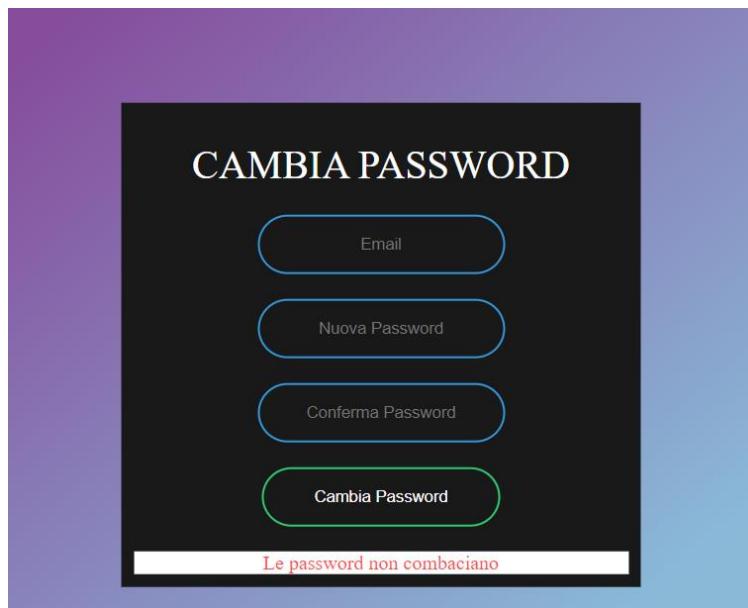
    statement.executeUpdate();
}

```

e mando l'utente nel form di login per poter accedere con la nuova password cambiata. E faccio visualizzare il messaggio “Password cambiata”

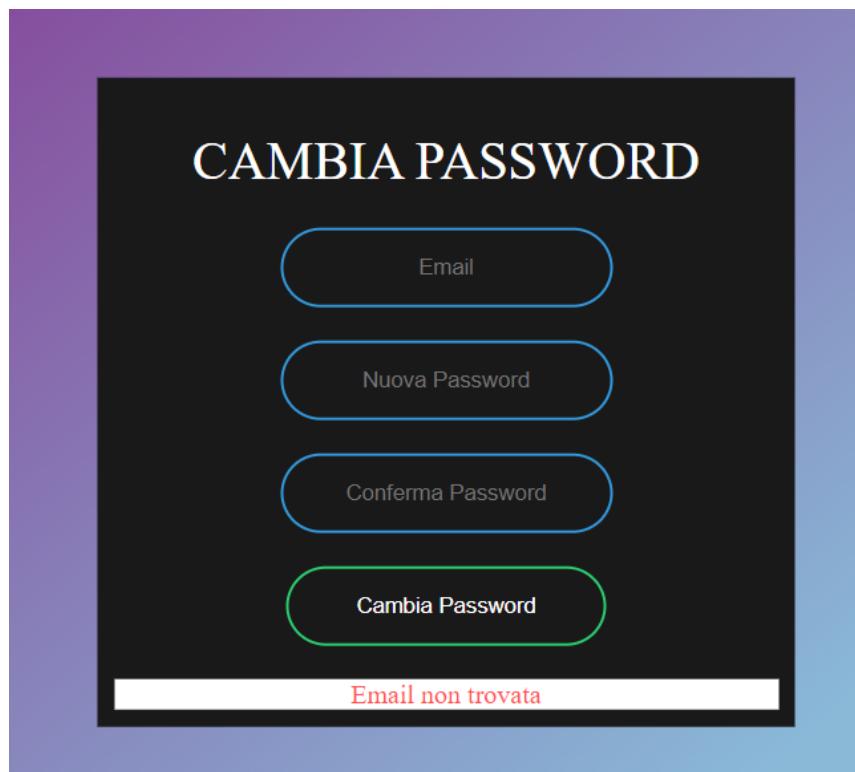


Altrimenti se la password di conferma e la nuova password non combaciano creo il messaggio “Le password non combaciano” e faccio visualizzare il messaggio nel form per cambiare la password



Altrimenti se l'email non esiste, vado a creare direttamente il messaggio "Email non trovata" e lo faccio visualizzare nel form.

