Perlini Nilo et Kazanjian Adrian

Documentation

Module 133 – Forum\_Doudix

Table des matières

[Introduction 3](#_Toc196668535)

[Description du projet 3](#_Toc196668536)

[Fonctionnalités 3](#_Toc196668537)

[Analyse 4](#_Toc196668538)

[Maquette 4](#_Toc196668539)

[Use case 7](#_Toc196668540)

[Activity Diagram 10](#_Toc196668541)

[Séquence System global 10](#_Toc196668542)

[Conception 12](#_Toc196668543)

[Diagramme de navigation 12](#_Toc196668544)

[Diagramme Architecte 12](#_Toc196668545)

[Architecte API Gateway 13](#_Toc196668547)

[Architecte Rest 14](#_Toc196668548)

[Développement 14](#_Toc196668549)

[Descente de code 14](#_Toc196668550)

[Tests 17](#_Toc196668551)

[Outils et langage 19](#_Toc196668552)

[Langages de programmation 19](#_Toc196668553)

[Outils de développement 20](#_Toc196668554)

[Bibliothèques 20](#_Toc196668558)

[Docker 20](#_Toc196668559)

[Docker Files 20](#_Toc196668560)

[Docker-Compose 24](#_Toc196668561)

[Conclusion 25](#_Toc196668562)

# Introduction

Dans le cadre du module 133, nous avons réalisé une application WEB utilisant les points suivants :

* Les sessions Utilisateurs
* Les requêtes SQL
* Des bases de données
* Dockers : (DockerFiles, DockerHUB..)
* GitHub : (Repository..)
* Java

Voici donc la documentation du projet que nous avons réalisé (Adrian Kazanjian et Nilo Perlini) durant ce module :

## Description du projet

Voici la description de notre projet :

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom du Projet** | | | **Doudix** |
| **Langage** | | Java-17 | **Description** |
| **Emplacement** | | WEB | Notre projet est de réaliser un forum WEB pour que nous puissions poser des questions et avoir notre propre forum de l’EMF. Crée et managé par des élèves de l’écoles. |
| **Développeurs** | | |
| Adrian | Kazanjian | |
| Nilo | Perlini | |

## Fonctionnalités

Voici les fonctionnalités principales de notre projet :

|  |
| --- |
| **Fonctionnalités** |
| Créer un compte, se connecter, se déconnecter, personnalisé ses informations |
| Créer des postes : (Proposer un thème, donner son avis..etc...) |
| Répondre à des postes : (Affichage utilisateurs, afficher des avis…etc.) |
| Admin : Contrôle des postes (Supprimer, modifier ) |
| Poster des images |
| Admin : Bannissement des utilisateurs |

Une image contenant Police, Graphique, logo, symbole

Le contenu généré par l’IA peut être incorrect.

# Analyse

## Maquette

Page Login : Au lancement du site, nous allons avoir deux options différentes. Soit se connecter à un compte existant ou bien en créer un directement en cliquant sur le bouton de la partie verte.

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Le contenu généré par l’IA peut être incorrect.

Page Register : Cette page s’affiche suite à l’interaction avec le bouton « Sign Up » dans la première page de base. Dans cet affichage, il sera possible de créer un compte afin d’accéder au forum. Il faut donc remplir les champs avec un nom d’utilisateur et un mot de passe.

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

Le contenu généré par l’IA peut être incorrect.

Page Home User : Voilà la page utilisateur après la connexion à un utilisateur client. Il sera possible de créer un poste, de cliquer sur un poste déjà existant afin d’y réponde et interagir avec l’auteur du post. L’utilisateur pourra aussi choisir une catégorie et afficher tous les postes affiliés à cette catégorie.

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

Le contenu généré par l’IA peut être incorrect.

Page Post : Voici la page lorsque l’utilisateur veut ajouter un poste, il doit ajouter un titre, une description, une catégorie et une couleur. Il peut s’il veut ajouter une photo au poste mais cela n’est pas obliger.

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

Le contenu généré par l’IA peut être incorrect.

Page Discussing : Cette page s’affiche lorsque l’utilisateur clique sur le poste publié d’un autre utilisateur. Il pourra répondre en écrivant un message et cela va permettre d’avoir une discussion sur n’importe quel sujet.

