|  |  |
| --- | --- |
| EMF-Shelf 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  Nathan Clapasson et Alexandre Haymoz  Groupe numéro et Classe  Module 133 du 23.03.2023 au 05.05.2023 |
|  |  |

**Table des matières**

1 Introduction et contexte du projet 3

2 Analyse à faire complètement avec EA 4

2.1 Use case avec les explications des cas pour les applications 4

2.1.1 Use case globale 4

2.1.2 Use case AP1 5

2.1.3 Use case AP2 6

2.1.4 Use case APIGateway 7

2.1.5 Use case REST1 8

2.1.6 Use case REST 2 9

2.2 Sequence System global entre les applications 10

2.2.1 Réserver un livre 10

2.2.2 Bannir un utilisateur 11

3 Conception à faire complétement avec EA 12

3.1 Class Diagram complet avec les explications de chaque application 12

3.1.1 Class Diagram APIGateway 12

3.1.2 Class Diagram REST 1 12

3.1.3 Class Diagram REST 2 12

4 Hébergement 12

5 Installation du projet complet avec les 5 applications 13

6 Outils, langages (versions, définitions, installations) 13

7 Tests de fonctionnement du projet 13

8 Auto-évaluations et conclusions 13

# Introduction et contexte du projet

Dans le cadre de ce module, nous devons réaliser un projet découpé en 5 parties : deux clients, une Gateway et deux serveurs REST. Nous avons donc décidé de créer un système de bibliothèque avec un client pour les utilisateurs, un client pour les admins, un REST pour gérer les livres et les réservations, un REST pour gérer les comptes utilisateurs et la Gateway qui fait le lien entre les clients et les serveurs.

Une image contenant diagramme

Description générée automatiquement

# Analyse à faire complètement avec EA

## Use case avec les explications des cas pour les applications

### Use case globale

Du côté client, nous avons trois rôles : visiteur qui peut se connecter, user qui peut faire des actions sur l’application client et admin qui peut faire des actions sur l’application administrateur.

Une image contenant diagramme

Description générée automatiquement

### Use case AP1

Dans l’application client, les visiteurs peuvent voir les livres, se connecter et créer un compte. Les utilisateurs connectés peuvent se déconnecter, réserver un livre et le rendre.

Une image contenant diagramme, schématique

Description générée automatiquement

### Use case AP2

Dans l’application administrateur, un visiteur peut uniquement se connecter. Un admin connecté peut gérer les livres et les réservations, bannir un utilisateur et se déconnecter.

Une image contenant diagramme

Description générée automatiquement

### Use case APIGateway

Les deux clients peuvent demander une connexion pour les utilisateurs et tuer la connexion. Le client utilisateur peut demander une création de compte, demander les livres et gérer les réservations de l’utilisateur. Le client admin peut gérer les livres et les réservations de tous les utilisateurs et bannir un utilisateur.

Une image contenant diagramme, schématique

Description générée automatiquement

### Use case REST1

L’API peut effectuer des CRUD (rechercher, créer, modifier et supprimer) pour les livres et les réservations sur la base de données.

Une image contenant diagramme

Description générée automatiquement

### Use case REST 2

L’API peut enregistrer un nouvel utilisateur, connecter un utilisateur et bannir un utilisateur.

Une image contenant diagramme

Description générée automatiquement

## Sequence System global entre les applications

### Réserver un livre

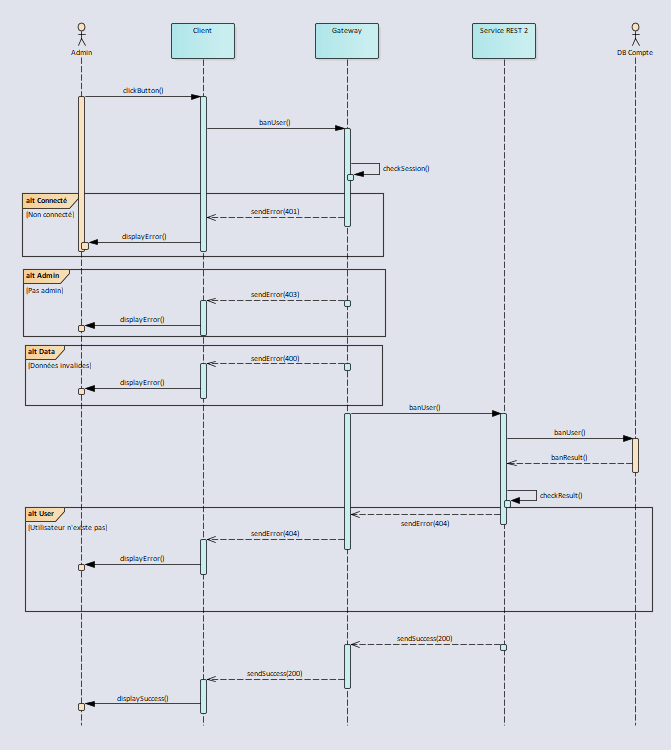
Lorsque l’utilisateur clique sur le bouton pour réserver un livre, le client va envoyer la requête au douanier qui va vérifier la session. Il va renvoyer des codes d’erreur si l’utilisateur n’est pas connecté (401) ou si des données sont manquantes (400). Il va par la suite envoyer la requête au REST 1 qui va vérifier si le livre est disponible. Si le livre n’est pas disponible, il va retourner une erreur 400. Pour finir, le service REST va créer la réservation et retourner un code 200 si tout va bien et 500 en cas de problème.

Une image contenant texte, ciel, carte

Description générée automatiquement

### Bannir un utilisateur

Lorsqu’un administrateur clique sur le bouton pour bannir un utilisateur, le client va envoyer la requête à la Gateway qui va vérifier la session. Il retourne un code d’erreur si l’admin n’est pas connecté (401), s’il n’est pas admin (403) ou si les données ne sont pas valides (400). Ensuite, la Gateway va envoyer la requête vers le service REST 2 qui va bannir l’utilisateur sur la base de données. Si l’utilisateur n’existe pas, un code d’erreur 404 est retourné.



# Conception à faire complétement avec EA

## Class Diagram complet avec les explications de chaque application

### Class Diagram APIGateway

### Class Diagram REST 1

Dans notre REST 1, nous avons deux beans pour les livres et les réservations et deux REST qui vont contrôler chacun leur Worker attribué. Les REST vont principalement implémenter les requêtes pouvant provenir de l’API à savoir les CRUD.

Une image contenant diagramme

Description générée automatiquement

### Class Diagram REST 2

Dans notre RESt2, nous avons un bean pour les utilisateurs et un REST qui va contrôler son Worker.

Une image contenant diagramme

Description générée automatiquement

# Hébergement

# Installation du projet complet avec les 5 applications

# Outils, langages (versions, définitions, installations)

# Tests de fonctionnement du projet

# Auto-évaluations et conclusions