```
52     if (contatore > 0){  
53  
54         RequestDispatcher rd =request.getRequestDispatcher( "/WEB-INF/ChangePass.jsp");  
55         rd.forward(request, response);  
56     }else {  
57  
58         boolean ritorno = utenteDAO.existEmail(email);  
59         if (ritorno) {  
60  
61             if (checkPass.equals(newPass)) {  
62  
63                 utenteDAO.changePass(newPass, email);  
64  
65                 request.setAttribute( "cambioPassato", "Password cambiata");  
66                 RequestDispatcher dispatcher = request.getRequestDispatcher( "/WEB-INF/login.jsp");  
67                 dispatcher.forward(request, response);  
68             } else {  
69  
70                 request.setAttribute( "checkPassed", "Le password non coincidono");  
71                 RequestDispatcher dispatcher = request.getRequestDispatcher( "/WEB-INF/ChangePass.jsp");  
72                 dispatcher.forward(request, response);  
73             }  
74         } else {  
75  
76             request.setAttribute( "existEmail", "Email non trovata");  
77             RequestDispatcher dispatcher = request.getRequestDispatcher( "/WEB-INF/ChangePass.jsp");  
78             dispatcher.forward(request, response);  
79         }  
80     }  
81 }  
82 }
```



Tutti questi messaggi li gestisco nel form *ChangePass.jsp*

```
27
28      <% if (request.getAttribute("messaggioEmail")!=null){ %>
29      <div class="mess"> <%= request.getAttribute("messaggioEmail")%> </div>
30      <% } %>
31
32      <% if (request.getAttribute("messaggioPass")!=null){ %>
33      <div class="mess"> <%= request.getAttribute("messaggioPass")%> </div>
34      <% } %>
35
36
37      <% if(request.getAttribute("cambioPasswd")!=null){%>
38      <div class="mess"> <%= request.getAttribute("cambioPasswd")%> </div>
39      <%}%>
40
41      <% if (request.getAttribute("checkPasswd")!=null){%>
42      <div class="mess"> <%= request.getAttribute("checkPasswd")%> </div>
43      <%}%>
44
45
46      <% if (request.getAttribute("existEmail")!=null){%>
47      <div class="mess"> <%= request.getAttribute("existEmail")%> </div>
48      <%}%>
```

Invece il messaggio “Password cambiata” la gestisco nel form di *login.jsp*

```
<% if (request.getAttribute("cambioPasswd")!=null){%>
    <div class="mess messCheck"> <%= request.getAttribute("cambioPasswd")%> </div>
<%}%>
```

Home

La servlet *Home* viene utilizzata sia se l'utente ha eseguito la registrazione che il login

```
22  public void doPost(HttpServletRequest request, HttpServletResponse response) {  
23  
24      response.setContentType("text/html");  
25  
26      ContattiDAO contattiDAO = new ContattiDAO();  
27      UtenteDAO utenteDAO = new UtenteDAO();  
28  
29  
30      HttpSession sessione = request.getSession( b: false);  
31  
32      try {  
33  
34          if (sessione.getAttribute( s: "nome1") == null & sessione.getAttribute( s: "cognome1") == null){  
35  
36              Utente utente = utenteDAO.PrendiUtente((String) sessione.getAttribute( s: "email1"));  
37              sessione.setAttribute( s: "nome1",utente.getNome());  
38              sessione.setAttribute( s: "cognome1",utente.getCognome());  
39  
40          }else {  
41              //  
42          }  
43  
44      } catch (SQLException e) {  
45          e.printStackTrace();  
46      }  
47  }  
48  
49
```

Inizialmente vado a creare un oggetto di tipo UtenteDAO (utilizzato già in precedenza) e un oggetto di tipo ContattiDAO utilizzato per gestire i contatti inseriti da ogni utente nella rubrica, che vedremo in seguito.

Poi vado a controllare se l'utente ha eseguito la registrazione o il login:

se il nome e il cognome hanno valore =null vuol dire che l'utente ha eseguito il login, perché inserendo solo email e password la sessione non ha salvato il nome e il cognome, quindi, vado a chiamare il metodo *PrendiUtente* (mi permette di prendere il nome e il cognome associato all'email)

```
98  public Utente PrendiUtente(String email) throws SQLException {  
99  
100      Connection connection = DriverManager.getConnection(URL, USER, PASS);  
101      PreparedStatement statement = connection.prepareStatement( sql: "SELECT nome, cognome FROM utenti WHERE email = ? ");  
102  
103      statement.setString( parameterIndex: 1, email);  
104  
105      ResultSet resultSet = statement.executeQuery();  
106      if(resultSet.next())  
107          return new Utente(resultSet.getString( columnIndex: 1), resultSet.getString( columnIndex: 2));  
108      else return null;  
109  }
```

e vado a utilizzare la classe *Utente*