Une image contenant texte, capture d’écran, logiciel, Page web

Le contenu généré par l’IA peut être incorrect.

Page Home Administrator : Cette page s’affiche lors de la connexion à un compte administrateur. Il aura la possibilité de supprimer n’importe quel poste et de rechercher un utilisateur dans la liste de tous les utilisateurs créer. Selon son envie il peut aussi bannir un utilisateur en supprimant son compte.

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

Le contenu généré par l’IA peut être incorrect.

## Use case

Use Case Global : Voici le Use Case Global.

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

Le contenu généré par l’IA peut être incorrect.

Usercase\_client1 : Voici le Use Case de la partie utilisateur. Nous avons comme acteur l’utilisateur de base et l’utilisateur authentifié à l’application. Le premier acteur ne peut que se connecter et créer un compte. Le deuxième peut ajouter un poste et y répondre.

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

Le contenu généré par l’IA peut être incorrect.

Usercase\_client2 : Voici le Use Case de la partie admin. Nous avons comme acteur l’utilisateur de base et l’administrateur authentifié à l’application. Le premier acteur ne peut que se connecter et créer un compte. Le deuxième peut supprimer un poste et bannir un utilisateur.

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

Le contenu généré par l’IA peut être incorrect.

Usercase\_APIGateway : Ce diagramme affiche toutes les redirections des APIGateway vers les Ctrl Rest. Il y a donc l’ajout de message, ajout d’un poste, la déconnexion, la connexion et l’inscription pour le client 1. Pour le 2e client, il y a la suppression d’un poste et le bannissement d’un utilisateur.

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

Le contenu généré par l’IA peut être incorrect.

Usercase\_rest1 : Ce diagramme affiche toutes les redirections des APIGateway vers les Ctrl Rest afin d’interagir avec la bonne base de données. Il y a donc l’ajout de message, ajout d’un poste, la déconnexion, la connexion et l’inscription pour le client 1.

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

Le contenu généré par l’IA peut être incorrect.

Usercase\_rest2 : Ce diagramme affiche toutes les redirections des APIGateway vers les Ctrl Rest afin d’interagir avec la bonne base de données. Pour le 2e client, il y a la suppression d’un poste et le bannissement d’un utilisateur.

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

Le contenu généré par l’IA peut être incorrect.

## Activity Diagram

Activity Diagram client-gw-rest1 : Voici le diagramme d’activité pour le client1 afin de créer un poste.

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

Le contenu généré par l’IA peut être incorrect.

Activity Diagram client-gw-rest2 : Voici le diagramme d’activité pour le client2 afin de supprimer un poste.

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

Le contenu généré par l’IA peut être incorrect.

## Séquence System global

Séquence System client-apigw-rest1 : Voici le diagramme de séquence du client1 pour créer un poste.

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

Le contenu généré par l’IA peut être incorrect.

Séquence System client-apigw-rest2 : Voici le diagramme de séquence du client2 pour supprimer un poste.

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

Le contenu généré par l’IA peut être incorrect.

# Conception

## Diagramme de navigation

Dans ce diagramme, tout le site web est conceptualiser sous forme de schéma et de liens. Il montre le processus d’authentification des utilisateurs, la différence des rôles client et administrateur et la redirection en fonction des permissions. L’API Gateway va avoir comme rôle un intermédiaire afin de traiter les requêtes et les envoyés au bon services.

Une image contenant texte, diagramme, Plan, Parallèle

Le contenu généré par l’IA peut être incorrect.

## Diagramme Architecte

Ce diagramme illustre la navigation et les interactions entre les différentes pages, les fichier JavaScript et l’API Gateway. Les utilisateurs commencent par se connecter depuis l’index.jsp ou s’inscrire depuis register.jsp, avec la validation qui est gérée par des scripts JavaScript. Après que les informations ont bien été interprété à l’API Gateway, celle-ci va rediriger l’utilisateur vers la page html lui étant attribué (forumClientSide.html pour les utilisateurs classique et forumAdminSide.html pour les administrateurs).

