|  |  |
| --- | --- |
| NexFlop 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  Leonetti Matte, Gambera Luca  Groupe 8, classe 300232  Module 133 du 24.03.2023 au 05.05.2023 |
|  |  |

**Table des matières**

1 Introduction II

2 Analyse II

2.1 Architecture globale II

2.2 Use case globale II

2.3 Use case serviceRest 1 IV

2.4 Use case serviceRest 2 IV

2.5 Use case API Gateway V

2.6 Use case AP client 1 VII

2.7 Use case AP client 2 VIII

2.8 Sequence System AP client 1 VIII

2.9 Sequence System AP client 2 IX

3 Conception XI

3.1 Diagramme de classe serviceRest 1 XI

3.2 Diagramme de classe serviceRest 2 XI

3.3 Diagramme de classe API Gateway XI

4 Base de données XI

4.1 Modèle WorkBench XI

4.1.1 DB XI

5 Hébergement XII

6 Installation du projet XIII

7 Outils, langages XIII

7.1 Implémentation des clients XIII

7.2 Outils et langages XV

8 Tests XVI

9 Auto-évaluation et conclusion XVIII

9.1 Conclusion de projet XVIII

9.1.1 Matteo XVIII

9.1.2 Luca XVIII

9.2 Ce que j’ai aimé XVIII

9.2.1 Matteo XVIII

9.2.2 Luca XIX

9.3 Ce que je n’ai pas aimé XIX

9.3.1 Matteo XIX

9.3.2 Luca XIX

# Introduction

Le but de notre projet est de créer une sorte de forum ou on peut mettre des commentaires sous différents films. Notre site se nomme nexFlop. Pour ce faire, nous allons créer une base de données qui nous permettras de savoir qui à mis quel commentaire sur le site. Nous allons lier notre site à une API qui nous permettras de retrouver la plupart des films connu. Le premier client pourra se connecter, se déconnecter, laisser des avis une fois connecter, lire tous les films et avis en étant déconnecté ou connecté. Le client 2 sera un administrateur, il pourra voir la liste de tous les commentaires avec le nom d’utilisateur qui correspond.

# Analyse

## Architecture globale

Voici l’architecture globale :

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

Description générée automatiquement

G08 Gestion des items

Le client Ap2 G08Admin

(Luca)

## Use case globale

Voici le use case globale, nous allons maintenant développer chaque partie de ce schéma en développant son fonctionnement :

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

Description générée automatiquement

**Explication des cas pour client AP1 :**

Visitor-> s’enregistre, il passe par l’API Gateway, et le serveurRest1 enregistre l’utilisateur

Visitor -> login, il se logue avec son compte et API gateway va checker son login et le logue si tout est correct et crée la session.

Visitor -> Lire film, Un visiteur non enregistrer vas pour voir voir la liste des films, les films sont fournis par l’API qui est gérer dans le service rest2.

Visitor -> Lire commentaires, il pourra lire les commentaires déjà inscrit mais pas en écrire étant donné qu’il n’est pas connecté, les commentaires sont fournis par la DB du serviceRest 2.

User -> Lire film, Un utilisateur enregistrer vas pour voir voir la liste des films, les films sont fournis par l’API qui est gérer dans le service rest2.

User-> Ajouter un commentaire, il va pouvoir ajouter un commentaire, cela va enregistrer dans la TR des avis dans la DB qui est géré par serviceRest 2

User->Lire les commentaires, il pourra lire sont commentaire et ceux des autres lié à un film et sont fournis par la DB du serviceRest 2

**Explication des cas pour client AP2 :**

Admin-> Voir tous les avis, l’admin va pouvoir voir tous les avis et le nom d’utilisateur qui y est lié, les commentaires sont fournis par la DB du serviceRest2.

## Use case serviceRest 1

Voici le use case du serviceRest 1 :

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

Description générée automatiquement

**Explications des cas :**

L’API Gateway->Ajouter utilisateur, va ajouter un utilisateur à la DB

L’API Gateway->vérifier login, va vérifier le login lors d’une connexion via la DB

L’API Gateway-> Ajouter Avis, va ajouter l’avis de l’utilisateur dans la DB

L’API Gateway-> Lire Tous les avis, va lire tous les avis stocker dans la DB

L’API Gateway-> Lire Avis, va lire les avis stocker dans la DB lié à un seul film.