```
1 package com.example.appweb;
2
3 public class Utente {
4     private String nome;
5     private String cognome;
6
7
8     public Utente(String nome, String cognome){
9
10         this.nome = nome;
11         this.cognome = cognome;
12
13     }
14
15     public String getNome() { return nome; }
16
17     public String getCognome() { return cognome; }
18
19     public String toString() { return nome+" "+cognome; }
20
21
22
23
24
25
26
```

Creando un costruttore che mi va a salvare i dati presi dal ResultSet, quindi nome e cognome e infine vado a salvarli nella sessione

Sempre nella serlvet *Home* vado a crearmi una lista contenente tutti i contatti inseriti da un utente

```
49
50     try {
51
52         List<Contatto> ritornoList = contattiDAO.visualizza((String) sessione.getAttribute("email1"));
53
54         request.setAttribute("contattoList", ritornoList);
55         RequestDispatcher dispatcher = request.getRequestDispatcher("WEB-INF/home.jsp");
56         dispatcher.forward(request, response);
57
58
59     } catch (ServletException e) {
60         e.printStackTrace();
61     } catch (SQLException e) {
62         e.printStackTrace();
63     } catch (IOException e) {
64         e.printStackTrace();
65     }
66
67 }
```

Creando un oggetto di tipo List<Contatto> e salvare il contenuto del metodo *visualizza* e faccio visualizzare l'intera lista nella *home.jsp*

Nella classe ContattiDAO creo dei metodi per gestire i contatti nel database.

Utilizzando la tabella *link* grazie all'associazione tra nTelefonico della tabella link e nTelefonico della tabella contatti associati all'email dell'utente, vado a creare un metodo di tipo List contenente i dati dei contatti inseriti da un utente.

```
68
69     public List<Contatto> visualizza(String email) throws SQLException {
70
71         Connection connection = DriverManager.getConnection(URL, USER, PASS);
72         PreparedStatement statement = connection.prepareStatement(sql: "SELECT * FROM link JOIN contatti ON link.nTelefonico = contatti.nTelefonico WHERE email = ? ");
73
74         statement.setString( parameterIndex: 1, email);
75         statement.executeQuery();
76
77         ResultSet resultSet = statement.getResultSet();
78
79         List<Contatto> contattoList = new ArrayList<>();
80
81
82         while (resultSet.next()){
83
84             Contatto contatto = new Contatto( resultSet.getString( columnIndex: 3),
85                                         resultSet.getString( columnIndex: 4), resultSet.getString( columnIndex: 5));
86
87             contattoList.add(contatto);
88         }
89
90
91     }  
}
```

Vado a utilizzare la classe Contatto, creando un costruttore, per salvare il nome, il cognome e il numero telefonico di ogni contatto dell'utente.

