|  |  |
| --- | --- |
| 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() VIII

3 Conception à faire complétement avec EA IX

3.1.1 Class Diagram complet avec les explications de chaque application ¨ IX

4 Bases de données X

4.1 Utilisateur X

4.2 Manga XI

4.3 Tr\_manga\_user XI

5 Implémentation des applications client Ap1 et client Ap2 XI

5.1 Application client 1 XII

5.1.1 Index.html XII

5.1.2 indexCtrl.js XIII

5.1.3 servicesHttp.js XVI

5.2 Application client 2 XX

5.2.1 Index.html XX

5.2.2 indexCtrl.js XXI

5.2.3 ServicesHttp.js XXV

6 Implémentation de l'aplication API Gateway XXIX

6.1 Partie Servlet sur serveur Tomcat commun XXIX

7 Implémentation des aplications Service Rest1 et Service Rest2 XXXV

7.1 Service Rest 1 XXXV

7.1.1 Rest XXXV

7.1.2 Wrk XXXVI

7.2 Service Rest 2 XXXIX

7.2.1 Rest XXXIX

7.2.2 Wrk XLI

7.2.3 Manga XLVIII

8 Hébergement XLIX

9 Outils, langages (versions, définitions, installations) XLIX

10 Tests de fonctionnement du projet XLIX

10.1 Service Rest 1 L

10.1.1 GetUser L

10.1.2 Login LI

10.2 Service Rest 2 LII

10.2.1 getManga LII

10.2.2 ajoutManga LIII

10.2.3 modifyManga LIV

10.2.4 deleteManga LV

10.2.5 getFavoris LVI

10.2.6 ajoutfavoris LVII

10.2.7 modifyFavoris LVIII

10.2.8 deleteFavoris LIX

11 Auto-évaluations et conclusions LIX

# 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()

Une image contenant texte, diagramme, capture d’écran, Parallèle

Description générée automatiquement

# Conception à faire complétement avec EA

### Class Diagram complet avec les explications de chaque application ¨

#### REST 1

Cette application va servir à gérer les mangas (ajouter, modifier et supprimer)

Une image contenant texte, capture d’écran, diagramme, Rectangle

Description générée automatiquement

#### REST 2

Cette application va principalement servir à gérer les favoris personnels des utilisateurs par exemple l’utilisateur Raphael vas avoir le tome 1,2,3,4 et 100,101,102,103 de one Piece et l’utilisateur Malcolm vas avoir dans ses favoris les tomes de 1 à 10. Et chacun des utilisateurs vas voir uniquement ses favoris.

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

#### API gateway

L’api vas servir à faire la liaison entre les services REST et les clients qui feront les requêtes.

Une image contenant texte, capture d’écran, logiciel, Rectangle

Description générée automatiquement

# Bases de données

## Utilisateur

Une image contenant texte

Description générée automatiquement

## Manga

Une image contenant texte, logiciel, nombre, Icône d’ordinateur

Description générée automatiquement

## Tr\_manga\_user

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

# Implémentation des applications client Ap1 et client Ap2

## Application client 1

### Index.html

<!doctype html>

<html lang="fr">

<head>

    <title>Grand Liste</title>

    <meta charset="UTF-8">

    <link rel="stylesheet" type="text/css" href="css/main.css" />

    <script type="text/javascript" src="js/helpers/jquery-1.10.2.min.js"></script>

    <script type="text/javascript" src="js/ctrl/indexCtrl.js"></script>

    <script type="text/javascript" src="js/services/servicesHttp.js"></script>

</head>

<body>

    <div class="container">

        <div class="titre">

            <h1>Grand Liste</h1>

        </div>

        <div class="connexion">

            <div class="loginLabel">

                <label class="labelLogin">Login</label>

            </div>

            <div class="passwordLabel">Password</div>

            <div class="login">

                <input name="username" id="username" />

            </div>

            <div class="password">

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

            </div>

            <div id="coDeco" class="connecter">

                <button class = "button" id="btnConnecter" type="submit" onclick="callLogin()">Connexion</button>

                <button class = "button" id="btnDeconnecter" type="submit" onclick="calldisconnect()">Deconnexion</button>

            </div>

            <div class="favoris">

                <button id="btnFavoris">Affichage des tomes obtenus</button>

            </div>

        </div>

    </div>

    <div id="btnGestion" class="btnGestion">

        <button id="addMangaBtn" onclick="infoForAdd()">Ajouter un Manga</button>

        <button id="modifyMangaBtn" onclick="infoForModify()">Modifier un Manga</button>

        <button id="deleteMangaBtn" onclick="infoForDelete()">Suprimer un Manga</button>

    </div>

    <div id="resultBtnPressed"></div>

</body>

</html>

### indexCtrl.js

/\*

 \* IndexCtrl de mon client

 \*

 \* @author Malcolm Gfeller

 \* @version 1.0 / 22.06.2023

 \*/

$().ready(function () {

    checkLogin(checkSuccess, servError);