## Use case serviceRest 2

Voici l’use case du serviceRest2 :

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

Description générée automatiquement

Explications des cas :

L’API Gateway->LireUnFilm, va lire un seul film lié à l’api.

L’API Gateway->Lire Films, va lire la totalité des film lié a l’api.

## Use case API Gateway

Voici le use case de l’API Gateway

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

Description générée automatiquement

Client Api1 -> Ajout d’user, va permettre d’ajouter un utilisateur via le serviceRest1

Client Api1 -> Ajouter avis, va ajouter un avis via le serviceRest1

Client Api1 ->Lire les Avis, va lire la totalité des avis via le serviceRest1

Client Api1 -> LireFilm, valire un film via le serviceRest2

Client Api1 -> Contrôler login, va vérifier le login via le serviceRest1

Client Api2 -> Lire les avis, va lire tous les avis via le serviceRest1

Client Api2 -> LireFilm, va lire un film via le serviceRest2

Client Api2 -> Contrôler login, va contrôler le login via le serviceRest 1

## Use case AP client 1

Voici le use case du Client AP1

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

Description générée automatiquement

User -> ajouter un avis, va pouvoir ajouter un avis via l’APIGATEWAY

Visitor-> inscrire, va pouvoir se créer un compte via l’APIGATEWAY

Visitor-> Connecter /déconnecter, va pouvoir se co/déco via l’APIGATEWAY

Visitor-> lire les films, va pouvoir lire tous les films via l’APIGATEWAY

Visitor-> lire avis va pouvoir les les avis d’un film via l’APIGATEWAY

## Use case AP client 2

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

Description générée automatiquement

Admin-> supprimer avis, va pouvoir supprimer un avis via l’apiGateway

Admin-> lire avis, va pouvoirs lire les avis via l’apiGateway

Admin-> lire les films, va pouvoir lire les films via l’apiGateway

Admin-> vo/déco, va pouvoir se connecter ou se déconnecter via l’apiGateway

## Sequence System AP client 1

Voici le diagramme de séquence du client AP 1

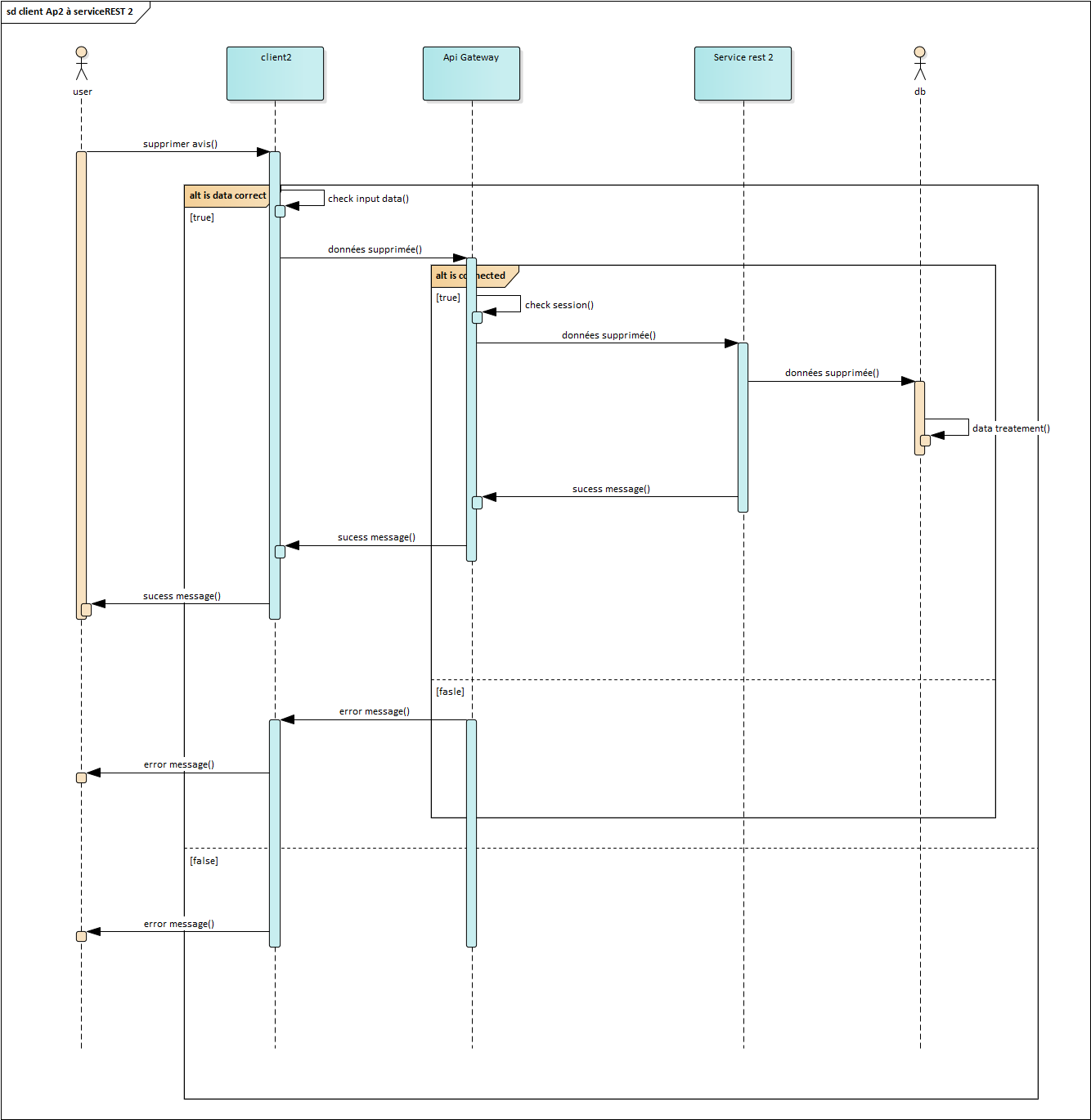
Une image contenant texte, diagramme, Parallèle, capture d’écran