## Une image contenant texte, capture d’écran, diagramme, ligne Le contenu généré par l’IA peut être incorrect.

## Architecte API Gateway

Dans ce diagramme, nous montrons l’architecture d’une API Gateway qui est utilisé pour la gestion des utilisateurs et la communication avec un backend. L’API Gateway va centralisé l’authtification et l’inscription des utilisateurs depuis un UserManager. Elle interagit avec le cas d’utilisation UseCaseRest1 qui va dépendre de l’interface ItfWrkCtrl. Le composant Wrk, va implémenter l’interface ItfCtrlWrk et communique avec un TCPClient qui va lui permettre de faire des interactions avec d’autre services.

Une image contenant texte, capture d’écran, Post-it, Rectangle

Le contenu généré par l’IA peut être incorrect.

## Architecte Rest

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

Le contenu généré par l’IA peut être incorrect.Ce diagramme va montrer le fonctionnement de la partie Rest du projet avec les requêtes pour interagir avec la base de données. Dans ce diagramme nous pouvons voir notre restController\_client1 qui va le Wrk UserManager car c’est ce dernier qui va faire la requête de création d’utilisateur dans la base de données.

# Développement

Dans ce chapitre je vais expliquer la partie de login de mon application WEB.

## Descente de code

Tous d’abord nous avons notre page HTML avec le bouton de login :

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

Le contenu généré par l’IA peut être incorrect.L’élément : <button id = « btnLogin » class= « btn»>Login</button>**

Une image contenant texte, carte de visite, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.Grâce à cette classe nous allons pouvoir appeler le javascript (Jquerry) :

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

Le contenu généré par l’IA peut être incorrect.Voici ce qui se passe dans le javascript :

Tous d’abord nous récupérons les valeurs données dans les champs :

1. #login-password
2. #login-username

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

Le contenu généré par l’IA peut être incorrect.Puis nous vérifions qu’ils ne sont pas vides et on concatène les infos pour finalement appeler la fonction ajax présente dans le fichier serviceHttp.js :

Cette requête peut se décomposer de la façon suivante :

* **url: 'http://localhost:8080/api/gateway/login',**
  + *Cette ligne indique la destination de la requête, c’est son URL.*
* **type: 'POST',**
  + Cette ligne indique le type de requête, en l’occurrence c’est un type « post », nous voulons donc créer ou mettre a jour une entrée dans le serveur.
* **contentType: 'application/json',**
  + Cette ligne indique le type de donnée envoyée.
* **data: JSON.stringify(loginInfo),**
  + Cette ligne transforme les données reçues en paramètre en JSON.
* **xhrFields: { withCredentials: true },** 
  + Cette ligne permet de forcer le transfert des informations de Session au serveur.
* **success: successCallback,**
* **error: errorCallback**
  + Erreur et successCallback

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

Le contenu généré par l’IA peut être incorrect.Nous sommes donc maintenant dans l’apiGateway de l’application. Nous pouvons observer l’en-tête :

@RestController : Cette annotation permet de définir le modèle du contrôleur

@RequestMapping : Cette annotation définis le chemin d’accès du contrôleur.

Puis il y a en dessous la requête « /login » que notre méthode ajax appelle :

Cette Requête est divisée en plusieurs partie :

D’abord la mise en place de la requête au service concerné :  
Une image contenant texte, capture d’écran, Police, conception

Le contenu généré par l’IA peut être incorrect.

Puis la requête et le résultat selon sa réponse :

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

Le contenu généré par l’IA peut être incorrect.

Voici maintenant mon serviceCtrl1 avec la méthode de login :  
Une image contenant texte, capture d’écran, ordinateur, affichage

Le contenu généré par l’IA peut être incorrect.

Nous pouvons donc constater la même annotation du modèle de serviceRestController avec son url de requête. Ensuite nous avons donc la méthode « /login ». Voici le fonctionnement de la requête décomposé :

Tous d’abord nous allons créer un utilisateur avec les dto reçus en paramètre :

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

Le contenu généré par l’IA peut être incorrect.

Puis nous allons contrôler que l’utilisateur n’est pas null est si ce n’est pas le cas nous allons ensuite valider la connexion de l’utilisateur ou alors au contraire retourner une erreur 401 :

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

Le contenu généré par l’IA peut être incorrect.

Voici maintenant le fonctionnement des services lors du contrôle de connexion fait par le service du serviceCtrl1 :

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

Le contenu généré par l’IA peut être incorrect.

