|  |  |
| --- | --- |
| 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 XV

8 Hébergement XXII

9 Installation du projet complet avec les 5 applications XXII

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

11 Tests de fonctionnement du projet XXII

11.1 Service Rest 1 XXII

11.1.1 GetUser XXII

11.1.2 Login XXIV

11.2 Service Rest 2 XXV

11.2.1 getManga XXV

11.2.2 ajoutManga XXV

11.2.3 modifyManga XXVI

11.2.4 deleteManga XXVI

11.2.5 getFavoris XXVII

11.2.6 ajoutfavoris XXVII

11.2.7 modifyFavoris XXVIII

11.2.8 deleteFavoris XXVIII

12 Auto-évaluations et conclusions XXVIII

# Introduction et contexte du projet

Nous allons faire un site web qui vas permettre d’avoir une liste de manga et une liste de favoris par utilisateur. Nous avons choisi de faire notre site sure le thème de manga car nous aimons tous les 2 ce thèmes.

# 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

/\*

\* 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.ArrayList;

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.DELETE;

import javax.ws.rs.FormParam;

import javax.ws.rs.Produces;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

import javax.ws.rs.Path;

import javax.ws.rs.PUT;

import javax.ws.rs.core.MediaType;

import worker.Wrk;

/\*\*

\* REST Web Service

\*

\* @author desousar

\*/

@Path("manga")

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 " de Sousa Raphael";

}

@GET

@Path("getManga")

@Produces(MediaType.TEXT\_PLAIN)

public String getManga() {

Gson builder = new Gson();

// Le constructeur va transformer notre ArrayList de résultats dans un format JSON.

String manga = builder.toJson(wrk.lireManga());

// On affiche notre résultat.

return manga;

}

@POST

@Path("ajoutManga")

@Produces(MediaType.TEXT\_PLAIN)

public String putManga(@FormParam("nomDuManga") String nomDuManga, @FormParam("nomDuTome") String nomDuTome, @FormParam("numDuTome") String numeroDuTome, @FormParam("image") String image) {

Gson gson = new Gson();

// Le constructeur va transformer notre ArrayList de résultats dans un format JSON.

boolean ok = wrk.addManga(nomDuTome, nomDuManga, numeroDuTome, image);

// On affiche notre résultat.

return gson.toJson(ok);

}

@PUT

@Path("modifyManga")

@Produces(MediaType.TEXT\_PLAIN)

public String modifyManga(@FormParam("Pk") int pk,@FormParam("nomDuManga") String nomDuManga, @FormParam("nomDuTome") String nomDuTome, @FormParam("numDuTome") String numeroDuTome, @FormParam("image") String image) {

Gson gson = new Gson();

// Le constructeur va transformer notre ArrayList de résultats dans un format JSON.

boolean ok = wrk.modifyManga(pk,nomDuTome, nomDuManga, numeroDuTome, image);

// On affiche notre résultat.

return gson.toJson(ok);

}

@DELETE

@Path("deleteManga")

@Produces(MediaType.TEXT\_PLAIN)

public String deleteManga(@FormParam("Pk") int pk) {

Gson gson = new Gson();

// Le constructeur va transformer notre ArrayList de résultats dans un format JSON.

boolean ok = wrk.deleteManga(pk);

// On affiche notre résultat.

return gson.toJson(ok);

}

@GET

@Path("getFavoris")

@Produces(MediaType.TEXT\_PLAIN)

public String getMangaUser() {

Gson builder = new Gson();

// Le constructeur va transformer notre ArrayList de résultats dans un format JSON.

String mangaUser = builder.toJson(wrk.lireFavoris());

// On affiche notre résultat.

return mangaUser;

}

@POST

@Path("ajoutFavoris")

@Produces(MediaType.TEXT\_PLAIN)

public String addFavoris(@FormParam("fkUser") int fkUser,@FormParam("fkManga") int fkManga) {

Gson gson = new Gson();

// Le constructeur va transformer notre ArrayList de résultats dans un format JSON.

boolean ok = wrk.addFavoris(fkUser,fkManga);

// On affiche notre résultat.

return gson.toJson(ok);

}

@PUT

@Path("modifyFavoris")

@Produces(MediaType.TEXT\_PLAIN)

public String modifyFavoris(@FormParam("newFkUser") int newFkUser,@FormParam("newFkManga") int newFkManga, @FormParam("oldFkUser") int oldFkUser, @FormParam("oldFkManga") int oldFkManga) {

Gson gson = new Gson();

// Le constructeur va transformer notre ArrayList de résultats dans un format JSON.

boolean ok = wrk.modifyFavoris(newFkUser,newFkManga,oldFkUser,oldFkManga);

// On affiche notre résultat.

return gson.toJson(ok);

}

@DELETE

@Path("deleteFavoris")

@Produces(MediaType.TEXT\_PLAIN)

public String deleteFavoris(@FormParam("FKM") int fkManga,@FormParam("FKU") int fkUser) {

Gson gson = new Gson();

// Le constructeur va transformer notre ArrayList de résultats dans un format JSON.

boolean ok = wrk.deleteFavoris(fkUser,fkManga);

// On affiche notre résultat.

return gson.toJson(ok);

}

}

### 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 java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

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://desousar.emf-informatique.ch:3306/desousar\_133\_gestionmanga?serverTimezone=CET";

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

//Ici, le nom d'utilisateur est "desousar\_root133",et le mot de pass est {Y[zROWnN;\*& .

