

Documentation Projet

Date de création : 31.03.2023  
Version 1 du 31.03.2023

Damian Piller

|  |  |  |
| --- | --- | --- |
|  |  |  |

Table des matières

[1 Introduction 1](#_Toc138421566)

[2 Analyse 3](#_Toc138421567)

[2.1 Use Case 3](#_Toc138421568)

[2.2 System Sequence 8](#_Toc138421569)

[3 Conception 9](#_Toc138421570)

[3.1 Navigation Client vers Serveur 9](#_Toc138421571)

[3.2 Diagramme de classe 11](#_Toc138421572)

[4 Réalisation 15](#_Toc138421573)

[4.1 ClientAP1 15](#_Toc138421574)

[4.1.1 Index.html 15](#_Toc138421575)

[4.1.2 Home.html 16](#_Toc138421576)

[4.2 ClientAP2 18](#_Toc138421577)

[4.3 Gateway 18](#_Toc138421578)

[4.3.1 Gateway.java 18](#_Toc138421579)

[4.3.2 TeamManager.java 22](#_Toc138421580)

[4.3.3 UserManager.java 22](#_Toc138421581)

[4.4 ServiceRest1 23](#_Toc138421582)

[4.4.1 ApplicationConfig.java 23](#_Toc138421583)

[4.4.2 TeamManager.java 24](#_Toc138421584)

[4.4.3 Wrk.java 25](#_Toc138421585)

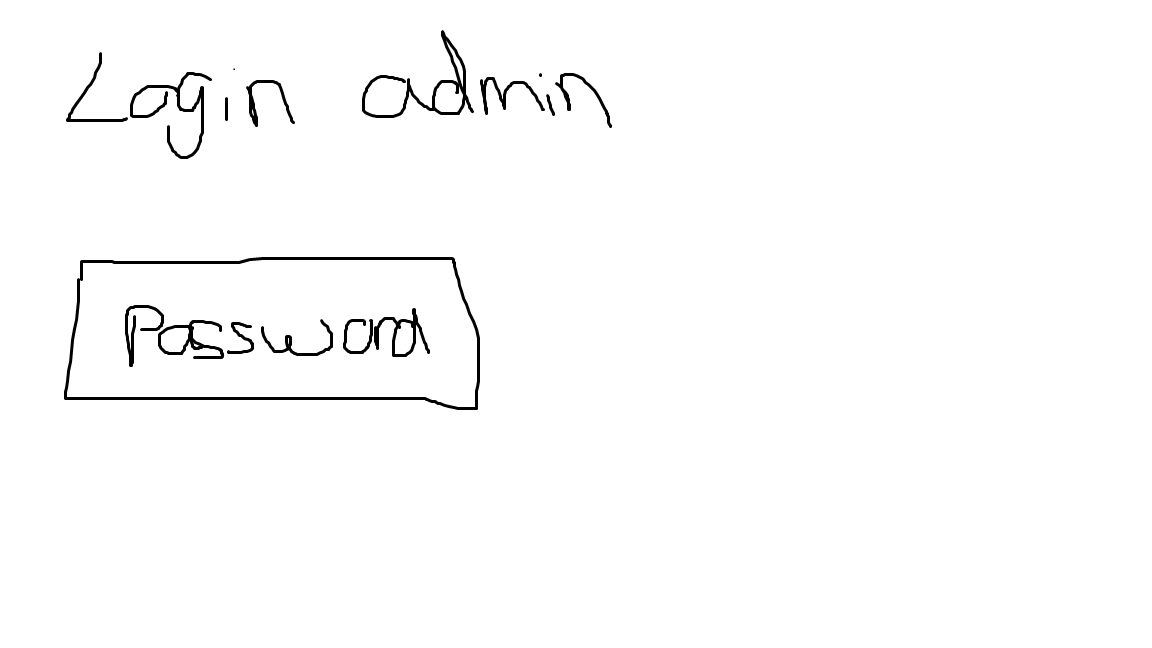
[4.5 ServiceRest2 28](#_Toc138421586)

[5 Conclusion 28](#_Toc138421587)

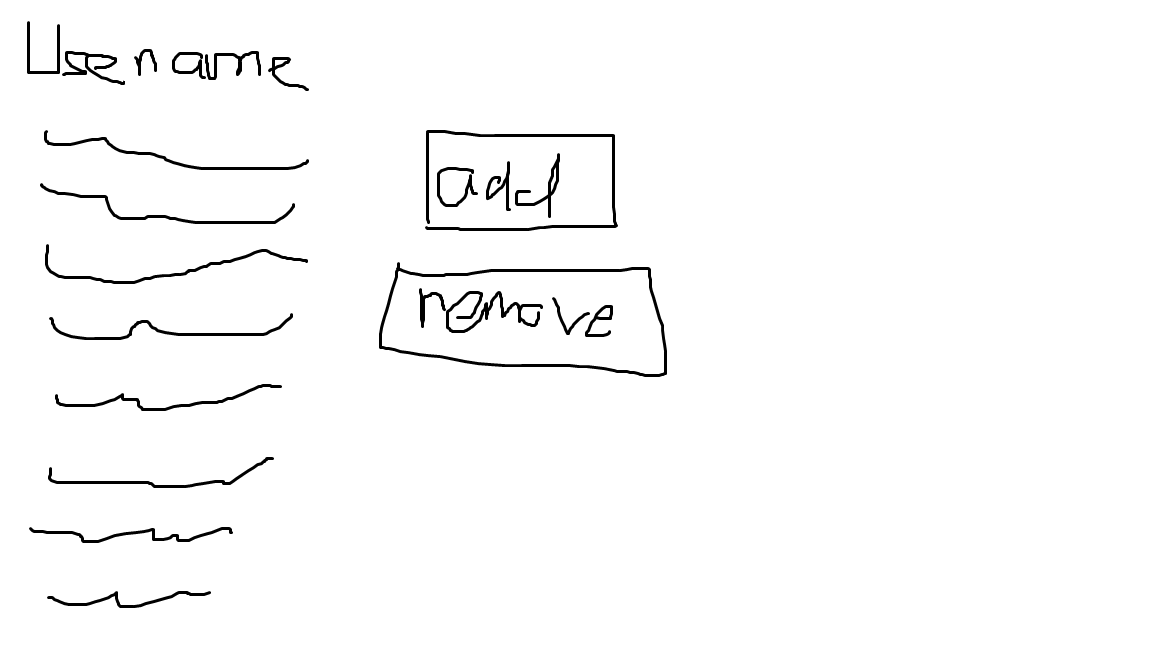
# Introduction

Dans notre projet, nous allons faire deux pages différentes. Sur la page de visite, on peut voir les équipes de football, dans quelle ligue elles jouent, un nom d'utilisateur et un mot de passe pour s'inscrire. Une fois connecté, on peut ajouter une équipe, supprimer une équipe ou éditer le nom ou l'abréviation de l'équipe. Sur la page d'administration, on ne peut se connecter qu'avec un seul compte, il n'y a donc qu'un seul champ pour le mot de passe. Une fois ce dernier saisi, on est redirigé vers une page où sont affichés les différents comptes d'utilisateurs, sur laquelle on peut ensuite supprimer les comptes ou en créer un nouveau. Ci-dessous, vous pouvez voir quelques dessins de la manière dont nous avons imaginé le tout :

