



INSTITUT NATIONAL DES POSTES ET TÉLÉCOMMUNICATIONS

RAPPORT DE PROJET

Dynamic Web project : Psychologue enline

Realisé par :
Driouch Meryam
Azzim Taha

Sous l'encadrement de :
Dr Mahmoud Rlhamlaoui

Année universitaire :2020-2021

Contents

1	Page Login	2
1.1	Page d'identification	2
1.2	Connexion avec une base de données MySQL : vérification des identifiants	3
1.2.1	Classe Session	3
1.2.2	Classe DB	4
1.2.3	Création de la base de données "userdb"	6
1.3	Login servlet	7
1.4	Teste de login	9
1.5	Configuration des differents acteurs : Psychologue, RH et Util- isateur	9
1.6	Partie CSS	13
2	Conception de la base de données	15
3	Interface "Utilisateur"	17
3.1	Classe Question	17
3.2	Chargement des questions depuis la base de données	18
3.3	Affichage des questions	20
3.4	Partie CSS	21

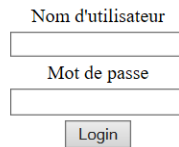
1 Page Login

1.1 Page d'identification

Créons premièrement un fichier Login.jsp dans lequel on aura la description de la page d'identification des différents utilisateurs.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="ISO-8859-1">
5 <title>Login</title>
6 </head>
7 <body>
8 <div align = "center">
9 <form action="login" method="post">
10 <table>
11 <tr>
12 <td align ="center">Nom d'utilisateur</td>
13 </tr>
14 <tr>
15 <td><input type="text" name="nom"></td>
16 </tr>
17 <tr>
18 <td align ="center">Mot de passe</td>
19 </tr>
20 <tr>
21 <td><input type="password" nom="mot de passe"></td>
22 </tr>
23 <tr>
24 <td align ="center"><input type="submit" value="Login"></td>
25 </tr>
26 </table>
27 </form>
28 </div>
29 </body>
30 </html>
```

On aura le resultat simple suivant



The image shows a simple login form. It consists of two text input fields stacked vertically. The first field is labeled 'Nom d'utilisateur' and the second field is labeled 'Mot de passe'. Below the second field is a button labeled 'Login'.

Figure 1: Login.jsp

1.2 Connexion avec une base de données MySQL : vérification des identifiants

1.2.1 Classe Session

On aura besoin d'une classe *Session* (package : loginssession) qui va récupérer pendant chaque identification le nom et le mot de passe entrés.

```
1
2 package loginssession;
3
4 public class session {
5     private String nom;
6     private String passe;
7     public String returnNom() {
8         return nom;
9     }
10    public String affecteNom(String nom) {
11        this.nom = nom;
12    }
13    public String returnPasse() {
14        return passe;
15    }
16    public String affectePasse(String passe) {
17        this.passe = passe;
18    }
19 }
```

1.2.2 Classe DB

La classe DB (Package : base_donnees) va permettre dans un premier lieux la connexion avec une base de données MySQL (userdb) qu'on va créer par la suite, puis vérifie si les identifiants (nom et mot de passe) entrés figurent dans cette base.

```
1 package base_donnees;
2 import java.sql.Connection;
3 import java.sql.DriverManager;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import loginsession.*;
8 public class DB {
9     private String dbUrl = "jdbc:mysql://localhost:3306/userdb?
10                             useJDBCCompliantTimezoneShift=true&
11                             useLegacyDatetimeCode=false&serverTimezone=UTC";
12     private String dbUname = "AzzimDriouich";
13     private String dbPassword = "0000";
14     private String dbDriver = "com.mysql.cj.jdbc.Driver";
15     public void loadDriver(String dbDriver)
16     {
17         try {
18             Class.forName(dbDriver);
19         } catch (ClassNotFoundException e) {
20             e.printStackTrace();
21         }
22     }
23     public Connection getConnection()
24     {
25         Connection con = null;
26         try {
27             con = DriverManager.getConnection(dbUrl, dbUname, dbPassword);
28         } catch (SQLException e) {
29             e.printStackTrace();
30         }
31     }
32 }
```