});

function callLogin() {

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

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

    login(username, password, loginSuccess, servError);

    $("#username").val("");

    $("#password").val("");

}

function loginSuccess(data) {

    if (data.result === true) {

        alert("Connexion réussie");

        $("#btnGestion").show();

    } else {

        alert("Nom d'utilisateur ou mot de passe invalide");

    }

}

function checkSuccess(data) {

    if (data.trim() === "{\"result\": false}") {

        $("#btnGestion").hide();

    }

}

function infoForAdd() {

    document.getElementById("resultBtnPressed").innerHTML = "";

    document.getElementById("resultBtnPressed").innerHTML +=

        "<div>" +

        "<div>" +

        "<label>nomDuManga</label>" +

        "<input name='nomDuManga' id='nomDuManga'/>" +

        "<label>nomDuTome</label>" +

        "<input name='nomDuTome' id='nomDuTome'/>" +

        "<label>numDuTome</label>" +

        "<input name='numDuTome' id='numDuTome'/>" +

        "<label>image</label>" +

        "<input name='image' id='image'/>" +

        "</div>" +

        "<button onclick='callAddManga()'>Ajouter</button>" +

        "</div>";

}

function infoForModify() {

    document.getElementById("resultBtnPressed").innerHTML = "";

    document.getElementById("resultBtnPressed").innerHTML +=

        "<div>" +

        "<label>PK</label>" +

        "<input name='PK' id='PK'/>" +

        "<label>nomDuManga</label>" +

        "<input name='nomDuManga' id='nomDuManga'/>" +

        "<label>nomDuTome</label>" +

        "<input name='nomDuTome' id='nomDuTome'/>" +

        "<label>numDuTome</label>" +

        "<input name='numDuTome' id='numDuTome'/>" +

        "<label>image</label>" +

        "<input name='image' id='image'/>" +

        "</div>";

}

function infoForDelete() {

    document.getElementById("resultBtnPressed").innerHTML = "";

    document.getElementById("resultBtnPressed").innerHTML +=

        "<div>" +

        "<label>PK</label>" +

        "<input name='PK' id='PK'/>" +

        "</div>";

}

/\*\*

 \* Adds an article with the filled fields. If any of the field isn't filled, we show an alert.

 \*/

function callAddManga() {

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

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

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

    var image = $("#image").val();

    if (nomDuManga === "" || image === "" || nomDuTome === "" || numDuTome === "") {

        alert("Please fill all the fields");

        return;

    } else {

        addManga(nomDuManga, nomDuTome, numDuTome, image, addSuccess, servError);

    }

}

function addSuccess(data) {

    console.log(data);

    if(data.trim() === "Manga ajoute"){

        alert("Votre manga a été ajouté");

        $("#nomDuManga").val("");

        $("#nomDuTome").val("");

        $("#numDuTome").val("");

        $("#image").val("");

    } else {

        alert("Les informations entrées sont invalides");

    }

}

/\*\*

 \* Adds an article with the filled fields. If any of the field isn't filled, we show an alert.

 \*/

function callModifyManga() {

    var PK = $("#PK").val();

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

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

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

    var image = $("#image").val();

    if (PK === "" || nomDuManga === "" || image === "" || nomDuTome === "" || numDuTome === "") {

        alert("Please fill all the fields");

        return;

    } else {

        modifyManga(PK, nomDuManga, nomDuTome, numDuTome, image, modifySuccess, servError);

    }

}

function modifySuccess() {

    alert("Votre manga a été modifié");

    $("#PK").val("");

    $("#nomDuManga").val("");

    $("#nomDuTome").val("");

    $("#numDuTome").val("");

    $("#image").val("");

}

/\*\*

 \* Adds an article with the filled fields. If any of the field isn't filled, we show an alert.

 \*/

function callDeleteManga() {

    var PK = $("#PK").val();

    if (PK === "") {

        alert("Please fill the field");

        return;

    } else {

        deleteManga(PK, deleteSuccess, servError);

    }

}

function deleteSuccess() {

    alert("Votre manga a été supprimé");

    $("#PK").val("");

}

function calldisconnect() {

    disconnect(disconnectSuccess, servError);

}

function disconnectSuccess() {

    alert("Déconnexion réussie");

    location.reload();

}

function servError() {

    alert("Erreur de connexion au serveur");

}

### servicesHttp.js

/\*

 \* Couche de services HTTP .

 \*

 \* @author Malcolm Gfeller

 \* @version 1.0 / 22.06.2023

 \*/

var URL = "https://gfellerm01.emf-informatique.ch/javaAPI-Gateway/GatewayServlet?action=";

/\*\*

 \* Login function, if the login is successful, we call loginSuccessCallBack, else we call loginErrorCallBack.