La page de Login dans le site admin



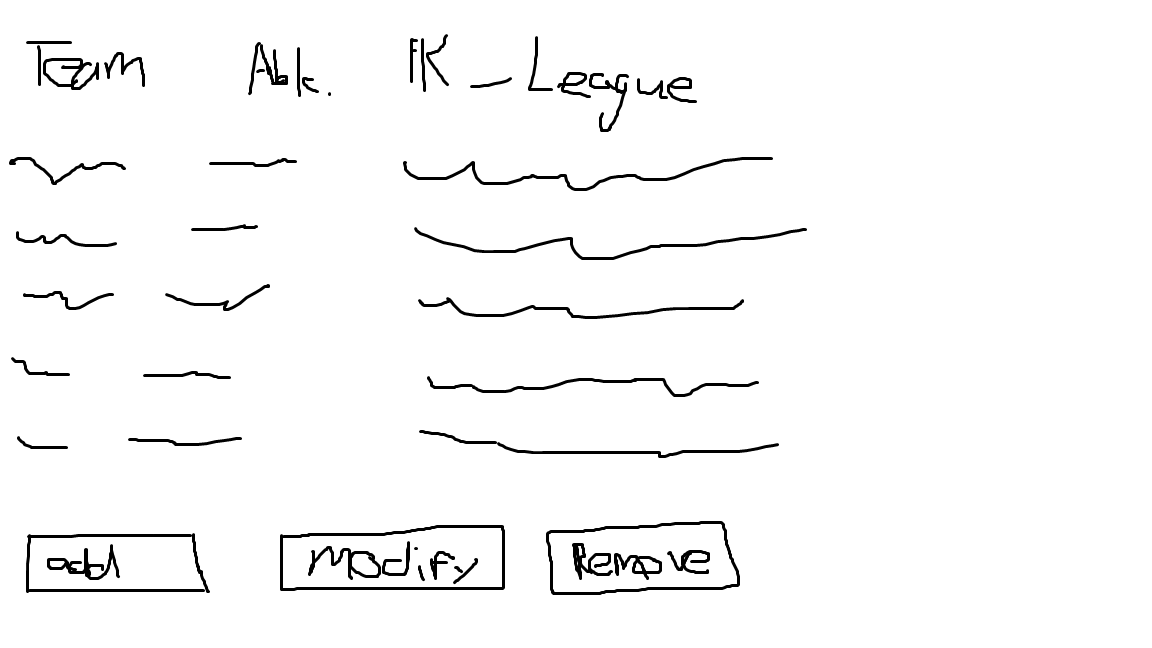
Si on est logué :



La page visiteur dans le site user :



Si on est logué :



# Analyse

## Use Case

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

Ein Bild, das Diagramm, Plan enthält.

Automatisch generierte Beschreibung

Ein Bild, das Diagramm, Plan enthält.

Automatisch generierte Beschreibung

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

Ein Bild, das Diagramm, Plan enthält.

Automatisch generierte Beschreibung

Ein Bild, das Diagramm, Plan enthält.

Automatisch generierte Beschreibung

## System Séquence

Ajouter une équipe dans la page utilisateur

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

Supprimer un utilisateur dans la page admin

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

# Conception

## Navigation Client vers Serveur

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

Ein Bild, das Diagramm, Plan enthält.

Automatisch generierte Beschreibung

## Diagramme de classe

Ein Bild, das Text, Screenshot, parallel, Schrift enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, parallel, Rechteck enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, parallel, Schrift enthält.

Automatisch generierte Beschreibung

# Réalisation

## ClientAP1

### Index.html

<html>

<head>

<title>Login Page</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<script src="/clientAp1/index.js"></script>

</head>

<body>

<div id="indexView">

<label for="user">Votre nom d'utilisateur :</label>

<input type="text" name="username" id="user" size="30" maxlength="30" />

<label for="pass">Votre mot de passe :</label>

<input type="password" name="password" id="pass" size="30" maxlength="30" />

<button id="loginBtn">Login</button>

<div id="response"></div>

<script>

$().ready(function () {

$.ajax({

url: 'https://pillerd.emf-informatique.ch/java\_APIGateway1/Gateway',

type: "POST",

dataType: "text",

data: 'action=teams',

success: function (data) {

$("#response").html(data);

},

error: function (jqXHR, textStatus, errorThrown) {

console.log(textStatus, errorThrown);

}

});

$("#loginBtn").on("click", function () {

console.log("loginBtn clicked");

var username = $("#user").val();

var password = $("#pass").val();

var action = "login"; // Set the value of the action parameter

console.log("Action parameter: " + action); // Log the value of the action parameter

$.ajax({

url: "https://pillerd.emf-informatique.ch/java\_APIGateway1/Gateway",

method: "POST",

dataType: "text",

data: {

action: action, // Pass the action parameter to the AJAX request

username: username,

password: password

},

success: function (response) {

console.log(response);

if (response) {

window.location.href = "home.html";

} else {

alert("Invalid username or password");

}

},

error: function (jqXHR, textStatus, errorThrown) {

console.log(textStatus, errorThrown);

}

});

});

});

</script>

<a href="http://WSTEMFB32-16:8085/clientAp2/index.html" >

<button>Login Admin</button>

</a>

</div>

</body>

</html>

### Home.html

<html>

<head>

<title>Home</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<script src="/clientAp1/index.js"></script>