```

31         return con;
32     }
33     public boolean valider_donnees(Session session)
34     {
35         boolean status = false;
36
37         loadDriver(dbDriver);
38         Connection con = getConnection();
39         String sql = "SELECT *
40                       FROM login
41                       WHERE nom = ?
42                       AND mot_de_passe =?";
43         PreparedStatement ps;
44         try {
45             ps = con.prepareStatement(sql);
46             ps.setString(1, session.returnNom());
47             ps.setString(2, session.returnPasse());
48             ResultSet rs = ps.executeQuery();
49             status = rs.next();
50
51         } catch (SQLException e) {
52             e.printStackTrace();
53         }
54         return status;
55     }
56 }

```

1.2.3 Création de la base de données "userdb"

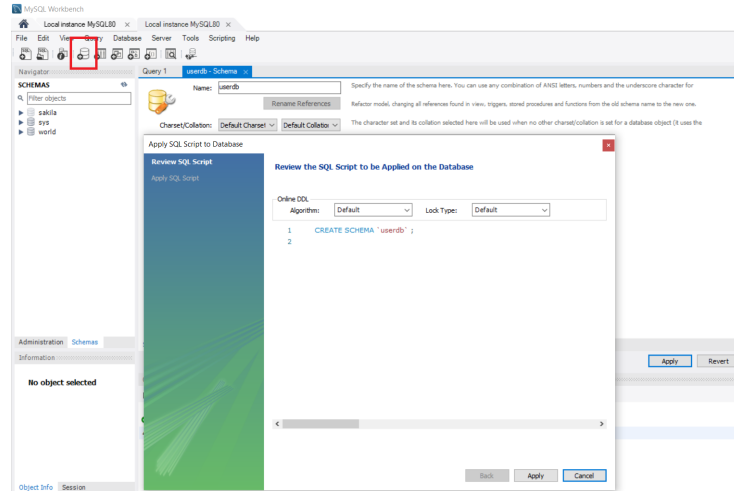


Figure 2: create schema

Après, on doit créer note tableau *login* avec les deux colonnes *nom* (Clé primaire et non null) et *mot_de_passe* (non null).

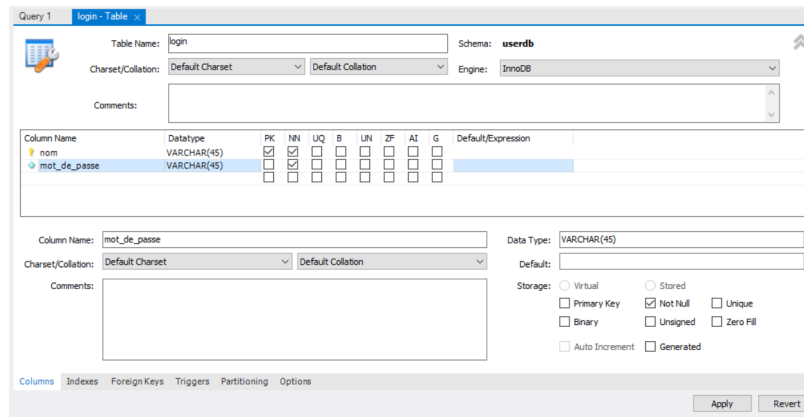


Figure 3: Création de la table

Afin de se connecter, on va insérer quelques utilisateurs à la table *login*.

Review the SQL Script to be Applied on the Database

```
1  INSERT INTO `userdb`.`login` (`nom`, `mot_de_passe`) VALUES ('Azzim', '111');
2  INSERT INTO `userdb`.`login` (`nom`, `mot_de_passe`) VALUES ('Driouich', '222');
3
```

Figure 4: Insertion des utilisateurs

1.3 Login servlet

Dans le package web, introduisant la première servlet qui va se servir de l'authentification et diriger l'utilisateur vers son compte si les identifiants sont corrects ou actualiser la page login sinon.

Pour cela, ajoutons un simple fichier Succes.jsp

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="ISO-8859-1">
5 <title>Succes</title>
6 </head>
7 <body>
8 Succes
9 </body>
10 </html>
```