Voici donc notre service qui va s’assurer que l’utilisateur est bien dans la base de données via une méthode du USERRepository.

Voici donc le chemin de connexion d’un utilisateur qui se terminera finalement par cette logique présente dans notre loginCtrl de notre javascript :

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

Le contenu généré par l’IA peut être incorrect.

Donc suivant la réponse de notre requête ajax, nous stockerons tous d’abord les informations dans les données de sessions puis après un petit test préalable pour savoir si une personne est admin ou non, nous redirigerons l’utilisateur sur la page principale du site.

# Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N° | Nom | Procédures | Résultat attendu | Résultat obtenu | Visa |
| 1 | Register d’un utilisateur (Page register) | Insertion d’un nom d’utilisateur et d’un mot de passe puis cliquer sur le bouton register. | Register fonctionnel et donc les entrées username et password affiché dans la DB dans la table user. | Register fonctionnel et donc les entrées username et password affiché dans la DB dans la table user. | Badge Tick1 avec un remplissage uni |
| 2 | Register avec un champ vide (Page register) | Cliquer sur le bouton register avec un des champs vide sans aucune entrée. | Pop-up alerte disant de remplir tous les champs. | Pop-up alerte disant de remplir tous les champs. | Badge Tick1 avec un remplissage uni |
| 3 | Register avec un utilisateur déjà pris (Page index) | Mettre les informations de création d’utilisateur d’un utilisateur déjà existant dans la DB et ensuite cliquer sur le bouton register. | Pop-up alerte définissant une erreur de création de compte. | Pop-up alerte définissant une erreur de création de compte. | Badge Tick1 avec un remplissage uni |
| 4 | Login avec un utilisateur Client (Page login) | Mettre les informations d’un utilisateur client qui existe dans la base de données puis cliquer sur login. | Après avoir cliqué sur login, connexion à la page home.html avec l’utilisateur connecté. | Après avoir cliqué sur login, connexion à la page home.html avec l’utilisateur connecté. | Badge Tick1 avec un remplissage uni |
| 5 | Login avec un utilisateur Admin (Page index) | Mettre les informations d’un utilisateur admin qui existe dans la base de données puis cliquer sur login. | Après avoir cliqué sur login, connexion à la page home.html avec l’utilisateur Admin. | Après avoir cliqué sur login, connexion à la page home.html avec l’utilisateur Admin. | Badge Tick1 avec un remplissage uni |
| 6 | Login avec mauvais identifiant (Page home) | Mettre des informations inexistantes dans la base de données et cliquer sur le bouton login. | Pop-up alerte disant que le nom d’utilisateur ou le mot de passe est incorrect. | Pop-up alerte disant que le nom d’utilisateur ou le mot de passe est incorrect. | Badge Tick1 avec un remplissage uni |
| 7 | Déconnexion d’un utilisateur (Page home) | En étant connecté à un utilisateur, cliquer sur le bouton Déconnexion | Déconnexion totale de l’utilisateur en détruisant totalement la session | Déconnexion totale de l’utilisateur en détruisant totalement la session | Badge Tick1 avec un remplissage uni |
| 8 | Affichage de la liste des postes (Page home) | En se connectant avec un utilisateur, alors récupérer la liste de tous les postes et l’afficher. | Affichage et chargement de tous les postes dans la page home. | Affichage et chargement de tous les postes dans la page home. | Badge Tick1 avec un remplissage uni |
| 9 | Recherche par mot clé (Page home) | Tape run mot-clé dans la barre de recherche | Affichage uniquement des postes contenant ce mot-clé dans le titre. | Affichage uniquement des postes contenant ce mot-clé dans le titre. | Badge Tick1 avec un remplissage uni |
| 10 | Filtrage par catégorie (Page home) | Cliquer sur une catégorie dans le sidebar | Affichage uniquement de postes de cette catégorie | Affichage uniquement de postes de cette catégorie | Badge Tick1 avec un remplissage uni |
| 11 | Réinitialisation du filtre (Page home) | Cliquer sur « Reset » | Tous les postes sont à nouveau affichés | Tous les postes sont à nouveau affichés | Badge Tick1 avec un remplissage uni |
| 12 | Création d’un poste | Remplir tous les champs + image, puis cliquer sur « Add Post » | Le nouveau post s’ajoute à la liste de home.html | Le nouveau post s’ajoute à la liste de home.html | Badge Tick1 avec un remplissage uni |
| 13 | Création de post sans image | Créer un post sans image | L’image par défaut est affichée dans la liste de postes | L’image par défaut est affichée dans la liste de postes | Badge Tick1 avec un remplissage uni |
| 14 | Redirection discussion | Cliquer sur un post | Redirection vers discussion.html avec contenu du post affiché à gauche | Redirection vers discussion.html avec contenu du post affiché à gauche | Badge Tick1 avec un remplissage uni |
| 15 | Publication d’un message | Ecrire un commentaire et cliquer sur « Envoyer » | Le message est affiché instantanément dans la liste à droite | Le message est affiché instantanément dans la liste à droite | Badge Tick1 avec un remplissage uni |
| 16 | Interface admin visible | Se connecter avec un compte admin | Le bouton « Ban Users » est visible | Le bouton « Ban Users » est visible | Badge Tick1 avec un remplissage uni |
| 17 | Interface admin cachée | Se connecter avec un compte non-admin | Le bouton « Ban Users » est caché | Le bouton « Ban Users » est caché | Badge Tick1 avec un remplissage uni |
| 18 | Bannissement d’un utilisateur | En tant qu’admin, ouvrir le pop-up sélectionner un utilisateur, ajouter une remarque et cliquer sur « BAN » | L’utilisateur est supprimé et un message de succès s’affiche | L’utilisateur est supprimé et un message de succès s’affiche | Badge Tick1 avec un remplissage uni |