</head>

<body>

<div id="homeView">

<div id="response"></div>

<script>

$(document).ready(function () {

$.ajax({

url: 'https://pillerd.emf-informatique.ch/java\_APIGateway1/Gateway',

type: "POST",

dataType: "text",

data: {action: "teams"},

success: function (data) {

$("#response").html(data);

},

error: function (jqXHR, textStatus, errorThrown) {

console.log(textStatus, errorThrown);

}

});

});

</script>

<button id="edit-btn">Edit</button>

<div id="popupEditForm" style="display:none;">

<form id="editForm">

<label for="name">Name:</label>

<input type="text" id="nameedit" name="nameedit"><br><br>

<label for="code">Code:</label>

<input type="text" id="codeedit" name="codeedit"><br><br>

<button id="editTeam">Edit</button>

</form>

</div>

<button id="delete">Delete</button>

<button id="addBtn">Add</button>

<div id="popupForm" style="display:none;">

<form id="addForm">

<label for="name">Name:</label>

<input type="text" id="name" name="name"><br><br>

<label for="code">Code:</label>

<input type="text" id="code" name="code"><br><br>

<label for="location">League:</label>

<select id="location" name="location">

<option value="1">Premiere League-GBR</option>

<option value="2">Ligue 1-FRA</option>

<option value="3">Bundesliga-GER</option>

<option value="4">Serie A-ITA</option>

<option value="5">LaLiga-ESP</option>

<option value="6">Super League-CH</option>

</select><br><br>

<button id="addTeam">Add</button>

</form>

</div>

<script>

function refreshTeamList() {

$.ajax({

url: 'https://pillerd.emf-informatique.ch/java\_APIGateway1/Gateway',

type: "POST",

dataType: "text",

data: {action: "teams"},

success: function (data) {

$("#response").html(data);

},

error: function (jqXHR, textStatus, errorThrown) {

console.log(textStatus, errorThrown);

}

});

}

$(document).ready(function () {

$("#addBtn").on("click", function () {

$("#popupForm").show();

});

$("#addForm").on("submit", function (e) {

e.preventDefault();

var name = $("#name").val();

var code = $("#code").val();

var location = $("#location").val();

$.ajax({

url: "https://pillerd.emf-informatique.ch/java\_APIGateway1/Gateway",

method: "POST",

data: {action: "addTeam", name: name, code: code, location: location},

success: function (response) {

alert("Team added");

$("#popupForm").hide();

refreshTeamList();

},

error: function (jqXHR, textStatus, errorThrown) {

console.log(textStatus, errorThrown);

}

});

});

// Load initial team list

refreshTeamList();

});

$(document).ready(function () {

$("#delete").on("click", function (e) {

e.preventDefault();

var teamId = $("#teams option:selected").val();

$.ajax({

url: "https://pillerd.emf-informatique.ch/java\_APIGateway1/Gateway",

method: "POST",

data: {action: "delTeam", teamId: teamId},

success: function (response) {

alert("Team deleted");

refreshTeamList();

},

error: function (jqXHR, textStatus, errorThrown) {

console.log(textStatus, errorThrown);

}

});

});

});

$(document).ready(function () {

$("#edit-btn").on("click", function () {

$("#popupEditForm").show();

});

$("#editForm").on("submit", function (e) {

e.preventDefault();

var name = $("#nameedit").val();

var code = $("#codeedit").val();

var teamId = $("#teams option:selected").val();

$.ajax({

url: "https://pillerd.emf-informatique.ch/java\_APIGateway1/Gateway",

method: "POST",

data: {action: "editTeam", name: name, code: code, teamId: teamId},

success: function (response) {

alert("Team edited");

$("#popupEditForm").hide();

refreshTeamList();

},

error: function (jqXHR, textStatus, errorThrown) {

console.log(textStatus, errorThrown);

}

});

});

});

</script>

</div>

</body>

</html>

## ClientAP2

### Index.html

<html>

<head>

<title>adminlogin</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<form method="post" action="">

<label>Mettez le mot de passe du compte admin!!!</label>

<input type="password" name="adminPassword" />

<input type="submit" name="btnLoginAdmin" id="btnLoginAdmin" />

</form>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<script>

$(document).ready(function () {

$('#btnLoginAdmin').click(function (e) {

e.preventDefault();

var adminPassword = $('input[name="adminPassword"]').val();

if (adminPassword === "admP") {

window.location.href = 'adminView.html';

} else {

console.log("Incorrect password. Please try again.");

}

});

});

</script>

</body>

</html>

### Adminview.html

<head>

<title>admin View</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<form method="post" >

<label>All the users:</label>

<div id="listUsers">

</div>

<div>

<button id="createButton" type="button">Create</button>

<div id="popupAddForm" style="display:none;">

<form id="addForm">

<label for="usernameadd">Username:</label>

<input type="text" id="usernameadd" name="usernameadd"><br><br>

<label for="password">Password:</label>

<input type="text" id="passwordadd" name="passwordadd"><br><br>

<button id="addUser">Add</button>

</form>

</div>

<button id="editButton" type="button">Edit</button>

<div id="popupEditForm" style="display:none;">

<form id="editForm">

<label for="username">Username:</label>

<input type="text" id="usernameedit" name="usernameedit"><br><br>

<label for="password">Password:</label>

<input type="text" id="passwordedit" name="passwordedit"><br><br>

<button id="editUser">Edit</button>

</form>

</div>

<button id="deleteButton">Delete</button>

</div>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<script>

$(document).ready(function () {

$.ajax({

url: 'https://pillerd.emf-informatique.ch/java\_APIGateway1/Gateway',

type: 'POST',

dataType: 'text',

data: "action=adminLogin",

success: function (data) {

$("#listUsers").html(data);

},

error: function (jqXHR, textStatus, errorThrown) {

console.log("list users error:" + errorThrown);

}

});

});

// Create Button

$(document).ready(function () {

$('#createButton').on("click", function () {

if ($('#popupAddForm').is(":visible")) {

$('#popupAddForm').hide();

}

$('#popupAddForm').show();

});

$('#addUser').on("click", function (e) {

e.preventDefault();

console.log("in adminview");

var nameuser = $("#usernameadd").val();

var namepass = $("#passwordadd").val();

console.log(nameuser);

console.log(namepass);

$.ajax({

url: 'https://pillerd.emf-informatique.ch/java\_APIGateway1/Gateway',

type: 'POST',

dataType: 'text',

data: {

action: 'addUser',

username: nameuser,

password: namepass

},

success: function (data) {

alert("Successfully added");

refreshUserList();