 \* @param username login name

 \* @param password user password

 \* @param successCallback success callback

 \* @param errorCallBack error callback

 \*/

function login(username, password, successCallback, errorCallBack) {

  $.ajax({

    type: "POST",

    dataType: "json",

    url: URL + "apiLogin",

    data: "username=" + username + "&password=" + password,

    xhrFields: {

      withCredentials: true,

    },

    async: false,

    crossDomain: true,

    success: successCallback,

    error: errorCallBack

  });

}

/\*\*

 \* Adds a manga

 \* @param NomDuManga name of the manga

 \* @param Image image of the tome

 \* @param NumeroDuTome num of the tome

 \* @param NomDuTome name of the tome

 \* @param successCallback success callback

 \* @param errorCallBack error callback

 \*/

function addManga(NomDuManga, NomDuTome, NumeroDuTome, Image, successCallback, errorCallback) {

  $.ajax({

    type: "POST",

    datatype: 'xml',

    url: URL + "apiAjoutManga",

    data: "nomDuManga=" + NomDuManga + "&nomDuTome=" + NomDuTome + "&numDuTome=" + NumeroDuTome + "&image=" + Image,

    xhrFields: {

      withCredentials: true

    },

    async: false,

    crossDomain: true,

    success: function(){},

    error: function(){}

  });

}

/\*\*

 \* Modifies a manga

 \* @param NomDuManga name of the manga

 \* @param Image image of the tome

 \* @param NumeroDuTome num of the tome

 \* @param NomDuTome name of the tome

 \* @param successCallback success callback

 \* @param errorCallBack error callback

 \*/

function modifyManga(PK, NomDuManga, NomDuTome, NumeroDuTome, Image, successCallback, errorCallback) {

  $.ajax({

    type: "PUT",

    datatype: 'json',

    url: URL + "apiModifyManga",

    data: "Pk=" + PK + "&nomDuManga=" + NomDuManga + "&nomDuTome=" + NomDuTome + "&numDuTome=" + NumeroDuTome + "&image=" + Image,

    xhrFields: {

      withCredentials: true

    },

    async: false,

    crossDomain: true,

    success: successCallback,

    error: errorCallback

  });

}

/\*\*

 \* Deletes a manga

 \* @param NomDuManga name of the manga

 \* @param Image image of the tome

 \* @param NumeroDuTome num of the tome

 \* @param NomDuTome name of the tome

 \* @param successCallback success callback

 \* @param errorCallBack error callback

 \*/

function deleteManga(PK, successCallback, errorCallback) {

  $.ajax({

    type: "DELETE",

    datatype: 'json',

    url: URL + "apiDeleteManga",

    data: "Pk=" + PK,

    xhrFields: {

      withCredentials: true

    },

    async: false,

    crossDomain: true,

    success: successCallback,

    error: errorCallback

  });

}

/\*\*

     \* Make an ajax request to check if the user is connected.

     \* @param successCallback The function called when the request is successful

     \*/

function checkLogin(successCallback, errorCallback) {

  $.ajax({

    type: "POST",

    dataType: "text",

    url: URL + "apiCheckLogin",

    xhrFields: {

      withCredentials: true

    },

    async: false,

    crossDomain: true,

    success: successCallback,

    error: errorCallback

  });

}

/\*\*

 \* disconnects the user

 \* @param successCallback success callback

 \* @param errorCallBack error callback

 \*/

function disconnect(successCallback, errorCallback) {

  $.ajax({

    type: "POST",

    datatype: 'json',

    url: URL + "apiDisconnect",

    xhrFields: {

      withCredentials: true

    },

    async: false,

    crossDomain: true,

    success: successCallback,

    error: errorCallback

  });

}

## Application client 2

### Index.html

<!doctype html>

<html lang="fr">

<head>

    <title>Grand List</title>

    <meta charset="UTF-8">

    <link rel="stylesheet" type="text/css" href="stylesheets/style.css" />

    <script type="text/javascript" src="helpers/jquery-1.10.2.min.js"></script>

    <script type="text/javascript" src="ctrl/indexCtrl.js"></script>

</head>

<body>

    <div class="container">

        <div class="titre">

            <h1>Grand List</h1>

        </div>

        <div class="connexion">

            <div class="loginLabel">

                <label class="labelLogin">Login</label>

            </div>

            <div class="passwordLabel">Password</div>

            <div class="login">

                <input name="username" id="username" />

            </div>

            <div class="password">

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

            </div>

            <div id="coDeco" class="connecter">

                <button class = "button" id="btnConnecter" type="submit" onclick="callLogin()">Connexion</button>

                <button class = "button" id="btnDeconnecter" type="submit" onclick="calldisconnect()">Deconnexion</button>