Description générée automatiquement

L’utilisateur souhaite ajouter un avis alors il appuie sur le bouton d’ajout, le client AP1 va vérifier les données et si c’est bon, il va envoyer ces données à l’API gateway, si ce n’est pas bon on aura une erreur qui va s’afficher. Quand l’API Gateway reçoit ces données, il va vérifier que l’utilisateur qui les envoie à bien une session, si c’est le cas il va envoyer les données au serviceREST 2 qui lui va l’envoyer à la DB pour enregistrer son achat et il on va retourner un message avec un code 200 que tout s’est bien passé. S’il n’a pas de session, il aura une erreur avec un code 401 car il n’est pas connecté.

## Sequence System AP client 2

Voici le diagramme de séquence du client AP 2



L’administrateur souhaite supprimer un article, il appuie sur le bouton de suppression après avoir sélectionner et supprimé l’avis. On envoie l’instruction au Client AP 2, le Client AP2 reçoit et vérifie les données qui ont été envoyées et si c’est bon, il envoie ces données à l’API Gateway, si ce n’est pas bon, il aura une erreur qui va s’afficher. Quand l’API reçoit ces données, il va vérifier que l’administrateur qui les envoie à bien une session, si c’est le cas, il va envoyer les données au serviceREST 2 qui lui va l’envoyer à la DB et enregistrer les modifications. Puis nous aurons un message de succès avec un code 200 pour prévenir l’administrateur que tout s’est bien passé. Si l’administrateur n’avait pas de session, on va recevoir un message d’erreur avec un code 401 car l’administrateur n’était en fait pas connecté.

# Conception

## Diagramme de classe serviceRest 1

Voici le diagramme de classe du service Rest1. Il y a une classe pour les appels à la DB, le Wrk appelle le WrkDB et le Wrk est appelé par le contrôleur.

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

Description générée automatiquement

## Diagramme de classe serviceRest 2

Voici le diagramme de classe du service Rest2. Il y a une classe pour les appels à la DB, le Wrk appelle le WrkDB et le Wrk est appelé par le contrôleur.

Une image contenant texte, capture d’écran

Description générée automatiquement

## Diagramme de classe API Gateway

L’API Gateway va appeler des services des services rest.Le Wrk va être appelé par le Ctrl et le Ctrl par l’API Gateway. Depuis cette API Gateway.