$('#popupAddForm').hide();

},

error: function (jqXHR, textStatus, errorThrown) {

alert("ERROR in addUser: " + errorThrown);

}

});

});

});

// Edit Button

$(document).ready(function () {

$('#editButton').click(function () {

if ($('#popupEditForm').show() === true) {

$('#popupEditForm').hide();

}

$('#popupEditForm').show();

});

$('#editUser').click(function () {

$.ajax({

url: 'https://pillerd.emf-informatique.ch/java\_APIGateway1/Gateway',

type: 'POST',

dataType: 'text',

data: {

action: 'modUser',

userID: $('#users option:selected').val(),

username: $('#usernameedit').val(),

password: $('#passwordedit').val()

},

success: function (data) {

alert("successfully edited");

refreshUserList();

$('#popupEditForm').hide();

},

error: function (jqXHR, textStatus, errorThrown) {

alert(errorThrown);

}

});

});

});

// Delete Button

$(document).ready(function () {

$('#deleteButton').click(function () {

$.ajax({

url: 'https://pillerd.emf-informatique.ch/java\_APIGateway1/Gateway',

type: 'POST',

dataType: 'text',

data: {

action: 'delUser',

userID: $('#users option:selected').val()

},

success: function (data) {

alert("User successfully deleted");

refreshUserList();

},

error: function (jqXHR, textStatus, errorThrown) {

// Handle error response

}

});

});

}

);

function refreshUserList() {

$.ajax({

url: 'https://pillerd.emf-informatique.ch/java\_APIGateway1/Gateway',

type: 'POST',

dataType: 'text',

data: {

action: "adminLogin"

},

success: function (data) {

$("#listUsers").html(data);

},

error: function (jqXHR, textStatus, errorThrown) {

console.log("list users error:" + errorThrown);

}

});

}

</script>

</html>

## Gateway

### Gateway.java

public class Gateway extends HttpServlet {

String action;

String adminP;

String addUsername;

String addPassword;

String modUsername;

String modPassword;

UserManager userManager;

TeamManager teamManager;

public Gateway() {

this.teamManager = new TeamManager();

this.userManager = new UserManager();

}

/\*\*

\* Processes requests for both HTTP <code>GET</code> and <code>POST</code>

\* methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/plain");

System.out.println("dans pprocess-----");

response.setContentType("text/plain");

System.out.println(response.getHeaderNames());

HttpSession session = request.getSession();

session.setMaxInactiveInterval(20); //temps en seconde

adminP = request.getParameter("adminPassword");

action = request.getParameter("action");

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

Gson gson = new Gson();

response.addHeader("Access-Control-Allow-Origin", "http://127.0.0.1:5500");

// processRequest(request, response);

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

action = request.getParameter("action");

response.setContentType("text/html");

Gson gson = new Gson();

if (action.equals("addTeam")) {

PrintWriter out = response.getWriter();

String name = request.getParameter("name");

String code = request.getParameter("code");

String leagueGanz = request.getParameter("location");

League league = new League();

if (leagueGanz != null) {

int leagueNbr = Integer.parseInt(leagueGanz);

switch (leagueNbr) {

case 1:

league.setPKLeague(1);

league.setName("Premier League");

league.setCountryCode("GBR");

break;

case 2:

league.setPKLeague(2);

league.setName("Ligue 1");

league.setCountryCode("FR");

break;

case 3:

league.setPKLeague(3);

league.setName("Bundesliga");

league.setCountryCode("DE");

break;

case 4:

league.setPKLeague(4);

league.setName("Serie A");

league.setCountryCode("IT");

break;

case 5:

league.setPKLeague(5);

league.setName("LaLiga");

league.setCountryCode("ESP");

break;

case 6:

league.setPKLeague(6);

league.setName("Super League");

league.setCountryCode("CH");

break;

default:

break;

}

}

ObjectMapper objectMapper = new ObjectMapper();

Team team = new Team(null, name, code, league);

String jsonTeam = objectMapper.writeValueAsString(team);

teamManager.addTeam(jsonTeam);

} else if (action.equals("teams")) {

System.out.println("slakdfjaslkfsjf");

PrintWriter out = response.getWriter();

out.println("<html><body>");

// get the teams and add them to the HTML response

String teams = teamManager.getTeams();

HashMap<String, List<String>> test = gson.fromJson(teams, HashMap.class);

// Team t = objectMapper.readValue(teamManager.getTeams(), Team.class);

out.println("<select id=\"teams\" name=\"teams\">");

List<String> teamsList = test.get("type");

// teamsList.sort((o1, o2) -> o1.compareTo(o2));

for (int i = 0; i < teamsList.size(); i += 4) {

String team1 = teamsList.get(i);

String team2 = teamsList.get(i + 1);

String leagueName = teamsList.get(i + 2);

String leaguePays = teamsList.get(i + 3);

out.println("<option value=\"" + team1 + "\">" + team1 + " - " + team2 + " - " + leagueName + " - " + leaguePays + "</option>");

}

out.println("</select>");

out.println("</body></html>");

out.close();

} else if (action.equals("delTeam")) {

String teamId = request.getParameter("teamId");

ObjectMapper objectMapper = new ObjectMapper();

String jsonTeamId = objectMapper.writeValueAsString(teamId);

teamManager.deleteTeam(jsonTeamId);

} else if (action.equals("editTeam")) {

String teamId = request.getParameter("teamId");

String name = request.getParameter("name");

String code = request.getParameter("code");

String all = teamId + "-" + name + "-" + code;

teamManager.modifyTeam(all);

} else if (action.equals("login")) {

PrintWriter out = response.getWriter();

HttpSession session = request.getSession();

System.out.println("lasdfjaskfj");

String username = request.getParameter("username");

String password = request.getParameter("password");

System.out.println(username + password);

String users = userManager.getUsers();

String[] splitedUsers = users.split("\"");

boolean isValid = false;

if (splitedUsers[3].equals(username) && splitedUsers[5].equals(password)) {

session.setAttribute("logguer", true);

System.out.println("good");

isValid = true;

} else {

isValid = false;

response.setStatus(400);

}

out.println(isValid);

System.out.println(users);

} else if (action.equals("addUser")) {

PrintWriter out = response.getWriter();

String username = request.getParameter("username");

String password = request.getParameter("password");

System.out.println(username + password);