            </div>

            <div class="favori">

                <button id="favori">Affichage des favoris</button>

            </div>

        </div>

    </div>

        <div id="listeManga" class="listeManga">

        </div>

    </div>

</body>

</html>

### indexCtrl.js

/\*

 \* Contrôleur de la vue "index.html"

 \*

 \* @author de Sousa Raphael

 \* @version 1.0 / 22.06.2023

 \*/

let user = null;

/\*\*

 \* Méthode appelée lors du retour avec succès de la liste des mangas

 \* @param {type} data

 \* @param {type} text

 \* @param {type} jqXHR

 \* @returns {undefined}

 \*/

function chargerMangaSuccess(data, text, jqXHR) {

  var jsonArray = JSON.parse(data);

  var txt = '';

  for (var i = 0; i < jsonArray.length; i++) {

    var manga = new Manga();

    manga.setNomDuManga(jsonArray[i].nomDuManga);

    manga.setPk(jsonArray[i].pk\_manga);

    manga.setNomDuTome(jsonArray[i].nomDuTome);

    manga.setNumeroDuTome(jsonArray[i].numeroDuTome);

    manga.setImage(jsonArray[i].image);

    txt = txt + "<div class=manga><table><tbody><tr><td>" + "<img id=myimage src=../../serveur/images/" + manga.getImg() + "></img>" + "</td></tr><tr><td>" + manga.getNomDuTome() + " - " + manga.getNomDuManga() + ", <br> Tome " + manga.getNumeroDuTome() + "<br></td></tr><tr><td><button class=butFavori id=manga" + manga.getPk() + " >Ajouter à la collection</button>" + "</td></tr></tbody></table></div>";

  }

  document.getElementById("listeManga").innerHTML = txt;

  var favori = $('#favori');

  document.getElementById('favori').innerHTML = "Favori";

  favori.off('click');

  favori.click(function (event) { getFavori(user, chargerFavori, chargerErrorManga) });

}

function chargerFavori(data, text, jqXHR) {

  var txt = '';

  $(data).find("manga").each(function () {

    var manga = new Manga();

    manga.setNomDuManga($(this).find("nomManga").text());

    manga.setPk($(this).find("pk\_manga").text());

    manga.setNomDuTome($(this).find("nomTome").text());

    manga.setNumeroDuTome($(this).find("numTome").text());

    manga.setImage($(this).find("image").text());

    txt = txt + "<div class=manga><table><tbody><tr><td>" + "<img id=myimage src=../serveur/images/" + manga.getImg() + "></img>" + "</td></tr><tr><td>" + manga.getNomDuTome() + " - " + manga.getNomDuManga() + ", <br> Tome " + manga.getNumeroDuTome() + "<br></td></tr><tr><td><button class=butFavori id=manga" + manga.getPk() + " >Ajouter à la collection</button>" + "</td></tr></tbody></table></div>";

  });

  document.getElementById("listeManga").innerHTML = txt;

  var favori = $('#favori');

  document.getElementById('favori').innerHTML = "Retour à la vue global";

  favori.off('click');

  favori.click(function (event) { getManga(chargerMangaSuccess, chargerErrorManga) });

}

function callLogin() {

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

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

  login(username, password, loginSuccess, loginError);

  $("#username").val("");

  $("#password").val("");

}

function loginSuccess(data) {

  if (data.result === true) {

    alert("Connexion réussie");

    applyCSSLogin();

  } else {

    alert("Nom d'utilisateur ou mot de passe invalide");

  }

}

function checkSuccess(data) {

  if (data.trim() === "{\"result\": false}") {

  }

}

function loginError() {

  alert("Erreur de connexion");

}

function calldisconnect() {

  disconnect(disconnectSuccess, disconnectError);

}

function disconnectSuccess() {

  alert("Déconnexion réussie");

  location.reload();

}

function disconnectError() {

  alert("Erreur de déconnexion");

}

function chargerError(request, status, error) {

  alert("L'identifiant ou le mot de passe est faux");

}

function chargerErrorManga(request, status, error) {

  alert("Erreur lors de l'ajoute des mangas dans les favoris");

}

/\*\*

 \* vas afficher les bouttons qui sont cacher lors du login

 \*/

function applyCSSLogin() {

  $("#favori").css("visibility", "visible");

  $(".butFavori").css("visibility", "visible");

  var deconnexion = $('#connecter');

  document.getElementById('connecter').innerHTML = "Déconnexion";

  deconnexion.off('click');

  deconnexion.click(function (event) { disconnect(applyCSSDisconnect, chargerError); });

  var ajouter = $('.butFavori');

  ajouter.click(function (event) {

    pk = this.id.slice(5);

    currentUser(setUser, chargerError);

    addManga(pk, user, mangaAjoute, chargerErrorManga);

  });

  ajouter.each(function () {

    $pk = this.id.slice(3);

    mangaDelete($pk);