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

jdbcConnection = DriverManager.getConnection(url, "desousar\_root133", "{Y[zROWnN;\*&");

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> lireManga() {

//On prépare nos variables.

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

Statement stmt = null;

ResultSet rs = null;

//On essaie de se connecter à la base de données.

if (dbConnect()) {

try {

//On essaie de créer une requête grâce à notre connexion.

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

//Ce string est la requête SQL qui va récupérer les enregistrements.

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

//On exécute la requête et on stocke la réponse dans un "ResulSet"

//Si notre "ResulSet" contient quelque chose, c'est qu'on a reçu une réponse !

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

//On effectue une boucle qui va parcourir notre résultat.

//La méthode ".next()" avance d'un index renvoie "true"

//s'il y a un résultat. Sinon, "false" quand il atteint

//un enregistrement null.

while (rs.next()) {

//On stocke le nom récupéré de la colonne "NomDuManga".

String s = "";

s = rs.getString(2) + " " + rs.getString(3) + " " + rs.getString(4) + " " + rs.getString(5);

resultat.add(s);

//On stocke notre String dans notre résultat final.

}

}

//On ferme le tout pour optimiser les performances.

rs.close();

rs = null;

stmt.close();

stmt.close();

}

} catch (SQLException ex) {

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

} //On repasse les variables pour vérifier que tout est bien fermé.

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 addManga(String nomDuTome, String nomDuManga, String numeroDuTome, String image) {

boolean ok = false;

Statement stmt = null;

ResultSet rs = null;

if (dbConnect()) {

PreparedStatement ps = null;

try {

//On essaie de créer une requête grâce à notre connexion.

// Create a SQL statement

ps = jdbcConnection.prepareStatement("INSERT INTO t\_manga (NomDuManga, NomDuTome, NumeroDuTome, Image) VALUES (?, ?, ?, ?)");

// Create a PreparedStatement object

ps.setString(1, nomDuManga);

ps.setString(2, nomDuTome);

ps.setString(3, numeroDuTome);

ps.setString(4, image);

// Execute the SQL statement

int rowsInserted = ps.executeUpdate();

ok = rowsInserted > 0;

//On ferme le tout pour optimiser les performances.

} catch (SQLException ex) {

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

} //On repasse les variables pour vérifier que tout est bien fermé.

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 (ps != null) {

ps.close();

ps = null;

}

} catch (SQLException ex) {

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

}

}

}

return ok;

}

public boolean modifyManga(int pk,String nomDuTome, String nomDuManga, String numeroDuTome, String image) {

boolean ok = false;

boolean opendb = dbConnect();

ResultSet rs = null;

if (opendb) {

PreparedStatement ps = null;

try {

//On essaie de créer une requête grâce à notre connexion.

// Create a SQL statement

ps = jdbcConnection.prepareStatement("Update t\_manga SET NomDuManga = ?, NomDuTome = ?, NumeroDuTome = ?, Image=? WHERE PK\_Manga = ?");

// Create a PreparedStatement object

ps.setString(1, nomDuManga);

ps.setString(2, nomDuTome);

ps.setString(3, numeroDuTome);

ps.setString(4, image);

ps.setInt(5, pk);

ps.executeUpdate();

ok =true;

//On ferme le tout pour optimiser les performances.

} catch (SQLException ex) {

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

} //On repasse les variables pour vérifier que tout est bien fermé.

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 (ps != null) {

ps.close();

ps = null;

}

} catch (SQLException ex) {

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

}

}

}

return ok;

}

public boolean deleteManga(int pk) {

boolean ok = false;

boolean opendb = dbConnect();

ResultSet rs = null;

if (opendb) {

PreparedStatement ps = null;

try {

//On essaie de créer une requête grâce à notre connexion.

// Create a SQL statement

ps = jdbcConnection.prepareStatement("DELETE FROM t\_manga WHERE PK\_Manga = ?");

// Create a PreparedStatement object

ps.setInt(1, pk);

ps.executeUpdate();

ok =true;

//On ferme le tout pour optimiser les performances.

} catch (SQLException ex) {

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

} //On repasse les variables pour vérifier que tout est bien fermé.

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 (ps != null) {

ps.close();

ps = null;

}

} catch (SQLException ex) {

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

}

}

}

return ok;

}

public ArrayList<String> lireFavoris() {

//On prépare nos variables.