# Outils et langage

## Langages de programmation

* **Java :** Langage principal utilisé pour le développement du Backend. Grâce à sa robustesse et à sa portabilité, Java permet de créer des applications fiables et maintenables. Il est utilisé en combinaison avec Spring Boot pour structurer la logique métier et gérer les requêtes côté serveur.
* **Spring Boot :** Framework utilisé pour le développement du backend. Il aide à la création d’API REST robuste, en fournissant une configuration rapide et des fonctionnalités prêtes à l’emploi.
* **MySQL :** Sert de système de gestion de base de données relationnelle. Permettant le stockage, l’interrogation et la manipulation efficace des données de l’application.
* **Lombok :** Bibliothèque Java intégrée au projet backend afin de simplifier l’écriture du code. Elle génère automatiquement des éléments répétitifs (getters, setters, constructeurs, etc.), ce qui allège beaucoup le code source.
* **JavaScript :** Utilisé pour le développement côté client (Frontend). Le projet repose sur du JavaScript standard afin de gérer l’interaction avec l’interface utilisateur et effectuer les appels au Backend.

## Outils de développement

## **Visual Studio Code :** Utilisé pour l’écriture et la gestion du code côté frontend (HTML, CSS, JavaScript). C’est un éditeur de code léger mais puissant, avec de nombreuses extensions qui facilitent le développement web.

## **MySQL Workbench :** Utilisé pour la gestion graphique de la base de données MySQL. Il permet de visualiser les tables, exécuter des requêtes SQL, concevoir des schémas, et inspecter les données plus facilement.

## **Postman :** Utilisé pour tester les API REST du backend. Il permet d’envoyer des requêtes HTTP (GET, POST, DELETE, etc.) et de visualiser les réponses, ce qui est essentiel pour vérifier que les services fonctionnent correctement.

## Bibliothèques

* **Spring Boot :** Framework Java utilisé côté backend. Il permet de créer des services web robustes et évolutifs, comme des API REST. Plusieurs modules sont utilisés :
* **spring-boot-starter-web** pour le développement des contrôleurs REST,
* **spring-boot-starter-data-jpa** pour la gestion des bases de données relationnelles avec JPA,
* **spring-boot-starter-test** pour les tests (désactivés ici pour accélérer la compilation).
* **Lombok :** Bibliothèque Java utilisée pour simplifier le code en générant automatiquement les méthodes standards (getters, setters, constructeurs, etc.), réduisant ainsi le code répétitif.
* **MySQL Connector :** Driver JDBC officiel permettant à l'application de communiquer avec la base de données MySQL.
* **Springdoc OpenAPI :** Bibliothèque utilisée pour générer automatiquement la documentation Swagger de l’API REST, facilitant les tests et la lecture des routes disponibles.