```
1  package com.example.appweb;
2
3  public class Contatto {
4
5      private String nome, cognome, nTelefonico;
6
7      public Contatto (String nome, String cognome, String nTelefonico){
8
9          this.nome = nome;
10         this.cognome = cognome;
11         this.nTelefonico = nTelefonico;
12     }
13
14
15     public String getNome(){return nome;}
16
17     public String getCognome(){return cognome;}
18
19     public String getnTelefonico(){return nTelefonico;}
20
21     public String toString(){return nome +" "+ cognome+" "+ nTelefonico;}
22 }
23
```

E infine vado ad aggiungere il contenuto del costruttore *Contatto* alla lista generale

Questa è come si presenterà la home appena registrati, ovviamente vuota

Ciao, Stefano Gagliarde

Registrazione andata a buon fine !!

Questa è la tua rubrica:

Inserisci persone
nella rubrica

Nome	Cognome	Numero Telefonico	Elimina
------	---------	-------------------	---------

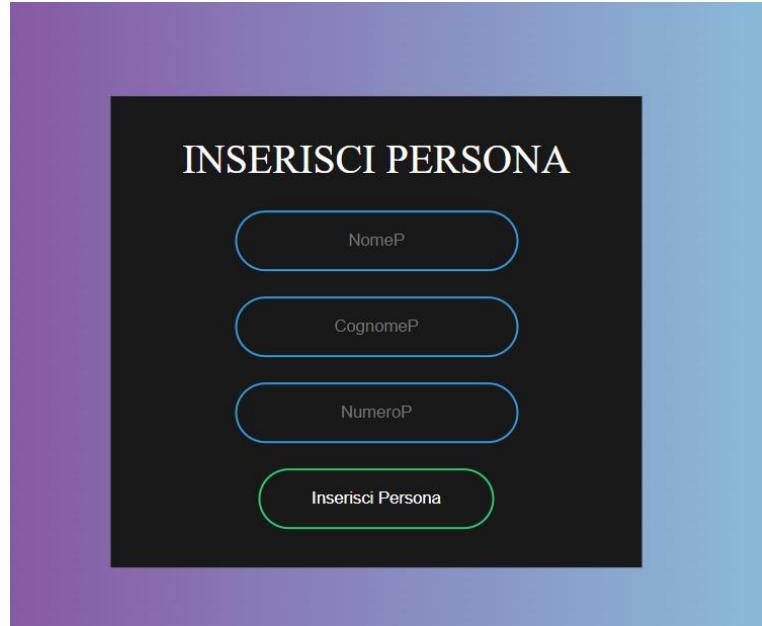
```
65 <%
66   List<Contatto> lista = (ArrayList<Contatto>) request.getAttribute("contattolist");
67
68 %>
69
70 <div>
71   <a href=".//inserisci.html">
72     <button class="button"> Inserisci persone nella rubrica </button>
73   </a>
74 </div>
75 <table>
76   <tr>
77     <th>Nome</th>
78     <th>Cognome</th>
79     <th>Numero Telefonico</th>
80     <th>Elimina</th>
81   </tr>
82
83   <% for (int i=0; i<lista.size(); i++){ %>
84     <tr>
85       <td><%= lista.get(i).getNome() %></td>
86       <td><%= lista.get(i).getCognome()%></td>
87       <td><%= lista.get(i).getnTelefonico()%></td>
88       <td> <a href="CancellaContattoServlet?nTelefonico=<%=lista.get(i).getnTelefonico()%>">  </a></td>
89     </tr>
90   <%} %>
91 </table>
92 </table>
```

Prima cosa che ho fatto è stato quello di creare una variabile di tipo `List<Contatto>` dove ho salvato la lista presa dalla servlet `Home`

Vado a creare la tabella per contenere i dati dei contatti.

Per scomporre la tabella ho utilizzato un ciclo `for`, aiutandomi con `.size()` (restituisce il conteggio degli elementi presenti nella lista). Con questo `for` vado a prendere uno a uno tutti i dati dei contatti, utilizzando i metodi della classe `Contatto` (`getNome`, `getCognome`, `getnTelefonico`) e me li vado a stampare nella tabella.

L'utente potrà inserire un contatto tramite il pulsante “Inserisci persona nella rubrica”, apparirà un form dove dovrà inserire nome, cognome e numero telefonico del contatto.



Dopo inseriti i dati vado a chiamare la servlet *InserisciPersone*.

Vado a salvare i dati inseriti nel form nelle variabili

Creo un oggetto di tipo ContattiDAO e vado a richiamare la sessione.

```
20
21 @Override
22     public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException{
23
24         response.setContentType("text/html");
25
26         String nomeP = request.getParameter("NomeP");
27         String cognomeP = request.getParameter("CognomeP");
28         String numeroP = request.getParameter("NumeroP");
29
30         ContattiDAO contattiDAO = new ContattiDAO();
31
32         HttpSession sessione = request.getSession(false);
```

Poi vado a fare dei controlli sul contatto aggiunto

```
32     try {
33
34         boolean ritornoLink = contattiDAO.existLink((String) sessione.getAttribute( "email1"), numeroP);
35
36         if (ritornoLink){
37             request.setAttribute( "messaggioExistLink", "Numero già registrato !");
38         }else {
39
40             boolean ritorno = contattiDAO.existNum( numeroP);
41
42             if (ritorno){
43
44                 contattiDAO.inserisciLink((String) sessione.getAttribute( "email1"), numeroP);
45
46             }else {
47                 contattiDAO.inserisci(nomeP, cognomeP, numeroP);
48                 contattiDAO.inserisciLink((String) sessione.getAttribute( "email1"), numeroP);
49             }
50
51         }
52
53
54         RequestDispatcher dispatcher = request.getRequestDispatcher( "home-servlet");
55         dispatcher.forward(request, response);
56     }
```

Creo una variabile di tipo boolean (ritornoLink) dove vado a salvare il risultato del metodo *existLink* (mi permette di controllare se il numero telefonico è già registrato per quell'utente oppure no).

```
36     public boolean existLink(String TuaEmail, String nTelefonico)throws SQLException{
37
38         Connection connection = DriverManager.getConnection(URL, USER, PASS);
39         PreparedStatement statement = connection.prepareStatement( sql: "SELECT * FROM link WHERE email = ? AND nTelefonico = ?");
40
41         statement.setString( parameterIndex: 1, TuaEmail);
42         statement.setString( parameterIndex: 2, nTelefonico);
43
44
45         ResultSet resultSet = statement.executeQuery();
46
47         if (resultSet.next()){
48             return true;
49         }else {
50             return false;
51         }
52     }
```

Se la variabile mi ritorna valore: true, vuol dire che il numero inserito si trova già nella rubrica di quell'utente, quindi, creo il messaggio “Numero già registrato” e lo vado a stampare nel file *home.jsp*