  })

}

function setUser(data) {

  user = data;

}

function verifyLogged(data) {

  if (data) {

    applyCSSLogin();

  }

}

/\*\*

 \* Méthode appelée en cas d'erreur lors de la lecture des mangas

 \* @param {type} request

 \* @param {type} status

 \* @param {type} error

 \* @returns {undefined}

 \*/

function chargerMangaError(request, status, error) {

  alert("Erreur lors de la lecture des mangas: " + error);

}

/\*\*

 \* Méthode appelée en cas d'erreur lors de la lecture des skieurs

 \* @param {type} request

 \* @param {type} status

 \* @param {type} error

 \* @returns {undefined}

 \*/

function chargerUtilisateurError(request, status, error) {

  alert("Erreur lors de la lecture des utilisateur: " + error);

}

/\*\*

 \* Méthode "start" appelée après le chargement complet de la page

 \* @param {type} data

 \*/

$(document).ready(function () {

  var cmbManga = $("#cmbManga");

  var manga = '';

  var connecter = $('#connecter');

  var favori = $('.favori')

  $.getScript("helpers/dateHelper.js", function () {

    console.log("dateHelper.js chargé !");

  });

  $.getScript("beans/manga.js", function () {

    console.log("manga.js chargé !");

  });

  $.getScript("beans/utilisateur.js", function () {

    console.log("utilisateur.js chargé !");

  });

  $.getScript("services/servicesHttp.js", function () {

    console.log("servicesHttp.js chargé !");

    getManga(chargerMangaSuccess, chargerMangaError);

    checkLogin(checkSuccess, loginError);

  });

  cmbManga.change(function (event) {

    manga = this.options[this.selectedIndex].value;

    chargerUtilisateur(JSON.parse(manga).pk, chargerUtilisateurSuccess, chargerUtilisateurError);

  });

  favori.click(function (event) {

    getFavori(user, chargerFavori, chargerError);

  })

});

### ServicesHttp.js

/\*

 \* Couche de services HTTP .

 \*

 \* @author de Sousa Raphael

 \* @version 1.0 / 22.06.2023

 \*/

var URL = "https://gfellerm01.emf-informatique.ch/javaAPI-Gateway/GatewayServlet?action=";

 /\*\*

 \* Login function, if the login is successful, we call loginSuccessCallBack, else we call loginErrorCallBack.

 \* @param username login name

 \* @param password user password

 \* @param successCallback success callback

 \* @param errorCallBack error callback

 \*/

function login(username, password, successCallback, errorCallBack) {

  $.ajax({

    type: "POST",

    dataType: "json",

    url: URL + "apiLogin",

    data: "username=" + username + "&password=" + password,

    xhrFields: {

      withCredentials: true,

    },

    async: false,

    crossDomain: true,

    success: successCallback,

    error: errorCallBack

  });

}

/\*\*

\* Make an ajax request to check if the user is connected.

\* @param successCallback The function called when the request is successful

\*/

function checkLogin(successCallback, errorCallback) {

  $.ajax({

    type: "POST",

    dataType: "text",

    url: URL + "apiCheckLogin",

    xhrFields: {

      withCredentials: true

    },

    async: false,

    crossDomain: true,

    success: successCallback,

    error: errorCallback

  });

}

/\*\*

 \* disconnects the user

 \* @param successCallback success callback

 \* @param errorCallBack error callback

 \*/

function disconnect(successCallback, errorCallback) {

  $.ajax({

    type: "POST",

    datatype: 'json',

    url: URL + "apiDisconnect",

    xhrFields: {

      withCredentials: true

    },

    async: false,

    crossDomain: true,

    success: successCallback,

    error: errorCallback

  });

}

  /\*\*

   \* Get all the manga

   \* @param successCallback success callback

   \* @param errorCallBack error callback

   \*/

  function getManga(successCallback, errorCallBack) {

    $.ajax({

      type: "GET",

      dataType: "json",

      url: URL +"apiGetManga",

      xhrFields: {

        withCredentials: true

      },

      async: false,

      crossDomain: true,

      success: successCallback,

      error: errorCallBack

    });

  }

  /\*\*

   \* Get all the favoris

   \* @param successCallback success callback

   \* @param errorCallBack error callback

   \*/

  function getFavoris(successCallback, errorCallBack) {

    $.ajax({

      type: "GET",

      dataType: "json",

      data: data,

      url: URL +"apiGetFavoris",

      xhrFields: {

        withCredentials: true

      },

      async: false,

      crossDomain: true,

      success: successCallback,

      error: errorCallBack

    });

