|  |  |
| --- | --- |
| GRAND LIST Projet module 133  S:\EMF\ElevesCommun\2015-2016\Waeberla\Logos\Logo_EMF-Informatique_FR_RVB_50.jpghttp://www.emf.ch/sites/default/files/images/logus/id_logus_fr.png | |
|  | http://www.emf.ch/sites/default/files/images/logus/id_logus_fr.png  Malcolm Gfeller et De Sousa Raphaël  Groupe 1 et Classe 300232  Module 133 du 23.03.2023 au 05.05.2023 |
|  |  |

**Table des matières**

1 Introduction et contexte du projet I

2 Analyse à faire complètement avec EA I

2.1 Use case I

2.1.1 Global I

2.1.2 Client 1 II

2.1.3 Client 2 III

2.1.4 Gateway IV

2.1.5 Service rest 1 V

2.1.6 Service rest 2 VI

2.2 Diagramme de séquence système VII

2.2.1 CheckLogin() VII

2.2.2 AfficherAnimes() VII

3 Conception à faire complétement avec EA VII

4 Bases de données VIII

4.1 Utilisateur VIII

4.2 Anime VIII

5 Implémentation des applications client Ap1 et client Ap2 VIII

6 Implémentation de l'aplication API Gateway VIII

7 Implémentation des aplications Service Rest1 et Service Rest2 VIII

7.1 Service Rest 1 VIII

7.1.1 Rest VIII

7.1.2 Wrk X

7.2 Service Rest 2 XII

7.2.1 Rest XII

7.2.2 Wrk XII

8 Hébergement XII

9 Installation du projet complet avec les 5 applications XII

10 Outils, langages (versions, définitions, installations) XII

11 Tests de fonctionnement du projet XII

11.1 Service Rest 1 XIII

11.1.1 GetUser XIII

11.1.2 Login XIV

12 Auto-évaluations et conclusions XIV

# Introduction et contexte du projet

# Analyse à faire complètement avec EA

## Use case

### Global

Une image contenant diagramme

Description générée automatiquement

### Client 1

Une image contenant diagramme, schématique

Description générée automatiquement

### Client 2

Une image contenant diagramme

Description générée automatiquement

### Gateway

Une image contenant diagramme, schématique

Description générée automatiquement

### Service rest 1

Une image contenant diagramme, schématique

Description générée automatiquement

### Service rest 2

Une image contenant diagramme, schématique

Description générée automatiquement

## Diagramme de séquence système

### CheckLogin()

Une image contenant graphique

Description générée automatiquement

### AfficherAnimes()

# Conception à faire complétement avec EA

3.1 Class Diagram complet avec les explications de chaque application

3.2 Navigation Diagram complet avec les explications des applications

# Bases de données

## Utilisateur

Une image contenant texte

Description générée automatiquement

## Anime

# Implémentation des applications client Ap1 et client Ap2

5.1 Application client 1

5.2 Application client 2

# Implémentation de l'aplication API Gateway

6.1 Partie Servlet sur serveur Tomcat commun

# Implémentation des aplications Service Rest1 et Service Rest2

## Service Rest 1

### Rest

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/WebServices/GenericResource.java to edit this template

\*/

package rest;

import com.google.gson.Gson;

import java.util.HashMap;

import javax.ws.rs.core.Context;

import javax.ws.rs.core.UriInfo;

import javax.ws.rs.Consumes;

import javax.ws.rs.Produces;

import javax.ws.rs.GET;

import javax.ws.rs.Path;

import javax.ws.rs.PUT;

import javax.ws.rs.QueryParam;

import javax.ws.rs.core.MediaType;

import worker.Wrk;

/\*\*

\* REST Web Service

\*

\* @author desousar

\*/

@Path("user")

public class Rest {

@Context

private UriInfo context;

private Wrk wrk;

/\*\*

\* Creates a new instance of Rest

\*/

public Rest() {

wrk = new Wrk();

}

/\*\*

\* Retrieves representation of an instance of rest.Rest

\*

\* @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 Rest

\*

\* @param content representation for the resource

\*/

@PUT

@Consumes(MediaType.APPLICATION\_XML)

public void putXml(String content) {

}

@GET

@Path("getAuthor")

@Produces(javax.ws.rs.core.MediaType.TEXT\_PLAIN)

public String getAuthor() {

String test = "";