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

Statement stmt = null;

ResultSet rs = null;

//On essaie de se connecter à la base de données.

if (dbConnect()) {

try {

//On essaie de créer une requête grâce à notre connexion.

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

//Ce string est la requête SQL qui va récupérer les enregistrements.

String sql = "SELECT \* FROM tr\_manga\_user";

//On exécute la requête et on stocke la réponse dans un "ResulSet"

//Si notre "ResulSet" contient quelque chose, c'est qu'on a reçu une réponse !

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

//On effectue une boucle qui va parcourir notre résultat.

//La méthode ".next()" avance d'un index renvoie "true"

//s'il y a un résultat. Sinon, "false" quand il atteint

//un enregistrement null.

while (rs.next()) {

//On stocke le nom récupéré de la colonne "NomDuManga".

String s = "";

s = rs.getString(1) + " " + rs.getString(2);

resultat.add(s);

//On stocke notre String dans notre résultat final.

}

}

//On ferme le tout pour optimiser les performances.

rs.close();

rs = null;

stmt.close();

stmt.close();

}

} catch (SQLException ex) {

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

} //On repasse les variables pour vérifier que tout est bien fermé.

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 addFavoris(int fkUser, int fkManga) {

boolean ok = false;

Statement stmt = null;

ResultSet rs = null;

if (dbConnect()) {

PreparedStatement ps = null;

try {

//On essaie de créer une requête grâce à notre connexion.

// Create a SQL statement

ps = jdbcConnection.prepareStatement("INSERT INTO tr\_manga\_user (FK\_user, FK\_manga) VALUES (?, ?)");

// Create a PreparedStatement object

ps.setInt(1, fkUser);

ps.setInt(2, fkManga);

// Execute the SQL statement

int rowsInserted = ps.executeUpdate();

ok = rowsInserted > 0;

//On ferme le tout pour optimiser les performances.

} catch (SQLException ex) {

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

} //On repasse les variables pour vérifier que tout est bien fermé.

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 (ps != null) {

ps.close();

ps = null;

}

} catch (SQLException ex) {

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

}

}

}

return ok;

}

public boolean modifyFavoris(int newFkUser,int newFkManga,int oldFkUser,int oldFkManga) {

boolean ok = false;

boolean opendb = dbConnect();

ResultSet rs = null;

if (opendb) {

PreparedStatement ps = null;

try {

//On essaie de créer une requête grâce à notre connexion.

// Create a SQL statement

ps = jdbcConnection.prepareStatement("Update tr\_manga\_user SET FK\_user = ?, FK\_manga = ? WHERE FK\_manga = ? AND FK\_user =?");

// Create a PreparedStatement object

ps.setInt(1, newFkUser);

ps.setInt(2, newFkManga);

ps.setInt(3, oldFkManga);

ps.setInt(4, oldFkUser);

ps.executeUpdate();

ok =true;

//On ferme le tout pour optimiser les performances.

} catch (SQLException ex) {

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

} //On repasse les variables pour vérifier que tout est bien fermé.

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 (ps != null) {

ps.close();

ps = null;

}

} catch (SQLException ex) {

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

}

}

}

return ok;

}

public boolean deleteFavoris(int fkUser, int fkManga) {

boolean ok = false;

boolean opendb = dbConnect();

ResultSet rs = null;

if (opendb) {

PreparedStatement ps = null;

try {

//On essaie de créer une requête grâce à notre connexion.

// Create a SQL statement

ps = jdbcConnection.prepareStatement("DELETE FROM tr\_manga\_user WHERE FK\_manga = ? AND FK\_user = ?");

// Create a PreparedStatement object

ps.setInt(1, fkManga);

ps.setInt(2, fkUser);

ps.executeUpdate();

ok =true;

//On ferme le tout pour optimiser les performances.

} catch (SQLException ex) {

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

} //On repasse les variables pour vérifier que tout est bien fermé.

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 (ps != null) {

ps.close();

ps = null;

}

} catch (SQLException ex) {

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

}

}

}

return ok;

}

}

# Hébergement

# Installation du projet complet avec les 5 applications

# Outils, langages (versions, définitions, installations)

# Tests de fonctionnement du projet

## Service Rest 1

### GetUser

**Local**

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

Description générée automatiquement

**En ligne**

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

Description générée automatiquement

### Login

**Local**

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

Description générée automatiquement

**En ligne**

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

Description générée automatiquement

## Service Rest 2

### getManga

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

### ajoutManga

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

### modifyManga

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

### deleteManga

Une image contenant capture d’écran, texte, logiciel, Logiciel multimédia

Description générée automatiquement

### getFavoris

Une image contenant capture d’écran, texte, logiciel, Logiciel multimédia

Description générée automatiquement

### ajoutfavoris

Une image contenant capture d’écran, texte, logiciel, Logiciel multimédia

Description générée automatiquement

### modifyFavoris

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

### deleteFavoris

Une image contenant capture d’écran, texte, logiciel, Logiciel multimédia

Description générée automatiquement

# Auto-évaluations et conclusions