ObjectMapper objectMapper = new ObjectMapper();

User user = new User(null, username, password);

String jsonUser = objectMapper.writeValueAsString(user);

System.out.println(jsonUser);

userManager.createUser(jsonUser);

} else if (action.equals("delUser")) {

String userId = request.getParameter("userID");

ObjectMapper objectMapper = new ObjectMapper();

String jsonTeamId = objectMapper.writeValueAsString(userId);

userManager.deleteUser(jsonTeamId);

} else if (action.equals("modUser")) {

String userId = request.getParameter("userID");

String username = request.getParameter("username");

String password = request.getParameter("password");

System.out.println(userId + username + password);

String all = userId + "-" + username + "-" + password;

userManager.modifyUser(all);

} else if (action.equals("adminLogin")){

PrintWriter out = response.getWriter();

System.out.println("in writer");

response.setContentType("text/json");

userManager = new UserManager();

String users = userManager.getUsers();

out.println("<html><body>");

// get the users and add them to the HTML response

System.out.println("realusers : " + users);

String[] usersSplited = users.split("\"");

out.println("<select id=\"users\" name=\"users\">");

for (int j = 3; j < usersSplited.length; j += 2) {

String string = usersSplited[j];

out.println("<option value=\"" + string + "\">" + string + "</option>");

}

out.println("</select>");

out.println("</body></html>");

response.setContentType("text/html");

}

System.out.println(action);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

### TeamManager.java

public class TeamManager {

private WebTarget webTarget;

private Client client;

private static final String BASE\_URI = "https://pillerd.emf-informatique.ch/java\_serviceREST1/webresources";

public TeamManager() {

client = javax.ws.rs.client.ClientBuilder.newClient();

webTarget = client.target(BASE\_URI).path("generic");

}

public String modifyTeam(Object requestEntity) throws ClientErrorException {

return webTarget.path("ModifyTeam").request(javax.ws.rs.core.MediaType.TEXT\_PLAIN).post(javax.ws.rs.client.Entity.entity(requestEntity, javax.ws.rs.core.MediaType.APPLICATION\_FORM\_URLENCODED), String.class);

}

public String deleteTeam(Object requestEntity) throws ClientErrorException {

return webTarget.path("DeleteTeam").request(javax.ws.rs.core.MediaType.TEXT\_PLAIN).post(javax.ws.rs.client.Entity.entity(requestEntity, javax.ws.rs.core.MediaType.TEXT\_PLAIN), String.class);

}

public String addTeam(Object requestEntity) throws ClientErrorException {

return webTarget.path("AddTeam").request(javax.ws.rs.core.MediaType.TEXT\_PLAIN).post(javax.ws.rs.client.Entity.entity(requestEntity, javax.ws.rs.core.MediaType.APPLICATION\_FORM\_URLENCODED), String.class);

}

public String getXml() throws ClientErrorException {

WebTarget resource = webTarget;

return resource.request(javax.ws.rs.core.MediaType.APPLICATION\_XML).get(String.class);

}

public void putXml(Object requestEntity) throws ClientErrorException {

webTarget.request(javax.ws.rs.core.MediaType.APPLICATION\_XML).put(javax.ws.rs.client.Entity.entity(requestEntity, javax.ws.rs.core.MediaType.APPLICATION\_XML));

}

public String getTeams() throws ClientErrorException {

WebTarget resource = webTarget;

resource = resource.path("GetTeams");

return resource.request(javax.ws.rs.core.MediaType.APPLICATION\_JSON).get(String.class);

}

public void close() {

client.close();

}

}

### UserManager.java

public class UserManager {

private WebTarget webTarget;

private Client client;

private static final String BASE\_URI = "https://franzenl.emf-informatique.ch/java\_serviceREST2/webresources";

public UserManager() {

client = javax.ws.rs.client.ClientBuilder.newClient();

webTarget = client.target(BASE\_URI).path("generic");

}

public String modifyUser(Object requestEntity) throws ClientErrorException {

return webTarget.path("ModifyUser").request(javax.ws.rs.core.MediaType.TEXT\_PLAIN).post(javax.ws.rs.client.Entity.entity(requestEntity, javax.ws.rs.core.MediaType.APPLICATION\_FORM\_URLENCODED), String.class);

}

public String getUsers() throws ClientErrorException {

System.out.println("getUsers");

WebTarget resource = webTarget;

resource = resource.path("GetUsers");

return resource.request(javax.ws.rs.core.MediaType.APPLICATION\_JSON).get(String.class);

}

public String deleteUser(Object requestEntity) throws ClientErrorException {

return webTarget.path("DeleteUser").request(javax.ws.rs.core.MediaType.TEXT\_PLAIN).post(javax.ws.rs.client.Entity.entity(requestEntity, javax.ws.rs.core.MediaType.APPLICATION\_FORM\_URLENCODED), String.class);

}

public String getXml() throws ClientErrorException {

WebTarget resource = webTarget;

return resource.request(javax.ws.rs.core.MediaType.APPLICATION\_XML).get(String.class);

}

public String createUser(Object requestEntity) throws ClientErrorException {

return webTarget.path("CreateUser").request(javax.ws.rs.core.MediaType.TEXT\_PLAIN).post(javax.ws.rs.client.Entity.entity(requestEntity, javax.ws.rs.core.MediaType.APPLICATION\_FORM\_URLENCODED), String.class);

}

public String isValidUser() throws ClientErrorException {

return webTarget.path("isValidUser").request(javax.ws.rs.core.MediaType.APPLICATION\_FORM\_URLENCODED).post(null, String.class);

}

public void putXml(Object requestEntity) throws ClientErrorException {

webTarget.request(javax.ws.rs.core.MediaType.APPLICATION\_XML).put(javax.ws.rs.client.Entity.entity(requestEntity, javax.ws.rs.core.MediaType.APPLICATION\_XML));

}

public void close() {

client.close();

}

}

## ServiceRest1

### ApplicationConfig.java

public class ApplicationConfig extends Application {

@Override

public Set<Class<?>> getClasses() {

Set<Class<?>> resources = new java.util.HashSet<>();

addRestResourceClasses(resources);

return resources;

}

/\*\*

\* Do not modify addRestResourceClasses() method.

\* It is automatically populated with

\* all resources defined in the project.

\* If required, comment out calling this method in getClasses().