Puis la servlet serait comme suit :

```
1 package web;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.annotation.WebServlet;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9 import base_donnees.*;
10 import loginsession.*;
11
12 @WebServlet("/login")
13 public class LoginServlet extends HttpServlet {
14
15     protected void doPost(HttpServletRequest request,
16                           HttpServletResponse response)
17     throws ServletException, IOException {
18         String nom = request.getParameter("nom");
19         String passe = request.getParameter("mot de passe");
20
21         Session session = new Session();
22         session.affecteNom(nom);
23         session.affectePasse(passe);
24
25         DB connexion_db = new DB();
26         if(connexion_db.valider_donnees(session)) {
27             response.sendRedirect("Succes.jsp");
28         }
29         else {
30             response.sendRedirect("login.jsp");
31         }
32     }
33 }
```

1.4 Teste de login

Exécutons le programme et essayons une authentification avec l'un des utilisateurs déclarés dans la base de données :

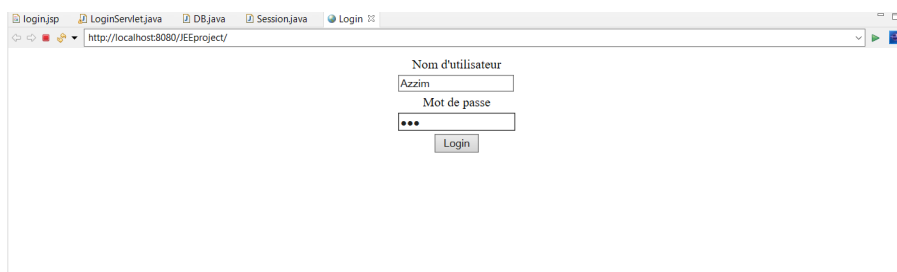


Figure 5: Exécution

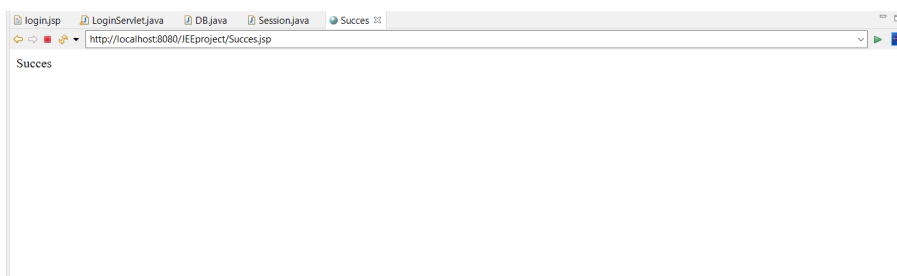
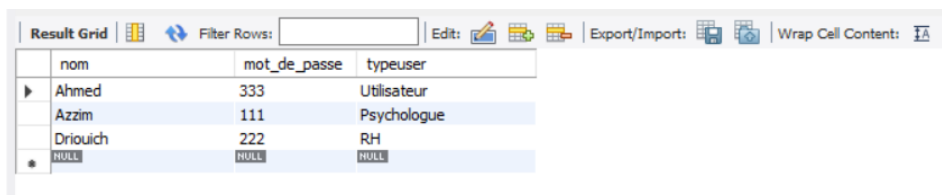


Figure 6: Authentification avec succès

1.5 Configuration des différents acteurs : Psychologue, RH et Utilisateur

Après l'authentification, l'inscrit doit être dirigé vers sa page personnelle. On distingue entre 3 type d'inscrits : Psychologue, RH et utilisateur. De ce fait, on doit modifier la table *login* en ajoutant la colonne *type*.



	nom	mot_de_passe	typeuser
▶	Ahmed	333	Utilisateur
	Azzim	111	Psychologue
	Driouch	222	RH
*	NULL	NULL	NULL

Figure 7: Table login

Ensuite on doit modifier la classe *Session* en ajoutant le String type et les méthodes affectType et returnType.

```
1 package loginsession;
2
3 public class Session {
4     private String nom;
5     private String passe;
6     private String typeUser;
7     public void affectType(String typeUser) {
8         this.typeUser = typeUser;
9     }
10    public String returnType() {
11        return typeUser;
12    }
13    public String returnNom() {
14        return nom;
15    }
16    public void affecteNom(String nom) {
17        this.nom = nom;
18    }
19    public String returnPasse() {
20        return passe;
21    }
22    public void affectePasse(String passe) {
23        this.passe = passe;
24    }
25 }
```