## Docker

### Docker Files

Le Dockerfile est divisé en **deux étapes** (multistage build) pour **construire** puis **exécuter** proprement une application Spring Boot développée avec Maven :

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

Le contenu généré par l’IA peut être incorrect.

**Étape 1 – Construction de l’application (build stage)**

Utilise une image légère basée sur Alpine Linux contenant Java 17.

Cette étape sert uniquement à construire le .jar.

Une image contenant texte, carte de visite, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Définit le répertoire de travail /app dans le conteneur.

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

Le contenu généré par l’IA peut être incorrect.

Installe Maven dans l’image pour compiler le projet.

Une image contenant texte, capture d’écran, multimédia, gadget

Le contenu généré par l’IA peut être incorrect.

Copie uniquement le fichier pom.xml pour pré-télécharger les dépendances.Une image contenant texte, capture d’écran, Police, carte de visite

Le contenu généré par l’IA peut être incorrect.

Télécharge toutes les dépendances Maven **sans compiler**, afin d’optimiser les futurs builds (mise en cache).

Une image contenant texte, carte de visite, capture d’écran, conception

Le contenu généré par l’IA peut être incorrect.

Copie le code source de l’application.

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

Le contenu généré par l’IA peut être incorrect.

Compile et package l’application dans un .jar, sans exécuter les tests (option -DskipTests).

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

Le contenu généré par l’IA peut être incorrect.

**Étape 2 – Exécution de l’application (runtime stage)**

Utilise une image plus légère pour exécuter Java, sans Maven ni outils inutiles.

Une image contenant texte, capture d’écran, Police, carte de visite

Le contenu généré par l’IA peut être incorrect.

Répertoire d’exécution de l’application dans le conteneur final.

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

Le contenu généré par l’IA peut être incorrect.

Copie uniquement le fichier .jar construit depuis l’étape précédente.

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Indique que l’application va écouter sur le port **8081** (à mapper lors du docker run).

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

Le contenu généré par l’IA peut être incorrect.

Commande exécutée par défaut pour **lancer l’application** Spring Boot.

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

### Docker-Compose

Voici l’explication de notre docker-compose.yml :

**But apigateway :**

C’est le point d’entrée de toute l’application. Il fait office de passerelle (Gateway) entre le client et les différents microservices REST (restctrl1 et restctrl2).

**Détails :**

Build depuis ***./apiGateway*** avec le Dockerfile de ce dossier.

Expose le port 8080 vers l’extérieur (localhost:8080).

Utilise la variable SPRING\_PROFILES\_ACTIVE=prod pour activer un profil de production Spring Boot.

Connecté au réseau interne backend pour pouvoir dialoguer avec les autres services.

Une image contenant texte, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.

**But restctrl1 :**

C’est le premier service REST de l'application. Il gère la création et la gestion des utilisateurs, des posts et des messages.

**Détails :**

* Build depuis ***./serviceRest1*** avec le Dockerfile associé.
* Expose le port 8081 en interne (redirigé vers le port 8080 du container).
* Connecté à la base de données rest1\_database (service mysql).
* Dépend de mysql pour démarrer seulement après que la base soit prête (healthcheck).
* Configure la connexion MySQL et Hibernate via des variables d’environnement.

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

Le contenu généré par l’IA peut être incorrect.

**But restctrl2 :**

C’est le deuxième service REST spécialisé pour la gestion des bannissements d'utilisateurs.

**Détails :**

* Build depuis ***./serviceRest2*** avec son Dockerfile.
* Expose le port 8082 en interne (redirigé vers le port 8080 du container).
* Connecté à la base de données rest2\_database (service mysql\_ban).
* Dépend de mysql\_ban pour démarrer seulement après que la base soit prête (healthcheck).
* Configure sa propre connexion MySQL indépendante via des variables d’environnement.

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Le contenu généré par l’IA peut être incorrect.

**But mysql (projet-mysql) :**

Base de données principale utilisée par le service restctrl1.  
Elle stocke toutes les informations liées aux utilisateurs, posts et messages.