\*/

private void addRestResourceClasses(Set<Class<?>> resources) {

resources.add(restfulservice.TeamManager.class);

}

}

### TeamManager.java

@Path("generic")

public class TeamManager {

private Connection jdbcConnection;

private Wrk wrk;

@Context

private UriInfo context;

/\*\*

\* Creates a new instance of GenericResource

\*/

public TeamManager() {

jdbcConnection = null;

wrk = new Wrk();

}

/\*\*

\* Retrieves representation of an instance of restfulservice.GenericResource

\*

\* @return an instance of java.lang.String

\*/

@GET

@Produces(MediaType.APPLICATION\_XML)

public String getXml() {

//TODO return proper representation object

throw new UnsupportedOperationException();

}

/\*\*

\* PUT method for updating or creating an instance of GenericResource

\*

\* @param content representation for the resource

\*/

@PUT

@Consumes(MediaType.APPLICATION\_XML)

public void putXml(String content) {

}

@GET

@Path("GetTeams")

@Produces(MediaType.APPLICATION\_JSON + ";charset=utf-8")

public String getTeams() {

Gson builder = new Gson();

String toJson = builder.toJson(wrk.lireTeams());

return "{\"type\":" + toJson + "}";

}

@POST

@Path("AddTeam")

@Produces(MediaType.TEXT\_PLAIN)

@Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)

public String addTeam(String json) {

String s = "";

try {

Gson builder = new Gson();

ObjectMapper objectMapper = new ObjectMapper();

Team t = objectMapper.readValue(json, Team.class);

if (wrk.ajouterTeam(t.getName(), t.getCode(), t.getFKLeague().getPKLeague())) {

s = "OK";

} else {

s = "KO";

}

} catch (Exception e) {

System.out.println(e.getMessage());

}

return s;

}

@POST

@Path("DeleteTeam")

@Produces(MediaType.TEXT\_PLAIN)

@Consumes(MediaType.TEXT\_PLAIN)

public String deleteTeam(String json) {

String s = "";

try {

Gson builder = new Gson();

ObjectMapper objectMapper = new ObjectMapper();

String t = objectMapper.readValue(json, String.class);

if (wrk.supprimerTeam(t)) {

s = "OK";

} else {

s = "KO";

}

} catch (Exception e) {

System.out.println(e.getMessage());

}

return s;

}

@POST

@Path("ModifyTeam")

@Produces(MediaType.TEXT\_PLAIN)

@Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)

public String modifyTeam(String json) {

String s = "";

try {

Gson builder = new Gson();

ObjectMapper objectMapper = new ObjectMapper();

String[] split = json.split("-");

if (wrk.modifierTeam(split[0], split[1], split[2])) {

s = "OK";

} else {

s = "KO";

}

} catch (Exception e) {

System.out.println(e.getMessage());

}

return s;

}

}

### Wrk.java

public class Wrk {

private Connection jdbcConnection;

public Wrk() {

jdbcConnection = null;

}

public boolean dbConnect() {

final String url = "jdbc:mysql://pillerd.emf-informatique.ch:3306/pillerd\_dbTeam?serverTimezone=CET";

// final String user = "root";

// final String pw = "";

final String user = "pillerd\_admin";

final String pw = "Y^@[&D9hyr$Q";

boolean result = false;

try {

//nécessaire pour fonctionnement en web

Class.forName("com.mysql.cj.jdbc.Driver");

} catch (ClassNotFoundException ex) {

System.out.println("Connexion au driver JDBC à échoué!\n" + ex.getMessage());

}

try {

jdbcConnection = DriverManager.getConnection(url, user, pw);

System.out.println("Connection successfull");

result = true;

} catch (SQLException ex) {

System.out.println("Connexion à la BD a échouée!\n" + ex.getMessage());

}

return result;

}

public boolean dbDisconnect() {

boolean ok = false;

if (jdbcConnection != null) {

try {

jdbcConnection.close();

jdbcConnection = null;

ok = true;

} catch (SQLException ex) {

Logger.getLogger(Wrk.class.getName()).log(Level.SEVERE, null, ex);

}

}

return ok;

}

public ArrayList<String> lireTeams() {

ArrayList<String> resultat = new ArrayList<>();

Statement stmt = null;

ResultSet rs = null;

if (dbConnect()) {

try {

if ((stmt = jdbcConnection.createStatement()) != null) {

String sql = "SELECT PK\_Team, t\_team.Name, Code, t\_league.Name, Country\_Code FROM pillerd\_dbTeam.t\_team INNER JOIN pillerd\_dbTeam.t\_league ON t\_team.FK\_League = t\_league.PK\_League";

if ((rs = stmt.executeQuery(sql)) != null) {

while (rs.next()) {

String s = rs.getString("t\_team.Name");

String s1 = rs.getString("Code");

String s2 = rs.getString("t\_league.Name");

String s3 = rs.getString("Country\_Code");

resultat.add(s);

resultat.add(s1);

resultat.add(s2);

resultat.add(s3);

}

}

rs.close();

rs = null;

stmt.close();

stmt.close();

}

} catch (SQLException ex) {

Logger.getLogger(Wrk.class.getName()).log(Level.SEVERE, null, ex);

} finally {

dbDisconnect();

try {

if (rs != null) {

rs.close();

rs = null;

}

} catch (SQLException ex) {

Logger.getLogger(Wrk.class.getName()).log(Level.SEVERE, null, ex);

}

try {

if (stmt != null) {

stmt.close();

stmt = null;

}

} catch (SQLException ex) {

Logger.getLogger(Wrk.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

return resultat;

}

public boolean ajouterTeam(String name, String code, int league) {

boolean ok = false;

PreparedStatement prestmt = null;

ResultSet rs = null;

if (dbConnect()) {

try {

String sql = "INSERT INTO pillerd\_dbTeam.t\_team (t\_team.Name, Code, FK\_League) values (?, ?, ?)";

if ((prestmt = jdbcConnection.prepareStatement(sql, Statement.RETURN\_GENERATED\_KEYS)) != null) {

prestmt.setString(1, name);

prestmt.setString(2, code);

prestmt.setInt(3, league);

prestmt.executeUpdate();

if ((rs = prestmt.getGeneratedKeys()) != null) {

if (rs.next()) {

ok = true;

}

}

}

prestmt.close();

prestmt = null;

rs.close();

rs = null;

} catch (SQLException ex) {

Logger.getLogger(Wrk.class.getName()).log(Level.SEVERE, null, ex);