Pour la classe DB, pendant la validation des identifiants (valide_donnees), on récupère le type de l'inscrit et on affecte sa valeur à userType en faisant appel à affectType

```
1 public boolean valider_donnees(Session session)
2 {
3     boolean status = false;
4
5     loadDriver(dbDriver);
6     Connection con = getConnection();
7     String sql = "select * from login where nom =? and mot_de_passe =?";
8     PreparedStatement ps;
9     try {
10        ps = con.prepareStatement(sql);
11        ps.setString(1, session.returnNom());
12        ps.setString(2, session.returnPasse());
13        ResultSet rs = ps.executeQuery();
14        status = rs.next();
15        session.affectType(rs.getString("typeuser"));
16    } catch (SQLException e) {
17        e.printStackTrace();
18    }
19    return status;
20 }
```

Finalement la servlet doit diriger chaque type d'inscrit vers sa page personnelle (Psychologue.jsp, RH.jsp ou Utilisateur.jsp)

```
1  @WebServlet("/login")
2  public class LoginServlet extends HttpServlet {
3
4      protected void doPost(HttpServletRequest request,
5                             HttpServletResponse response)
6      throws ServletException, IOException {
7          String nom = request.getParameter("nom");
8          String passe = request.getParameter("mot de passe");
9
10         Session session = new Session();
11         session.affecteNom(nom);
12         session.affectePasse(passe);
13         DB connexion_db = new DB();
14         if(connexion_db.valider_donnees(session)) {
15             if(session.returnType().equals("Psychologue")) {
16                 response.sendRedirect("Psychologue.jsp");
17             }
18             else if(session.returnType().equals("Utilisateur")) {
19                 response.sendRedirect("Utilisateur.jsp");
20             }
21             else if(session.returnType().equals("RH")) {
22                 response.sendRedirect("RH.jsp");
23             }
24         }
25         else {
26             response.sendRedirect("login.jsp");
27         }
28     }
29
30 }
```

1.6 Partie CSS

Pour finir cette partie, introduisant un fichier css *loginCSS.css* pour les raisons esthétiques.

```
1  button {
2      background-color: white;
3      color: black;
4      padding: 14px 20px;
5      border-radius: 10px;
6      margin: 8px 0;
7      border: none;
8      cursor: pointer;
9      width: 100%;
10 }
11 input[type=text], input[type=password] {
12     width: 100%;
13     border-radius: 10px;
14     padding: 12px 20px;
15     margin: 8px 0;
16     display: inline-block;
17     border: 1px solid #ccc;
18     box-sizing: border-box;
19 }
20 button:hover {
21     opacity: 0.8;
22     background-color: black;
23     color: white;
24 }
25 body {
26     background-image: url("1.gif");
27     background-attachment: fixed;
28     background-repeat: no-repeat;
29     background-position: center;
30     background-size: cover;
31     background-color: white;
32 }
```

```

33
34 .login {
35     overflow: hidden;
36     opacity: 0.8;
37     background-color: #8e8e8e;
38     padding: 20px 30px 30px 30px;
39     border-radius: 10px;
40     top: 100px;
41     width: 400px;
42     box-shadow: 5px 10px 10px rgba(green, 0.2);
43 }