```
34
35     <% if (request.getAttribute("messaggioExistLink")!=null){%>
36         <%= request.getAttribute("messaggioExistLink")%>
37     <% } %>
38
```

Ciao, Stefano Gagliarde

Numero già registrato !

Questa è la tua rubrica:

Inserisci persone
nella rubrica

Nome	Cognome	Numero Telefoni
giorgio	gagliarde	333444555
Luca	Giordano	666777888

Dopo aver fatto questo controllo ne eseguo un altro:

Vado a controllare se un numero si trova già nella tabella dei contatti oppure se è la prima volta che viene registrato quel numero.

Creo un'altra variabile booleana dove mi salvo il risultato del metodo *existNum*

```
54     public boolean existNum(String nTelefonico) throws SQLException{
55         Connection connection = DriverManager.getConnection(URL, USER, PASS);
56         PreparedStatement statement = connection.prepareStatement(sql: "SELECT * FROM contatti WHERE nTelefonico = ?");
57
58         statement.setString( parameterIndex: 1, nTelefonico);
59
60         ResultSet resultSet = statement.executeQuery();
61
62         if( resultSet.next()){
63             return true;
64         }else {
65             return false;
66         }
67     }
```

Se mi ritorna qualcosa, vuol dire che il numero inserito è già presente nella tabella contatti, quindi, vado a chiamare il metodo *inserisciLink* dove vado a salvare l'associazione tra email dell'utente e il numero del contatto senza salvare nuovamente il contatto

```
26     public void inserisciLink(String TuaEmail, String nTelefonico) throws SQLException{
27
28         Connection connection = DriverManager.getConnection(URL, USER, PASS);
29         PreparedStatement statement1 = connection.prepareStatement(sql: "INSERT INTO link VALUE (?,?)");
30         statement1.setString( parameterIndex: 1, TuaEmail);
31         statement1.setString( parameterIndex: 2, nTelefonico);
32
33         statement1.executeUpdate();
34     }
```

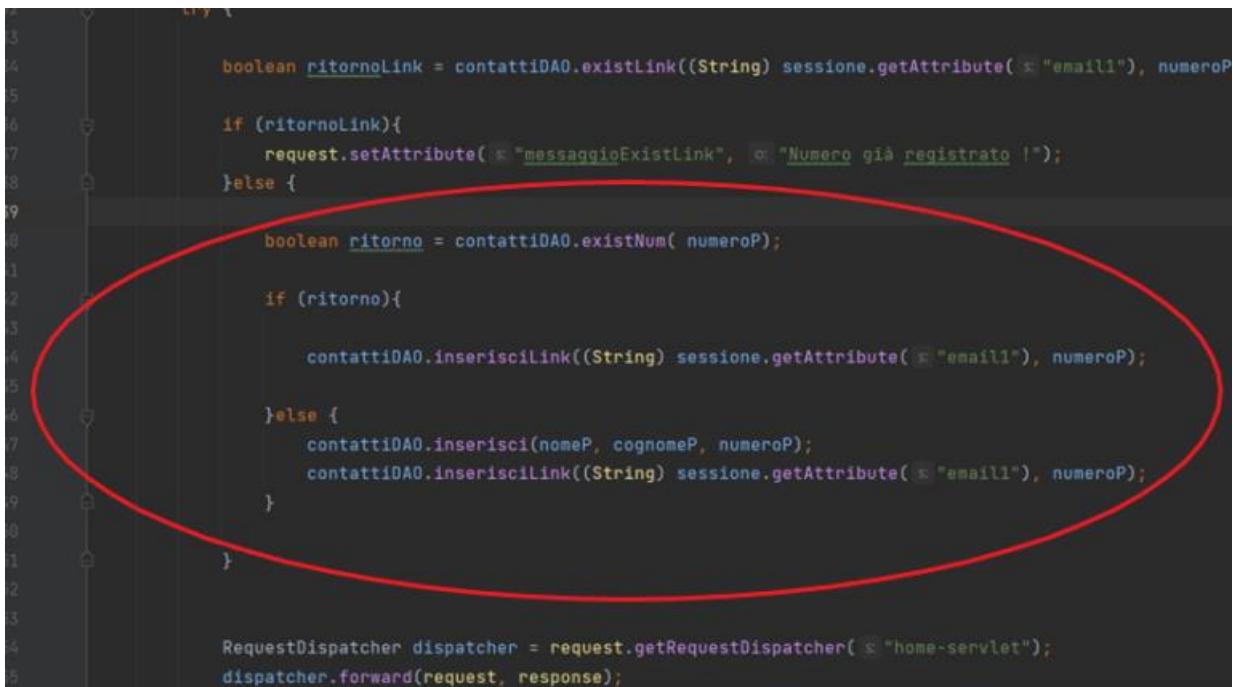
Altrimenti, se il numero è stato inserito per la prima volta, chiamo sia il metodo *inserisciLink* sia *inserisci* (mi permette di inserire il contatto nella tabella contatti)