  }

  /\*\*

   \* adds an favoris

   \* @param FK\_manga fk of the manga

   \* @param FK\_user fk of the user

   \* @param successCallback success callback

   \* @param errorCallBack error callback

   \*/

  function addFavoris(FK\_manga, FK\_user, successCallback, errorCallback) {

    $.ajax({

      type: "POST",

      datatype: 'json',

      data: data +"fkManga=" + FK\_manga + "&fkUser=" + FK\_user,

      url: URL +"apiGetFavoris",

      xhrFields: {

        withCredentials: true

      },

      async: false,

      crossDomain: true,

      success: successCallback,

      error: errorCallback

    });

  }

  /\*\*

   \* adds an favoris

   \* @param FK\_manga fk of the manga

   \* @param FK\_user fk of the user

   \* @param successCallback success callback

   \* @param errorCallBack error callback

   \*/

  function delFavoris(FK\_manga, FK\_user, successCallback, errorCallback) {

    $.ajax({

      type: "DELETE",

      datatype: 'json',

      data: data + "fkManga=" + FK\_manga + "&fkUser=" + FK\_user,

      url: URL +"apiDeleteFavoris",

      xhrFields: {

        withCredentials: true

      },

      async: false,

      crossDomain: true,

      success: successCallback,

      error: errorCallback

    });

  }

  /\*\*

   \* disconnects the user

   \* @param successCallback success callback

   \* @param errorCallBack error callback

   \*/

  function disconnect(successCallback, errorCallBack) {

    $.ajax({

      type: "POST",

      dataType: "json",

      data: "action=logout",

      url: URL,

      xhrFields: {

        withCredentials: true

      },

      async: false,

      crossDomain: true,

      success: successCallback,

      error: errorCallBack

    });

  }

# Implémentation de l'aplication API Gateway

## Partie Servlet sur serveur Tomcat commun

/\*

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

\* Click nbfs://nbhost/SystemFileSystem/Templates/JSP\_Servlet/Servlet.java to edit this template

\*/

package controllers;

import beans.User;

import beans.ParameterStringBuilder;

import beans.SimpleResponse;

import com.google.gson.Gson;

import com.google.gson.GsonBuilder;

import java.io.BufferedReader;

import java.io.DataOutputStream;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.HttpURLConnection;

import java.net.URL;

import java.net.URLEncoder;

import java.util.HashMap;

/\*\*

\*

\* @author GfellerM01

\*/

@WebServlet(name = "GatewayServlet", urlPatterns = {"/GatewayServlet"})

public class GatewayServlet extends HttpServlet {

// <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 {

response.setHeader("Access-Control-Allow-Origin", request.getHeader("Origin")); // pour le problème de CORS Policy

response.setHeader("Access-Control-Allow-Credentials", "true");

if (session == null) {

session = request.getSession();

}

PrintWriter out = response.getWriter();

String result = "";

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

switch (action) {

case "apiGetManga":

result = apiGetManga(request, response);

out.println(result);

break;

case "apiGetFavoris":

if (session.getAttribute("login") == null) {

response.sendError(401, "You are not logged in");

break;

}

result = apiGetFavoris(request, response, session.getAttribute("login").toString());

out.println(result);

break;

}

}

public String apiGetManga(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String code = sendGet("http://desousar.emf-informatique.ch/javaRestFull2/webresources/manga/getManga");

return code;

}

public String apiGetFavoris(HttpServletRequest request, HttpServletResponse response, String login) throws ServletException, IOException {

String code = sendGet("http://desousar.emf-informatique.ch/javaRestFull2/webresources/manga/getFavoris?u=" + login);

return code;

}

private String sendGet(String url) {

try {

HttpURLConnection conn = (HttpURLConnection) new URL(url).openConnection();

conn.setRequestMethod("GET");

BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));

String inputLine;

StringBuilder response = new StringBuilder();

while ((inputLine = in.readLine()) != null) {

response.append(inputLine);

}

in.close();

return response.toString();

} catch (Exception e) {

e.printStackTrace();

}

return "";

}

private String toUrlFormat(String login) {

return login.replace(" ", " ");

}

/\*\*

\* 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 {

response.setHeader("Access-Control-Allow-Origin", request.getHeader("Origin")); // pour le problème de CORS Policy

response.setHeader("Access-Control-Allow-Credentials", "true");

if (session == null) {

session = request.getSession();

}

response.setContentType("application/json;charset=UTF-8");

PrintWriter out = response.getWriter();

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

String result = "{\"result\": false}";

switch (action) {

case "apiLogin":

result = apiLogin(request, response, session);

break;

case "apiCheckLogin":

if (session.getAttribute("login") != null) {

result = "User : " + session.getAttribute("login");

}

break;

case "apiAjoutManga":

if (session.getAttribute("login") == null) {

response.sendError(401, "You are not logged in");

break;

}

result = apiAjoutManga(request, response);

break;

case "apiAjoutFavoris":

if (session.getAttribute("login") == null) {

response.sendError(401, "You are not logged in");

break;