```

le fichier *login.jsp* sera aussi changé

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="ISO-8859-1">
5 <title>Login</title>
6 </head>
7 <link href="loginCSS.css" rel="stylesheet" type="text/css"><body>
8 <div align = "center">
9 <form class = login action="login" method="post">
10 <table>
11 <tr>
12 <td><input type="text" name="nom" placeholder="Nom d'utilisateur"></td>
13 </tr>
14 <tr>
15 <td><input type="password" name="mot de passe" placeholder="Password"></td>
16 </tr>
17 <tr>
18 <td align ="center"><button type="submit" value="Login">login</button></td>
19 </tr>
20 </table>
21 </form>
22 </div>
23 </body>
24 </html>

```

Résultat final :

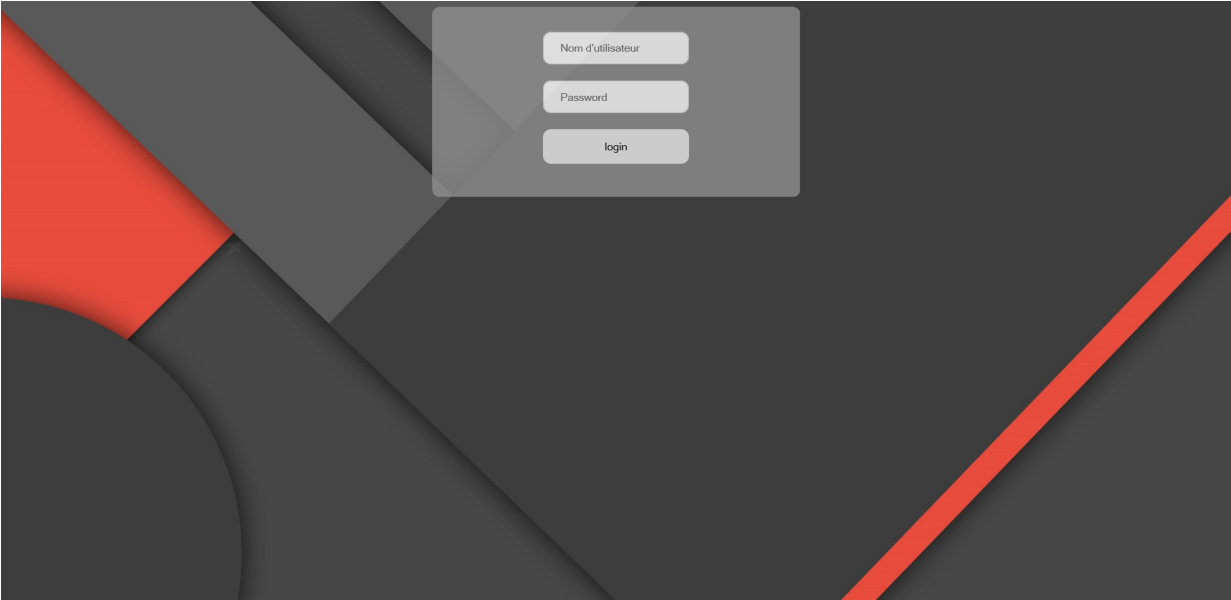


Figure 8: Page login

2 Conception de la base de données

De plus du tableau *login* on aura besoin d'autres pour stocker les questions posés par les psychologues et les réponses récupérés de la part des utilisateurs.

On propose l'ajout de deux tableaux :

- **Formulaires** : renferme l'id du formulaire (Clé primaire), le nom du psychologue (Créateur du formulaire), le nom d'utilisateur (Destinataire) et l'état du formulaire (Approuvé ou non par le RH).
- **Questions** : contient la question, son l'id (Clé primaire), l'id du formulaire où se trouve et les réponses fournies par les utilisateurs.

Formulaires			
id_formulaire	utilisateur	psychologue	etat
.	.	.	.
.	.	.	.
.	.	.	.

Questions			
id_question	question	id_formulaire	reponse
.	.	.	.
.	.	.	.
.	.	.	.

La création des deux tableaux consiste à lancer les deux requêtes :

```

1  CREATE TABLE `userdb`.`formulaires` (
2    `id_formulaire` INT NOT NULL,
3    `nom` VARCHAR(45) NOT NULL,
4    `psychologue` VARCHAR(45) NOT NULL,
5    `etat` TINYINT(1) NOT NULL,
6    PRIMARY KEY (`id_formulaire`));
7

```

Figure 9: table formulaires

```

1  CREATE TABLE `userdb`.`questions` (
2    `id_question` INT NOT NULL,
3    `id_formulaire` INT NOT NULL,
4    `question` MEDIUMTEXT NOT NULL,
5    `reponse` TINYINT(1) NULL,
6    PRIMARY KEY (`id_question`));
7

```

Figure 10: table questions

NOTE : Pour éviter la confusion entre les id des formulaires et ceux des questions, les id des formulaires débuteraient toujours par un "10" (exemple : 101, 102 ...) alors que ceux des questions débuteraient avec un "20" (exemple : 201 202 ...)

Insérant par suite quelques données dans les tables pour qu'elles nous aident pendant la construction des interfaces des utilisateurs (formulaire 101 avec 2 questions 201 et 202 et formulaire 102 avec une question 203)