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

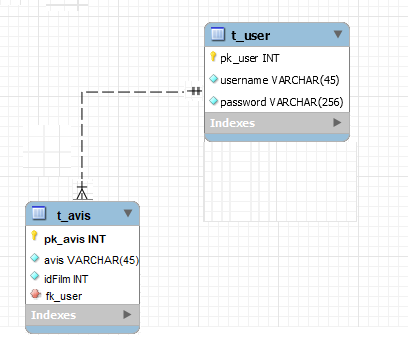
Description générée automatiquement

# Base de données

## Modèle WorkBench

### DB

La DB des utilisateurs, nous avons la t\_user qui est les utilisateurs et la t\_avis qui est les différents avis lié à un user.



# Hébergement

Pour l’hébergement, nous avons dû configurer notre compte Tizoo (qui est un cpanel). Pour commencer nous avons été dans nos outils, ea-tomcat, conf, et nous avons changé le xml sur serveur qui se nomme « server.xml ». Nous avons dû mettre la valeur de « unpackWARs » à true pour qu’il unzip et traite les .war qu’on lui upload.

Une image contenant texte, logiciel, Page web, Site web

Description générée automatiquement

Ensuite il a fallu aller sur la console du cpanel et y mettre cette commande :

export PATH=$(dirname $(readlink/usr/local/cpanel/3rdparty/bin/perl)):$PATH

ubic restart ea-tomcat85

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Description générée automatiquement[[1]](#footnote-1)

Ensuite, il n’y a plus qu’à se connecter avec FileZilla ou winSCP et mettre son projet en .war et le serveur fait le reste du travail.

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

Description générée automatiquement

# Installation du projet

Nous avons mis sur le site de matteo le client1 et le service rest1. Pour le client AP 1, il est sur le compte Tizoo de Matteo Leonetti : <https://leonettim.emf-informatique.ch/Service_Client/>

Nous avons mis sur le site de luca le client2 l’api gateway et le service rest2. Pour le client AP 2 : il est sur le compte Tizoo de Luca Gambera : <https://gamberal01.emf-informatique.ch/Service_Admin/>

# Outils, langages

## Implémentation des clients

Pour l’implémentation du Client AP 1 et AP 2, nous avons repris le code html et css de l’index du projet de Simon Losey du module 307 :

Une image contenant texte, collage, graphisme, capture d’écran

Description générée automatiquement

**Voici les différents vu du côté client. Pour commencer voici l’index :**

Une image contenant texte, collage, graphisme, capture d’écran

Description générée automatiquement

Ensuite voici la page qui permet de gérer la connexion/inscription :

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

Description générée automatiquementUne image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Voici la page qui permet de lire tous les avis lorsqu’on clique sur un film :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Voici la page qui permet d’ajouter un avis quand on est connecté et qu’on clique sur le bouton :

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

Description générée automatiquement

**Voici les différentes vues du second site. Voici l’index admin, on peut y voir tous les avis avec le nom de personnes qui l’on publiés :**

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

## Outils et langages

Dans notre projet nous avons utilisé comme la plupart des groupes pour la partie backend NetBeans car nous le trouvions plus pratique et moins complexe à l’utilisation. Tous les projets java utilise Maven qui se qui permet d’avoir les dépendances dans un fichier et au moment de compiler il va les télécharger. Dans notre cas nous utilisons deux artefacts, un pour un environnement exploded qui va simplement héberger les fichiers .class et l’autre qui va compiler le projet en fichie .war qui pourra être déployer sur le compte tizoo.

Une image contenant texte, capture d’écran, écran

Description générée automatiquement

Pour nos tests nous avons utilisé Postman qui permet de tester nos requêtes avant de les mettre en place :

Une image contenant texte, capture d’écran, ligne

Description générée automatiquement

Pour faire nos client web nous avons utilisé un IDE, WebStorm qui nous permet de tester nos clients avec un serveur web intégrer au logiciel.