**Détails :**

* Utilise l’image officielle mysql:8.0.
* Mot de passe root : emf123.
* Crée automatiquement la base de données rest1\_database.
* Monte les données sur un volume Docker nommé mysql\_data.
* Healthcheck intégré pour vérifier que le service est prêt avant d’être utilisé.
* Accessible localement sur le port 3309.

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

Le contenu généré par l’IA peut être incorrect.

**But mysql\_ban (ban-mysql) :**

Base de données secondaire utilisée par le service restctrl2.  
Elle est dédiée uniquement à la gestion des bannissements.

**Détails :**

* Utilise aussi l’image officielle mysql:8.0.
* Mot de passe root : emf123.
* Crée automatiquement la base rest2\_database.
* Monte ses données sur un volume séparé mysql\_ban\_data.
* Healthcheck intégré pour vérifier que le service est prêt.
* Accessible localement sur le port 3310.

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

Le contenu généré par l’IA peut être incorrect.

**But volumes :**

Les volumes sont utilisés pour **stocker les données** MySQL en dehors des containers.  
Cela va permettre de **conserver les données** même si les containers sont arrêtés ou supprimés.

**Détails :**

* mysql\_data : Volume Docker qui stocke de manière persistante les données de la base rest1\_database du service mysql.
* mysql\_ban\_data : Volume Docker qui stocke de manière persistante les données de la base rest2\_database du service mysql\_ban.

Sans volumes, toutes les données seraient perdues à chaque redémarrage ou reconstruction des containers.

**But networks :**

Le réseau backend permet aux services de **communiquer entre eux** de façon privée, isolée du réseau externe.

**Détails :**

* Un réseau Docker de type bridge est créé sous le nom backend.
* Tous les services (apigateway, restctrl1, restctrl2, mysql, mysql\_ban) sont connectés à ce réseau.
* Permet d'utiliser les noms des services (restctrl1, restctrl2, etc.) comme adresses pour les requêtes réseau à l’intérieur du projet.

Cela sécurise la communication entre les services et évite d’exposer inutilement les bases de données ou d'autres services sur Internet.

Une image contenant texte, capture d’écran, carte de visite, Police

Le contenu généré par l’IA peut être incorrect.

# Conclusion

## Auto-évaluation et conclusion de Nilo Perlini

### 6.1 Ce que j’ai appris, ce qui est important

J’ai renforcé mes compétences en développement back-end, surtout avec Spring Boot et les APIs REST. Ce qui m'a le plus marqué, c’est le déploiement via Docker. J’ai compris qu’il ne suffit pas de coder, mais aussi de bien configurer le serveur, gérer les variables d’environnement, et les volumes Docker.

### 6.2 Ce que j’ai apprécié

J’ai aimé coder les fonctionnalités et résoudre des bugs, même si parfois c’était frustrant. Ça m’a permis d’approfondir certains concepts techniques et de devenir plus autonome.

### 6.4 Conclusion personnelle

### 6.4.1 Points forts

* Bonne maîtrise du développement Java et Spring
* Capacité à résoudre des problèmes techniques de manière autonome

### 6.4.2 Points faibles

* Difficultés avec la configuration et le déploiement Docker
* Quelques erreurs de configuration qui m’ont pris du temps à résoudre

## Auto-évaluation et conclusion de Adrian Kazanjian

### 6.1 Ce que j’ai appris, ce qui est important

J’ai appris à construire une architecture microservices complète avec Spring Boot, Docker, Docker Compose et MySQL et j’ai compris l’importance de séparer les services, d'utiliser une API Gateway et de gérer plusieurs bases de données.

### Ce que j’ai apprécié

J’ai apprécié travailler sur un projet concret du développement au déploiement. Voir la communication entre microservices fonctionner correctement et aussi la communication avec mon camarade de projet Nilo Perlini.

### 6.4 Conclusion personnelle

### 6.4.1 Points forts

J’ai une bonne maîtrise de Spring Boot, Docker, JPA et mise en place d’une infrastructure stable. Je m’en suis aussi bien sortie avec la communication au sein du groupe.

### 6.4.2 Points faibles

Quelques pertes de temps sur des détails m’ayant fait cogiter longtemps et donc j’ai vraiment perdu du temps.