```

1  INSERT INTO `userdb`.`formulaires` (`id_formulaire`, `nom`, `psychologue`, `etat`) VALUES ('101', 'Ahmed', 'Azzim', '1');
2  INSERT INTO `userdb`.`formulaires` (`id_formulaire`, `nom`, `psychologue`, `etat`) VALUES ('102', 'Karima', 'Azzim', '0');
3

```

Figure 11: table formulaires

```

1  INSERT INTO `userdb`.`questions` (`id_question`, `id_formulaire`, `question`) VALUES ('201', '101', 'Ouvrais-tu un enveloppe contenant la date de ta propre mort?');
2  INSERT INTO `userdb`.`questions` (`id_question`, `id_formulaire`, `question`) VALUES ('202', '101', 'Pourrais-tu etre ami avec toi meme?');
3  INSERT INTO `userdb`.`questions` (`id_question`, `id_formulaire`, `question`) VALUES ('203', '102', 'Avez-vous deja ete surpris par la qualite du travail que quelqu'un vous a presente?');
4

```

Figure 12: table questions

Maintenant que tout est prêt, construisons les interfaces des utilisateurs.

3 Interface "Utilisateur"

L'utilisateur doit être capable de voir les questions du formulaire fournit par le psychologue et approuvé par le RH. Il doit aussi avoir le droit de répondre à chaque question avec oui ou non. Finalement, il confirme ses réponses pour qu'elles soient envoyés au psychologue.

3.1 Classe Question

La première étape consiste à créer une classe *Question* (Similaire à *Session*) dont les attributs sont les colonnes du tableau Questions.

```

1  package loginsession;
2
3  public class Question {
4      public int id_question;
5      public int id_formulaire;
6      public String questiontext;
7      public boolean reponse;
8
9      public void affectIdQuestion(int id_question) {

```

```

10         this.id_question =id_question;
11     }
12     public int returnIdQuestion() {
13         return id_question;
14     }
15     public void affectIdFormulaire(int id_formulaire) {
16         this.id_formulaire =id_formulaire;
17     }
18     public int returnIdformulaire() {
19         return id_formulaire;
20     }
21     public String returnQuestion() {
22         return questiontext;
23     }
24     public void affectQuestion(String questiontext) {
25         this.questiontext = questiontext;
26     }
27     public boolean returnReponse() {
28         return reponse;
29     }
30     public void affectReponse(boolean reponse) {
31         this.reponse =reponse;
32     }
33
34 }

```

3.2 Chargement des questions depuis la base de données

Ensuite, on déclare la fonction `question_utilisateur()` qui prend en argument une *Session* et une liste des objets de la classe *Question* puis ajoute dans cette liste les questions du formulaire envoyé à cet utilisateur et qui sont approuvés par le RH.

Le tableau désiré est résultat de la requête :

```

SELECT *
FROM userdb.questions q
INNER JOIN userdb.formulaires f

```

ON f.id_formulaire = q.id_formulaire
WHERE f.nom=?
AND f.etat=1

Le champ f.nom serait remplis par session.nom.

```

1  public void question_utilisateur(Session session, List<Question> userquestion)
2      {
3          boolean status;
4
5          loadDriver(dbDriver);
6          Connection con = getConnection();
7          String sql = "SELECT * FROM userdb.questions q
8                        INNER JOIN userdb.formulaires f
9                        ON f.id_formulaire=q.id_formulaire
10                       WHERE f.nom=? AND f.etat=1";
11          PreparedStatement ps;
12          try {
13              ps = con.prepareStatement(sql);
14              ps.setString(1, session.returnNom());
15              ResultSet rs = ps.executeQuery();
16              status = rs.next();
17              while(status) {
18                  Question question = new Question();
19                  question.affectIdQuestion(rs.getInt("id_question"));
20                  question.affectIdFormulaire(rs.getInt("id_formulaire"));
21                  question.affectReponse(rs.getBoolean("reponse"));
22                  question.affectQuestion(rs.getString("question"));
23
24                  userquestion.add(question);
25                  status = rs.next();
26              }
27          } catch (SQLException e) {
28              e.printStackTrace();
29          }
30      }
  
```

3.3 Affichage des questions

Finalement, dans la partie *Utilisateur* de la servlet, on introduit la liste *userquestion* des objets de la classe *Question* qui serait remplies par la fonction *question_utilisateur()* et envoyée vers *Utilisateur.jsp*.