System.out.println("test" + test);

return " Malcolm Gfeller";

}

@GET

@Path("getUser")

@Produces(MediaType.APPLICATION\_JSON)

public String getUser(@QueryParam("PK") int pk) {

Gson gson = new Gson();

String json;

try {

json = gson.toJson(wrk.getUser(pk));

} catch (Exception e) {

json = gson.toJson("error");

}

return json;

}

@GET

@Path("login")

@Produces(MediaType.APPLICATION\_JSON)

public String login(@QueryParam("username") String username, @QueryParam("password") String password) {

Gson gson = new Gson();

HashMap<String, String> result = new HashMap<>();

try {

boolean isValid = wrk.checkLogin(username, password);

if (isValid) {

result.put("status", "success");

} else {

result.put("status", "fail");

}

} catch (Exception e) {

result.put("status", "error");

result.put("message", e.getMessage());

}

return gson.toJson(result);

}

}

### Wrk

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package worker;

import com.google.common.hash.Hashing;

import java.nio.charset.StandardCharsets;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author desousar

\*/

public class Wrk {

private Connection jdbcConnection;

public Wrk() {

jdbcConnection = null;

}

private boolean dbConnect() {

boolean ok = false;

try {

if (jdbcConnection == null) {

//On spécifie que notre driver est JDBC

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

//L'URL se compose de l'adresse de notre base de données, ainsi que

//la base de données à utiliser.

String url = "jdbc:mysql://gfellerm01.emf-informatique.ch:3306/gfellerm01\_133-Projet-Utilisateurs?serverTimezone=CET";

//On essaie de se connecter à notre URL à partir d'un identifiant.

//Ici, le nom d'utilisateur est "root", et il n'y a pas de mot de passe.

//Si la connexion est réussie, la méthode "getConnection" renverra "true".

jdbcConnection = DriverManager.getConnection(url, "gfellerm01\_133-Project", "Pa$$w0rdena");

ok = true;

}

} catch (SQLException b) {

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

} catch (ClassNotFoundException ex) {

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

}

//Si la connexion s'est bien passée, on renvoie "true", sinon "false".

return ok;

}

private boolean dbDisconnect() {

boolean ok = false;

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

if (jdbcConnection != null) {

try {

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

jdbcConnection.close();

jdbcConnection = null;

ok = true;

} catch (SQLException ex) {

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

}

}

return ok;

}

public ArrayList<String> getUser(int PK) {

ArrayList<String> lstUser = null;

boolean result = dbConnect();

if (result) {

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

PreparedStatement ps = null;

String user = "";

lstUser = new ArrayList<String>();

try {

ps = jdbcConnection.prepareStatement("SELECT \* FROM t\_user where PK\_User = " + PK + ";");

ResultSet rs = ps.executeQuery();

while (rs.next()) {

user = (String) rs.getString(1);

user += "," + (String) rs.getString(2);

lstUser.add(user);

}

rs.close();

result = true;

System.out.println("OK");

} catch (Exception ex) {

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

}

if (result) {

result = dbDisconnect();

}

}

return lstUser;

}

public boolean checkLogin(String username, String password) {

boolean success = false;

boolean result = dbConnect();

if (result) {

try {

PreparedStatement ps = jdbcConnection.prepareStatement("SELECT \* FROM t\_user WHERE Username = ?");

ps.setString(1, username);

ResultSet rs = ps.executeQuery();

while (rs.next()) {

String hashedPassword = rs.getString("Password").toUpperCase();

// On calcule le hash SHA256 du mot de passe fourni par l'utilisateur

String inputHash = Hashing.sha256().hashString(password, StandardCharsets.UTF\_8).toString().toUpperCase();

if (hashedPassword.equals(inputHash)) {

// Si les mots de passe correspondent, la connexion est réussie

success = true;

}

}

rs.close();

} catch (SQLException ex) {

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

} finally {

result = dbDisconnect();

}

}

return success;

}

}

## Service Rest 2

### Rest

### Wrk

# Hébergement

# Installation du projet complet avec les 5 applications

# Outils, langages (versions, définitions, installations)

# Tests de fonctionnement du projet

## Service Rest 1

### GetUser

Une image contenant texte, écran, capture d’écran, noir

Description générée automatiquement

### Login

Une image contenant texte, écran, capture d’écran, noir

Description générée automatiquement

# Auto-évaluations et conclusions