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

Description générée automatiquement

# Tests

|  |  |  |
| --- | --- | --- |
| **No test** | **Description** | **Résultats attendus** |
| **1** | Visitor veut créer un compte sur le site web sur un navigateur (client). | Le visiteur peut créer son compte. |
|  | **Résultats obtenus** | **Conclusion** |
|  | Le visiteur a pu créer un compte | Une image contenant texte, clipart  Description générée automatiquement |

|  |  |  |
| --- | --- | --- |
| **No test** | **Description** | **Résultats attendus** |
| **2** | Un visiteur se logue à son via le site (client). | Le visiteur peut se loguer à son compte. |
|  | **Résultats obtenus** | **Conclusion** |
|  | Le visiteur a pu se connecter à son compte. | Une image contenant texte, clipart  Description générée automatiquement |

|  |  |  |
| --- | --- | --- |
| **No test** | **Description** | **Résultats attendus** |
| **3** | Un utilisateur met un avis (ajouter un avis) en appuyant sur le bouton « ajouter un avis » sur l’application web (client). | L’utilisateur peut laisser un avis |
|  | **Résultats obtenus** | **Conclusion** |
|  | L’utilisateur peut laisser un avis | Une image contenant texte, clipart  Description générée automatiquement |

|  |  |  |
| --- | --- | --- |
| **No test** | **Description** | **Résultats attendus** |
| **4** | Un visiteur essayer de mettre un avis via le bouton « laisser un avis » (client) | Le visiteur ne peut pas laisser d’avis |
|  | **Résultats obtenus** | **Conclusion** |
|  | Le visiteur ne peut pas laisser d’avis | Une image contenant texte, clipart  Description générée automatiquement |

|  |  |  |
| --- | --- | --- |
| **No test** | **Description** | **Résultats attendus** |
| 5 | L’utilisateur souhaite se déconnecter via l’application web (client). | L’utilisateur peut se déconnecter. |
|  | **Résultats obtenus** | **Conclusion** |
|  | L’utilisateur a pu se déconnecter. | Une image contenant texte, clipart  Description générée automatiquement |

|  |  |  |
| --- | --- | --- |
| **No test** | **Description** | **Résultats attendus** |
| 6 | Un administrateur regarde la liste des avis avec le nom d’utilisateur lié à ceux-ci | L’admin peut tout voir |
|  | **Résultats obtenus** | **Conclusion** |
|  | L’admin peut tout voir | Une image contenant texte, clipart  Description générée automatiquement |

# Auto-évaluation et conclusion

## Conclusion de projet

### Matteo

J’ai trouvé ce projet très intéressant, je pense qu’il aurait fallut encore une semaine pour finir a 100%le projet mais dans l’ensemble je suis satisfait du travail que j’ai fait. Le fait de mélanger différents langages pour faire un site est très intéressant et je pense que sa me sera très utile pour plus tard. Je pense que nous avons un peut perdu de temps au début se qui nous a stressé sur la fin. Malgré se contre temps nous avons atteint les objectifs attendus donc je suis satisfait.

### Luca

Je trouve que dans l’ensemble du projet nous avons été très productif. J’ai aimé se projet car il mélange technique et originalité. Je pense qu’il aurait fallu encore un peut plus de temps pour finir les détails. Je suis satisfait de se que j’ai fait. Je pense que nous aurions dû finir dès le début tous les schémas pour ne pas perdre de temps plus tard.

## Ce que j’ai aimé

### Matteo

J’ai beaucoup apprécié coder les clients car nous pouvons faire parler notre créativité. J’ai aussi apprécié faire les service rest car j’ai trouvé intéressant de mélanger la java des différents services avec l’api gateway et le js des clients.

### Luca

J’ai bien aimé programmer le gateway et le service rest. J’ai trouvé très intéressant de voir le tout fonctionner et de faire une communication entre les différentes parties du projet.

## Ce que je n’ai pas aimé

### Matteo

J’ai moins aimé le fait d’avoir un autant gros projet en si peut de temps, je pense qu’il aurait fallu encore une semaine. J’ai également moins aimé le fait de passer autant de temps sur els schéma alors que comme on est un petit groupe un schéma global aurait suffi.

### Luca

J’ai moins aimé la partie théorique et la partie sur les schémas qui à mon gout ont pris trop de temps. Je pense également qu’il aurait fallu quelque jour en plus pour finir les détails de notre site web. La partie sur les schémas prends énormément de temps ce pourquoi je pense que pour un projet de cette taille il aurait fallu plus de temps.

1. Image de M. Celato [↑](#footnote-ref-1)