```

13     public void inserisci(String nome, String cognome, String nTelefonico) throws SQLException {
14
15         Connection connection = DriverManager.getConnection(URL, USER, PASS);
16         PreparedStatement statement = connection.prepareStatement( sql: "INSERT INTO contatti VALUE (?, ?, ?); ");
17
18         statement.setString( parameterIndex: 1, nome);
19         statement.setString( parameterIndex: 2, cognome);
20         statement.setString( parameterIndex: 3, nTelefonico);
21
22         statement.executeUpdate();
23

```

E infine faccio un dispatcher sulla servlet *home* per aggiornare la lista, quindi visualizzare la lista con il nuovo contatto aggiunto.



```

13
14     boolean ritornoLink = contattiDAO.existLink((String) sessione.getAttribute( <: "email1"), numeroP);
15
16     if (ritornoLink){
17         request.setAttribute( <: "messaggioExistLink", <: "Numero già registrato !");
18     }else {
19
20         boolean ritorno = contattiDAO.existNum( numeroP);
21
22         if (ritorno){
23
24             contattiDAO.inserisciLink((String) sessione.getAttribute( <: "email1"), numeroP);
25
26         }else {
27             contattiDAO.inserisci(nomeP, cognomeP, numeroP);
28             contattiDAO.inserisciLink((String) sessione.getAttribute( <: "email1"), numeroP);
29         }
30     }
31
32
33     RequestDispatcher dispatcher = request.getRequestDispatcher( <: "home-servlet");
34     dispatcher.forward(request, response);
35

```

Ritorniamo alla Home:

Ciao, Stefano Gagliarde

Questa è la tua rubrica:

Inserisci persone
nella rubrica

Nome	Cognome	Numero Telefonico	Elimina
giorgio	gagliarde	333444555	
Luca	Giordano	666777888	

Ogni utente può eliminare un contatto dalla propria rubrica

```

12
13      <% for (int i=0; i<lista.size(); i++){ %>
14          <tr>
15              <td><%= lista.get(i).getNome() %></td>
16              <td><%= lista.get(i).getCognome()%></td>
17              <td><%= lista.get(i).getnTelefonico()%></td>
18              <td> <a href="CancellaContattoServlet?nTelefonico=<%=lista.get(i).getnTelefonico()%>">  </a></td>
19          </tr>
20      <%}%>
21  </table>
22

```

Una volta cliccato l'Icona del cestino

Si avvierà la servlet *CancellaContatto* dove gli passo come parametro il numero telefonico del contatto

```

12     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
13
14         HttpSession sessione = request.getSession( false );
15
16         ContattiDAO contattiDAO = new ContattiDAO();
17         try {
18
19             contattiDAO.deleteContatto((String) sessione.getAttribute("email1"),request.getParameter("nTelefonico"));
20
21             RequestDispatcher dispatcher = request.getRequestDispatcher("home-servlet");
22             dispatcher.forward(request, response);
23

```

Vado a richiamare la sessione e il metodo *deleteContatto* (mi permette di eliminare un contatto per quell'utente)

```

94     public void deleteContatto(String email, String nTelefonico) throws SQLException {
95
96         Connection connection = DriverManager.getConnection(URL, USER, PASS);
97         PreparedStatement statement = connection.prepareStatement(sql: "DELETE FROM link WHERE email = ? AND nTelefonico = ?");
98
99         statement.setString( parameterIndex: 1, email );
100        statement.setString( parameterIndex: 2, nTelefonico );
101
102        statement.executeUpdate();
103    }
104

```

E richiamo la servlet *Home* per ricaricare la rubrica.

Admin

Alla fine di tutto, c'è la sezione admin.

In questa sezione si potrà visualizzare tutti gli utenti registrati, tutti i contatti registrati e tutte le associazioni tra utenti e contatti

Come abbiamo visto prima nella servlet *Login* ho creato due variabili per entrare nella sezione admin

```
String emailAdmin = "admin@gmail.com";
String passwdAdmin = "admin123#";

try {

    if (email.equals(emailAdmin) && pass.equals(passwdAdmin)){

        RequestDispatcher dispatcher = request.getRequestDispatcher("homeAdmin-servlet");
        dispatcher.forward(request, response);
    }
}
```

Se l'email e la password combaciano con quelle dell'admin, vado a chiamare la servlet *Admin*