```
1 else if(session.returnType().equals("Utilisateur")) {
2     List<Question> userquestion = new ArrayList<Question>();
3     connexion_db.question_utilisateur(session, userquestion);
4     request.setAttribute("userquestion", userquestion);
5     request.setAttribute("utilisateur", session.returnNom());
6     RequestDispatcher rst = request.getRequestDispatcher("Utilisateur.jsp");
7     rst.forward(request, response);
8 }
```

Finalement, la liste des questions peut être affichée à l'aide de *foreach*.

```
1 <%@page contentType="text/html" pageEncoding="UTF-8"%>
2 <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="UTF-8">
7 <title>${utilisateur}</title>
8
9 </head>
10 <link href="UtilisateurCSS.css" rel="stylesheet" type="text/css">
11 <body>
12 <form class="envoie" action="envoie" method="post">
13 <br>
14 <br>
15 <h1 align="center">Veuillez repondre aux questions suivantes</h1>
16 <br>
17 <br>
18 <div class="container">
19 <table align="center">
20 <thead>
21 <tr>
22 <th>Question</th>
23 <th>Reponse</th>
24 </tr>
```

```

25     </thead>
26     <tbody>
27         <c:forEach items="${userquestion}" var="question">
28             <tr>
29                 <td><c:out value="${question.returnQuestion()}" /></td>
30                 <td>
31                     <select name="reponse" class="dropbtn">
32                         <option value="false">Oui</option>
33                         <option value="true">Non</option>
34                     </select></td>
35             </tr>
36         </c:forEach>
37     </tbody>
38 </thead>
39 <tr>
40     <td></td>
41     <th align=center><button type="submit" value="Login">Envoyer</button></th>
42 </tr>
43 </thead>
44 </table>
45 </div>
46 </form>
47 </body>
48 </html>

```

3.4 Partie CSS

Le fichier CSS décrivant la page utilisateur est donné par :

```

1 @import url(https://fonts.googleapis.com/css?family=Open+Sans:400,600);
2 h1{
3     font-size: 20px;
4     font-family: 'Courier New';
5     color : white;
6 }
7 button {
8     background-color: white;
9     color: black;
10    display:inline-block;

```

```

11     padding: 10px 30px;
12     border-radius: 10px;
13     font-size: 20px;
14     margin: 30px 0;
15     border: none;
16     cursor: pointer;
17     display: inline-block;
18     transition: all 0.25s;
19 }
20
21 button:hover {
22     opacity: 0.8;
23     background-color: #E2B842;
24     color: white;
25 }
26
27 *, *:before, *:after {
28     margin: 0;
29     padding: 0;
30     box-sizing: border-box;
31 }
32 .dropbtn {
33     background-color: #012B39;
34     color: #ffffff;
35     padding: 10px 30px;
36     font-size: 15px;
37     border: none;
38     cursor: pointer;
39 }
40 body {
41     background: #105469;
42     font-family: 'Open Sans', sans-serif;
43 }
44 table {
45     width: 70%;
46     background: #012B39;

```

```

47     border-radius: 0.25em;
48     border-collapse: collapse;
49     margin: 1em;
50 }
51 th {
52     border-bottom: 1px solid #364043;
53     color: #E2B842;
54     font-size: 20px;
55     font-weight: 600;
56     padding: 0.5em 1em;
57     text-align: Left;
58 }
59 td {
60     color: #fff;
61     font-weight: 400;
62     padding: 0.65em 1em;
63 }
64 .disabled td {
65     color: #4F5F64;
66 }
67 tbody tr {
68     transition: background 0.25s ease;
69 }
70 tbody tr:hover {
71     background: #014055;
72 }

```

Résultat donné pour utilisateur "Ahmed" :

Veuillez répondre aux questions suivantes

Question	Réponse
Ouvrirais-tu un enveloppe contenant la date de ta propre mort ?	Oui <input checked="" type="checkbox"/>
Pourrais-tu être ami avec toi-même ?	Oui <input checked="" type="checkbox"/>

Envoyer

Figure 13: Interface utilisateur