} finally {

dbDisconnect();

try {

if (rs != null) {

rs.close();

rs = null;

}

} catch (SQLException ex) {

Logger.getLogger(Wrk.class.getName()).log(Level.SEVERE, null, ex);

}

try {

if (prestmt != null) {

prestmt.close();

prestmt = null;

}

} catch (SQLException ex) {

Logger.getLogger(Wrk.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

return ok;

}

public boolean supprimerTeam(String name) {

boolean ok = false;

PreparedStatement prestmt = null;

if (dbConnect()) {

try {

String sql = "DELETE FROM t\_team WHERE t\_team.Name = ?";

if ((prestmt = jdbcConnection.prepareStatement(sql)) != null) {

prestmt.setString(1, name);

int numRowsDeleted = prestmt.executeUpdate();

if (numRowsDeleted > 0) {

ok = true;

}

}

prestmt.close();

prestmt = null;

} catch (SQLException ex) {

Logger.getLogger(Wrk.class.getName()).log(Level.SEVERE, null, ex);

} finally {

dbDisconnect();

try {

if (prestmt != null) {

prestmt.close();

}

} catch (SQLException ex) {

Logger.getLogger(Wrk.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

return ok;

}

public boolean modifierTeam(String oldName, String newName, String newCode) {

boolean ok = false;

PreparedStatement prestmt = null;

if (dbConnect()) {

try {

String sql = "UPDATE t\_team SET t\_team.Name=?, Code=? WHERE t\_team.Name=?";

if ((prestmt = jdbcConnection.prepareStatement(sql)) != null) {

prestmt.setString(1, newName);

prestmt.setString(2, newCode);

prestmt.setString(3, oldName);

int numRowsAffected = prestmt.executeUpdate();

if (numRowsAffected > 0) {

ok = true;

}

}

prestmt.close();

prestmt = null;

} catch (SQLException ex) {

Logger.getLogger(Wrk.class.getName()).log(Level.SEVERE, null, ex);

} finally {

dbDisconnect();

try {

if (prestmt != null) {

prestmt.close();

prestmt = null;

}

} catch (SQLException ex) {

Logger.getLogger(Wrk.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

return ok;

}

}

## ServiceRest2

### ApplicationConfig.java

@javax.ws.rs.ApplicationPath("webresources")

public class ApplicationConfig extends javax.ws.rs.core.Application {

@Override

public Set<Class<?>> getClasses() {

Set<Class<?>> resources = new java.util.HashSet<>();

addRestResourceClasses(resources);

return resources;

}

/\*\*

\* Do not modify addRestResourceClasses() method. It is automatically

\* populated with all resources defined in the project. If required, comment

\* out calling this method in getClasses().

\*/

private void addRestResourceClasses(Set<Class<?>> resources) {

resources.add(rest.UserManager.class);

}

}

### UserManager.java

@Path("generic")

public class UserManager {

@Context

private UriInfo context;

private WrkDB wrkDB;

/\*\*

\* Creates a new instance of GenericResource

\*/

public UserManager() {

wrkDB = new WrkDB();

}

/\*\*

\* Retrieves representation of an instance of controllers.UserManager

\*

\* @return an instance of java.lang.String

\*/

@GET

@Produces(javax.ws.rs.core.MediaType.APPLICATION\_XML)

public String getXml() {

//TODO return proper representation object

throw new UnsupportedOperationException();

}

/\*\*

\* PUT method for updating or creating an instance of UserManager

\*

\* @param content representation for the resource

\*/

@PUT

@Consumes(javax.ws.rs.core.MediaType.APPLICATION\_XML)

public void putXml(String content) {

}

@GET

@Path("GetUsers")

@Produces(MediaType.APPLICATION\_JSON + ";charset=utf-8")

public String getUsers() {

Gson builder = new Gson();

String toJson = builder.toJson(wrkDB.dbGetUsers());

return "{\"type\":" + toJson + "}";

}

@POST

@Path("CreateUser")

@Produces(MediaType.TEXT\_PLAIN)

@Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)

public String createUser(String json) {

String s = "";

try {

System.out.println(json);

Gson builder = new Gson();

ObjectMapper objectMapper = new ObjectMapper();

User t = objectMapper.readValue(json, User.class);

System.out.println(t.getUsername() + "..." + t.getPassword());

if (wrkDB.dbCreateUser(t.getUsername(), t.getPassword())) {

s = "OK";

} else {

s = "KO";

}

} catch (Exception e) {

System.out.println(e.getMessage());

}

return s;

}

@POST

@Path("ModifyUser")

@Produces(MediaType.TEXT\_PLAIN)

@Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)

public String modifyUser(String json) {

String s = "";

try {

System.out.println(json);

Gson builder = new Gson();

ObjectMapper objectMapper = new ObjectMapper();

String[] split = json.split("-");

System.out.println(split[0] + split[1] + split[2]);

if (wrkDB.dbModifyUser(split[0], split[1], split[2])) {

s = "OK";

} else {

s = "KO";

}

System.out.println(s);

} catch (Exception e) {

System.out.println("catch?");

System.out.println(e.getMessage());

}

return s;

}

@POST

@Path("DeleteUser")

@Produces(MediaType.TEXT\_PLAIN)

@Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)

public String deleteUser(String json) {

String s = "";

try {

Gson builder = new Gson();

ObjectMapper objectMapper = new ObjectMapper();

String t = objectMapper.readValue(json, String.class);

if (wrkDB.dbDeleteUser(t)) {

s = "OK";

} else {

s = "KO";

}

} catch (Exception e) {

System.out.println(e.getMessage());

}

return s;

}

@POST

@Path("isValidUser")

@Produces(MediaType.TEXT\_PLAIN)

@Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)

public String isValidUser(@FormParam("username") String username, @FormParam("password") String password) {

String s;

try {

//User u = wrkDB.getUser(pk\_user);

wrkDB.isValidUser(username, password);

s = "OK";

} catch (Exception e) {

s = "KO";

}

return s;

}

}

### WrkDB.java

public class WrkDB {

private String port;

private String dbName;

private Connection dbConnection;

private String URL;

public WrkDB() {

this.dbConnection = null;

this.port = "3306";

this.dbName = "franzenl\_dbuser";

this.URL = "jdbc:mysql://franzenl.emf-informatique.ch:3306/franzenl\_dbuser?serverTimezone=Europe/Zurich";

dbConnect();