}

result = apiAjoutFavoris(request, response);

break;

case "apiDisconnect":

session.invalidate();

result = "{\"result\": true}";

session = null;

break;

}

out.println(result);

}

public String apiLogin(HttpServletRequest request, HttpServletResponse response, HttpSession session) throws ServletException, IOException {

String valeurRetour = "";

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

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

String code = sendCUD("http://gfellerm01.emf-informatique.ch/javaServiceRest1/webresources/user/login", "POST", new HashMap<String, String>() {

{

put("username", username);

put("password", password);

}

});

SimpleResponse status = new Gson().fromJson(code, SimpleResponse.class);

if ("success".equals(status.getStatus())) {

// Login succeeded, do something here

session.setAttribute("login", username);

response.setHeader("Set-Cookie", "JSESSIONID=" + session.getId() + "; HttpOnly; SameSite=none; secure");

valeurRetour = "{\"result\": true}";

} else {

// Error occurred, do something here

valeurRetour = "{\\\"result\\\": false}";

}

return valeurRetour;

}

public String apiAjoutManga(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String valeurRetour = "";

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

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

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

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

String code = sendCUD("http://desousar.emf-informatique.ch/javaRestFull2/webresources/manga/ajoutManga", "POST", new HashMap<String, String>() {

{

put("nomDuManga", nomDuManga);

put("nomDuTome", nomDuTome);

put("numDuTome", numDuTome);

put("image", image);

}

});

boolean status = Boolean.parseBoolean(code.trim());

if (status) {

valeurRetour = "<return>Manga ajoute</return>";

} else {

valeurRetour = "<return>Erreur de ajout, veuillez remplir comme demande</return>";

}

return valeurRetour;

}

public String apiAjoutFavoris(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String valeurRetour = "";

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

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

String code = sendCUD("http://desousar.emf-informatique.ch/javaRestFull2/webresources/manga/ajoutFavoris", "POST", new HashMap<String, String>() {

{

put("fkUser", fkUser);

put("fkManga", fkManga);

}

});

boolean status = Boolean.parseBoolean(code.trim());

if (status) {

valeurRetour = "Ajoute en favoris";

} else {

valeurRetour = "Erreur de ajout, veuillez choisir des IDs valables";

}

return valeurRetour;

}

public String sendCUD(String url, String method, HashMap<String, String> data) {

String result = "false";

try {

HttpURLConnection conn = (HttpURLConnection) new URL(url).openConnection();

conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

conn.setRequestMethod(method);

conn.setDoOutput(true);

DataOutputStream out = new DataOutputStream(conn.getOutputStream());

out.writeBytes(ParameterStringBuilder.getParamsString(data));

out.flush();

out.close();

BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));

StringBuilder sb = new StringBuilder();

String line;

while ((line = br.readLine()) != null) {

sb.append(line + "\n");

}

br.close();

result = sb.toString();

} catch (Exception e) {

e.printStackTrace();

}

return result;

}

@Override

protected void doDelete(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setHeader("Access-Control-Allow-Origin", "\*"); // pour le problème de CORS Policy

if (session == null) {

session = request.getSession();

}

PrintWriter out = response.getWriter();

String result = "";

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

switch (action) {

case "apiDeleteManga":

System.out.println(request.toString());

result = apiDeleteManga(request, response);

out.println(result);

break;

case "apiDeleteFavoris":

result = apiDeleteFavoris(request, response);

out.println(result);

break;

}

}

public String apiDeleteManga(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String valeurRetour = "";

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

String code = sendCUD("http://desousar.emf-informatique.ch/javaRestFull2/webresources/manga/deleteManga", "DELETE", new HashMap<String, String>() {

{

put("Pk", Pk);

}

});

boolean status = Boolean.parseBoolean(code.trim());

if (status) {

valeurRetour = "Manga supprime avec succes";

} else {

valeurRetour = "Erreur de suppression";

}

return valeurRetour;

}

public String apiDeleteFavoris(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String valeurRetour = "";

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

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

String code = sendCUD("http://desousar.emf-informatique.ch/javaRestFull2/webresources/manga/deleteFavoris", "DELETE", new HashMap<String, String>() {

{

put("FKM", FKM);

put("FKU", FKU);

}

});

boolean status = Boolean.parseBoolean(code.trim());

if (status) {

valeurRetour = "Favoris supprime avec succes";

} else {

valeurRetour = "Erreur de suppression";

}

return valeurRetour;

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

HttpSession session;

}

# 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(String userId) {

Gson builder = new Gson();

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

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

// 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 beans.Manga;

import com.google.gson.Gson;

import com.google.gson.GsonBuilder;

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.List;

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 String lireManga() {

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

Statement stmt = null;

ResultSet rs = null;

if (dbConnect()) {

try {

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

String sql = "SELECT \* FROM t\_manga ORDER BY NomDuManga;";