```
20
21     AdminDAO adminDAO = new AdminDAO();
22
23
24     try {
25
26         List<utentiAdmin> listaUtenti = adminDAO.visualizzaUtenti();
27         request.setAttribute("ListaUtenti", listaUtenti);
28
29         List<contattiAdmin> listaContatti = adminDAO.visualizzaContatti();
30         request.setAttribute("ListaContatti", listaContatti);
31
32         List<linkAdmin> listaLink = adminDAO.visulizzaLink();
33         request.setAttribute("ListaLink", listaLink);
34
35
36
37         RequestDispatcher dispatcher = request.getRequestDispatcher("WEB-INF/home-admin.jsp");
38         dispatcher.forward(request, response);
39 }
```

Vado a creare un oggetto di tipo AdminDAO dove vado a chiamare i 3 metodi per stampare tutte le liste dove ogni lista ha il proprio costruttore per salvare i dati.

Quindi vado a utilizzare la stessa logica che ho utilizzato per stampare i contatti nella home degli utenti

ListaUtenti

Metodo:

```
13     public List<utentiAdmin> visualizzaUtenti() throws SQLException {
14
15         Connection connection = DriverManager.getConnection(URL, USER, PASS);
16         PreparedStatement statement = connection.prepareStatement( sql: "SELECT * FROM utenti");
17
18         ResultSet resultSet = statement.executeQuery();
19
20         List<utentiAdmin> utentiList = new ArrayList<>();
21
22         while (resultSet.next()){
23
24             utentiAdmin utentiAdmin = new utentiAdmin(resultSet.getString( columnIndex: 1),
25                 resultSet.getString( columnIndex: 2), resultSet.getString( columnIndex: 3), resultSet.getString( columnIndex: 4));
26
27             utentiList.add(utentiAdmin);
28         }
29         return utentiList;
30     }
```

Classe:

```
1 package com.example.appweb;
2
3 public class utentiAdmin {
4
5     private String nome, cognome, email, password;
6
7     public utentiAdmin(String nome, String cognome, String email, String password) {
8
9         this.nome = nome;
10        this.cognome = cognome;
11        this.email = email;
12        this.password = password;
13    }
14
15    public String getNome(){return nome;}
16
17    public String getCognome(){return cognome;}
18
19    public String getEmail(){return email;}
20
21    public String getPassword(){return password;}
22
23    public String toString(){return nome +""+ cognome +""+ email +""+password;}
24
25 }
```

ListaContatti

Metodo:

```
32     public List<contattiAdmin> visualizzaContatti() throws SQLException {
33
34         Connection connection = DriverManager.getConnection(URL, USER, PASS);
35         PreparedStatement statement = connection.prepareStatement( sql: "SELECT * FROM contatti");
36
37         ResultSet resultSet = statement.executeQuery();
38
39         List<contattiAdmin> contattiList = new ArrayList<>();
40
41         while (resultSet.next()){
42
43             contattiAdmin contattiAdmin = new contattiAdmin(resultSet.getString( columnIndex: 1),
44                     resultSet.getString( columnIndex: 2), resultSet.getString( columnIndex: 3));
45
46             contattiList.add(contattiAdmin);
47         }
48         return contattiList;
49     }
```

Classe:

```
1  package com.example.appweb;
2
3  public class contattiAdmin {
4
5      private String nome, cognome, nTelefonico;
6
7      public contattiAdmin(String nome, String cognome, String nTelefonico){
8
9          this.nome = nome;
10         this.cognome = cognome;
11         this.nTelefonico = nTelefonico;
12     }
13
14
15     public String getNome(){return nome;}
16
17     public String getCognome(){return cognome;}
18
19     public String getNum(){return nTelefonico;}
20
21
22     public String toString(){return nome +""+ cognome+""+ nTelefonico;}
23 }
```

ListaLink

Metodo:

```
51  public List<linkAdmin> visulizzaLink() throws SQLException {  
52  
53      Connection connection = DriverManager.getConnection(URL, USER, PASS);  
54      PreparedStatement statement = connection.prepareStatement( sql: "SELECT * FROM link JOIN contatti ON link.nTelefonico = contatti.nTelefonico");  
55  
56      ResultSet resultSet = statement.executeQuery();  
57  
58      List<linkAdmin> linkList = new ArrayList<>();  
59  
60      while (resultSet.next()){  
61  
62          linkAdmin linkAdmin = new linkAdmin(resultSet.getString( columnIndex: 1),  
63                                         resultSet.getString( columnIndex: 3), resultSet.getString( columnIndex: 4), resultSet.getString( columnIndex: 5));  
64  
65          linkList.add(linkAdmin);  
66      }  
67      return linkList;  
68  }  
69
```

Classe:

```
1  package com.example.appweb;  
2  
3  public class linkAdmin {  
4  
5      private String email, nome, cognome, nTelefonico;  
6  
7      public linkAdmin(String email, String nome, String cognome, String nTelefonico) {  
8  
9          this.email = email;  
10         this.nome = nome;  
11         this.cognome = cognome;  
12         this.nTelefonico = nTelefonico;  
13     }  
14  
15     public String getEmail(){return email;}  
16  
17     public String getCognome(){return cognome;}  
18  
19     public String getNome(){return nome;}  
20  
21     public String getNum(){return nTelefonico;}  
22  
23  
24  ⚡  public String toString(){return nome +""+ cognome+""+ email +""+nTelefonico;}  
25  }  
26
```

Poi faccio visualizzare la homeAdmin.jsp

Dove vado a richiamare le liste e le vado a salvare in altre liste

Ho creato 3 pulsanti per andare nella tabella interessata utilizzando dei link interni

```
19 <body>
20
21
22 <%
23     List<utentiAdmin> listaUtenti = (ArrayList<utentiAdmin>) request.getAttribute("listaUtenti");
24     List<contattiAdmin> listaContatti = (ArrayList<contattiAdmin>) request.getAttribute("listaContatti") ;
25     List<linkAdmin> listaLink = (ArrayList<linkAdmin>) request.getAttribute("listaLink");
26 %>
27
28 <h1>Sezione Admin</h1>
29
30 <P> Vai a: </P>
31 <a href="#utenti">
32     <button class="scelta">Utenti Registrati</button>
33 </a>
34
35 <a href="#contatti">
36     <button class="scelta">Contatti Registrati</button>
37 </a>
38
39 <a href="#associati">
40     <button class="scelta">Contatti associati agli Utenti</button>
41 </a>
42
```

E infine vado a stampare le 3 liste nella homeAdmin.jsp

```
<table>
    <tr>
        <th>Nome</th>
        <th>Cognome</th>
        <th>Email</th>
        <th>Password</th>
    </tr>

    <% for (int i = 0; i < listaUtenti.size(); i++){%>

        <tr>
            <td> <%= listaUtenti.get(i).getNome() %> </td>
            <td> <%= listaUtenti.get(i).getCognome() %> </td>
            <td> <%= listaUtenti.get(i).getEmail() %> </td>
            <td> <%= listaUtenti.get(i).getPassword() %> </td>
        </tr>

    <% }%>
</table>
```

```
<table>

    <tr>
        <th>Nome</th>
        <th>Cognome</th>
        <th>Numero Telefonico</th>
    </tr>

    <% for (int j = 0; j < listaContatti.size(); j++)%>

        <tr>
            <td> <%= listaContatti.get(j).getNome()%> </td>
            <td> <%= listaContatti.get(j).getCognome()%> </td>
            <td> <%= listaContatti.get(j).getNum()%> </td>
        </tr>

    <%}%>

</table>
```

```
100 <table>
101
102     <tr>
103         <th>Email Utente</th>
104         <th>Nome Contatto</th>
105         <th>Cognome Contatto</th>
106         <th>Numero Telefonico Contatto</th>
107     </tr>
108
109     <% for (int k = 0; k < listaLink.size(); k++)%>
110         <tr>
111             <td> <%= listaLink.get(k).getEmail() %> </td>
112             <td> <%= listaLink.get(k).getNome() %> </td>
113             <td> <%= listaLink.get(k).getCognome() %> </td>
114             <td> <%= listaLink.get(k).getNum() %> </td>
115         </tr>
116
117         <%}%>
118     </table>
119
120
```

La HomeAdmin si presenta in questo modo:

Utenti Registrati

Nome	Cognome	Email	Password
Luca	Giordano	giordano@gmail.com	giordano123%
Giorgio	Gagliarde	giorgio@gmail.com	giorgio123@
Stefano	Gagliarde	stefano@gmail.com	stefano33@

Contatti Registrati

Nome	Cognome	Numero Telefonico
Giorgio	Gagliarde	777777
Stefano	Gagliarde	888888
Luca	Giordano	999999

Contatti associati agli Utenti

Email Utente	Nome Contatto	Cognome Contatto	Numero Telefonico Contatto
giordano@gmail.com	Giorgio	Gagliarde	777777
stefano@gmail.com	Giorgio	Gagliarde	777777
giordano@gmail.com	Stefano	Gagliarde	888888
giorgio@gmail.com	Stefano	Gagliarde	888888
giorgio@gmail.com	Luca	Giordano	999999
stefano@gmail.com	Luca	Giordano	999999