}

//Faire la connexion DB

public boolean dbConnect() {

boolean ok = false;

try {

if (dbConnection == null) {

Class.forName("com.mysql.cj.jdbc.Driver");

dbConnection = DriverManager.getConnection(URL, "franzenl\_admin", "~O(@Fpn^F)3o");

ok = true;

}

} catch (SQLException b) {

Logger.getLogger(WrkDB.class.getName()).log(Level.SEVERE, null, b);

} catch (ClassNotFoundException ex) {

Logger.getLogger(WrkDB.class.getName()).log(Level.SEVERE, null, ex);

}

return ok;

}

//Efface connexion DB

public boolean dbDisconnect() {

boolean ok = false;

// On vérifie si une connexion est toujours présente (donc pas nulle)

if (dbConnection != null) {

try {

// On essaie de fermer la connexion, puis "vide" la variable.

dbConnection.close();

dbConnection = null;

ok = true;

} catch (SQLException ex) {

Logger.getLogger(WrkDB.class.getName()).log(Level.SEVERE, null, ex);

}

}

return ok;

}

//test if db is connected

public boolean isConnected() {

boolean result = false;

try {

if (!dbConnection.isClosed()) {

result = true;

}

} catch (SQLException ex) {

Logger.getLogger(WrkDB.class.getName()).log(Level.SEVERE, null, ex);

}

return result;

}

public ArrayList<String> dbGetUsers() {

ArrayList<String> resultat = new ArrayList<>();

Statement stmt = null;

ResultSet rs = null;

if (isConnected()) {

try {

if ((stmt = dbConnection.createStatement()) != null) {

String sql = "SELECT \* FROM t\_user";

if ((rs = stmt.executeQuery(sql)) != null) {

while (rs.next()) {

String s = rs.getString("t\_user.username");

String s1 = rs.getString("t\_user.password");

resultat.add(s);

resultat.add(s1);

}

}

rs.close();

rs = null;

stmt.close();

stmt.close();

}

} catch (SQLException ex) {

Logger.getLogger(WrkDB.class.getName()).log(Level.SEVERE, null, ex);

} finally {

dbDisconnect();

try {

if (rs != null) {

rs.close();

rs = null;

}

} catch (SQLException ex) {

Logger.getLogger(WrkDB.class.getName()).log(Level.SEVERE, null, ex);

}

try {

if (stmt != null) {

stmt.close();

stmt = null;

}

} catch (SQLException ex) {

Logger.getLogger(WrkDB.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

return resultat;

}

//Modify User

public boolean dbModifyUser(String oldname, String username, String password) throws DBException {

boolean result = false;

PreparedStatement ps = null;

try {

String prep = "update t\_user set username=?, password=? where username=?";

if ((ps = dbConnection.prepareStatement(prep)) != null) {

ps.setString(1, username);

ps.setString(2, password);

ps.setString(3, oldname);

int nb = ps.executeUpdate();

if (nb > 0) {

result = true;

}

}

ps.close();

} catch (SQLException ex) {

throw new DBException(ex.getMessage());

}

return result;

}

//Delete User

public boolean dbDeleteUser(String name) throws DBException {

boolean result = false;

String prep = "delete from t\_user where username=?";

try ( PreparedStatement ps = dbConnection.prepareStatement(prep)) {

ps.setString(1, name);

int nb = ps.executeUpdate();

result = true;

if (nb != 1) {

throw new DBException("Erreur de mise à jour !!!");

}

} catch (SQLException ex) {

throw new DBException(ex.getMessage());

}

return result;

}

public boolean dbCreateUser(String username, String password) {

boolean result = false;

PreparedStatement prestmt = null;

ResultSet rs = null;

System.out.println("before if connect");

if (isConnected()) {

try {

System.out.println("before if");

String sql = "INSERT INTO t\_user (t\_user.username, t\_user.password) values (?, ?)";

if ((prestmt = dbConnection.prepareStatement(sql, Statement.RETURN\_GENERATED\_KEYS)) != null) {

System.out.println("in if");

prestmt.setString(1, username);

prestmt.setString(2, password);

System.out.println("before executeUpdate");

System.out.println("username: " + username + ", pass: " + password);

System.out.println("prestatement: " + prestmt);

prestmt.executeUpdate();

System.out.println("after executeUpdate");

if ((rs = prestmt.getGeneratedKeys()) != null) {

if (rs.next()) {

result = true;

}

}

}

prestmt.close();

prestmt = null;

rs.close();

rs = null;

} catch (SQLException ex) {

Logger.getLogger(WrkDB.class.getName()).log(Level.SEVERE, null, ex);

} finally {

dbDisconnect();

try {

if (rs != null) {

rs.close();

rs = null;

}

} catch (SQLException ex) {

Logger.getLogger(WrkDB.class.getName()).log(Level.SEVERE, null, ex);

}

try {

if (prestmt != null) {

prestmt.close();

prestmt = null;

}

} catch (SQLException ex) {

Logger.getLogger(WrkDB.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

return result;

}

public boolean isValidUser(String username, String password) {

boolean result = false;

if (isConnected()) {

try {

User userToCompare = getUser(username);

if (userToCompare.getUsername().equals(username) && userToCompare.getPassword().equals(password)) {

result = true;

}

} catch (Exception ex) {

System.out.println(ex.getMessage());

}

}

return result;

}

public User getUser(String username) throws DBException {

// boolean result = dbConnect();

boolean result = true;

User user = new User();

if (result) {

System.out.println("connection ok");

PreparedStatement ps = null;

try {

ps = dbConnection.prepareStatement("SELECT \* FROM dbuser.t\_user WHERE username=" + username);

ResultSet rs = ps.executeQuery();

while (rs.next()) {

user.setPKUser(Integer.parseInt(rs.getString(1)));

user.setUsername(rs.getString(2));

user.setPassword(rs.getString(3));

}

rs.close();

result = true;

System.out.println("OK");

} catch (Exception ex) {

System.out.println(ex.getMessage());

}

if (result) {

result = dbDisconnect();

}

}

return user;

}

}

# Conclusion

Nous avons eu quelques problèmes avec le téléchargement de la passerelle sur Internet, mais nous les avons plus ou moins résolus. En outre, nous n'avons pas très bien réussi la connexion. Dans l'ensemble, le module nous a bien plu dans le groupe et nous l'avons relativement bien réussi vers la fin. Nous avons certainement appris beaucoup de choses sur la création d'un site web dynamique sur plusieurs applications.