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

while (rs.next()) {

Manga manga = new Manga();

manga.setPk\_manga(rs.getString("pk\_manga"));

manga.setNomDuManga(rs.getString("NomDuManga"));

manga.setNomDuTome(rs.getString("NomDuTome"));

manga.setNumeroDuTome(rs.getString("NumeroDuTome"));

manga.setImage(rs.getString("Image"));

resultat.add(manga);

}

}

rs.close();

rs = null;

stmt.close();

stmt = 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 (stmt != null) {

stmt.close();

stmt = null;

}

} catch (SQLException ex) {

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

}

}

}

Gson gson = new GsonBuilder().create();

String json = gson.toJson(resultat);

/\*json.replace("[","");

json.replace("]", "");\*/

return json;

}

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 String lireFavoris(String userId) {

ArrayList<Manga> 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 sqlUM = "SELECT \* FROM tr\_manga\_user";

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) {

while (rs.next()) {

Manga manga = new Manga();

manga.setPk\_manga(rs.getString("pk\_manga"));

manga.setNomDuManga(rs.getString("NomDuManga"));

manga.setNomDuTome(rs.getString("NomDuTome"));

manga.setNumeroDuTome(rs.getString("NumeroDuTome"));

manga.setImage(rs.getString("Image"));

resultat.add(manga);

}

}

//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);

} 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);

}

}

}

Gson gson = new GsonBuilder().create();

String json = gson.toJson(resultat);

return json;

}

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;

}

}

### Manga

/\*

\* 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 beans;

/\*\*

\*

\* @author desousar

\*/

public class Manga {

private String pk\_manga;

private String nomDuManga;

private String nomDuTome;

private String numeroDuTome;

public String getPk\_manga() {

return pk\_manga;

}

public void setPk\_manga(String pk\_manga) {

this.pk\_manga = pk\_manga;

}

public String getNumeroDuTome() {

return numeroDuTome;

}

public void setNumeroDuTome(String numeroDuTome) {

this.numeroDuTome = numeroDuTome;

}

public String getImage() {

return image;

}

public void setImage(String image) {

this.image = image;

}

private String image;

public Manga() {

}

public String getNomDuManga() {

return nomDuManga;

}

public void setNomDuManga(String nomManga) {

this.nomDuManga = nomManga;

}

public String getNomDuTome() {

return nomDuTome;

}

public void setNomDuTome(String nomTome) {

this.nomDuTome = nomTome;

}

}

# Hébergement

Nous avons hébergé le code de notre projet sur tizoo qui ont pour lien :

desousar.emf-informatique.ch et gfellerm01.emf-informatique.ch

Veuillez se référer au fichier Readme de github pour les liens des clients.

# Outils, langages (versions, définitions, installations)

Nous avons utilisé netbeans pour l’implémentation des REST et de l’API et nous avons utilisé Visual studio code pour les clients.

# 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

#### Local

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

Description générée automatiquement

#### En ligne

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

Description générée automatiquement

### ajoutManga

#### Local

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

Description générée automatiquement

#### En ligne

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

Description générée automatiquement

### modifyManga

#### Local

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

Description générée automatiquement

#### En ligne

Une image contenant logiciel, Logiciel multimédia, Logiciel de graphisme, capture d’écran

Description générée automatiquement

### deleteManga

#### Local

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

Description générée automatiquement

#### En ligne

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

Description générée automatiquement

### getFavoris

#### Local

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

Description générée automatiquement

#### En ligne

Une image contenant logiciel, Logiciel multimédia, texte, Logiciel de graphisme

Description générée automatiquement

### ajoutfavoris

#### Local

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

Description générée automatiquement

#### En ligne

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

Description générée automatiquement

### modifyFavoris

#### Local

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

Description générée automatiquement

#### En ligne

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

Description générée automatiquement

### deleteFavoris

#### Local

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

Description générée automatiquement

#### En ligne

Une image contenant logiciel, Logiciel multimédia, capture d’écran, Logiciel de graphisme

Description générée automatiquement

# Auto-évaluations et conclusions

Raphael de Sousa

Durant ce projet je ne me suis pas mis à fond depuis le début car je pensai que ça allait être facile car je me sentais un peu confiant dans le domaine du web. Mais je me suis vite rendu compte qu’il fallait tous faire pour le réussir en se donnant à fond. Malgré tous je suis content de mon travail et de mon projet. J’ai beaucoup aimé travailler avec une API et des REST même si ça été dur.

**Malcolm Gfeller**

J’ai bien aimé ce module, c’était très intéressant, mais malheureusement je n’ai pas réussi à bien m’activer et travailler à la maison pendant le module, donc j’ai eu un peu de retard sur le projet, vu que j’ai raté 1.5 jours de cours. Malgré ce retard, j’ai fait le mieux que je pouvais pour combler ce retard avec mon collègue.