# Révision de Java, Docker, JPA, REST et Git.

Emmanuel BRUNO

2024-09-29

## 1 Introduction

Ce projet vise à développer une application Java pour la gestion des utilisateurs et des groupes au sein d'un serveur Discord. L'objectif est de mettre en œuvre les concepts de programmation orientée objet, de persistance de données (JPA), de développement d'API REST et d'intégration avec des services tiers. Les étudiants devront concevoir une architecture robuste, implémenter des fonctionnalités avancées et respecter les bonnes pratiques de développement. Ce projet permettra d'évaluer les compétences en conception, développement et intégration de systèmes informatiques.

## 2 Objectifs pédagogiques

- Maîtrise de Java: POO, collections, exceptions, …
- Gestion de projets: Utilisation de Git, Maven, SonarQube
- Conception d'architectures: Couches MVC, DAO, …
- Tests unitaires et d'intégration: JUnit, Mockito
- Containerisation: Docker, Docker Compose
- Développement d'API REST: JAX-RS, JSON
- Intégration avec des services tiers: Discord4J

## 3 Objectifs Techniques

L'objectif de ce TP est de réviser les concepts de base du langage Java que vous devez maîtriser. Il permettra aussi une révision de la mise en pratique de Git et de Maven.

Vous lirez le sujet en entier. Vous mettrez en place un projet Github à partir lien fournit en cours pour suivre l'avancement de vos révisions (mise en place d'une roadmap, de tickets pour chaque question, suivi des bug et livraisons).

Pour tous les exercices, vous devez écrire la javadoc minimale, des tests utiles. Le code sera vérifié et amélioré avec sonarlint au fur et à mesure de l'avancement.

La compilation, la génération de la documentation et des artefacts (fichier jar et executables) seront réalisés avec Maven.

Fabriquer automatiquement avec maven la meilleures forme excéutable.

Les messages doivent être gérés le Logger du jdk ou mieux de Sl4J.

## 4 Modélisation des données pour Discord

La première étape consiste à concevoir une représentation objet des entités clés de Discord (utilisateur, groupe, rôle, canal). Vous élaborerez un diagramme de classes UML détaillé afin de visualiser les relations entre ces entités et leurs attributs. Vous implémenterez ensuite ces classes en Java, en respectant les principes de la programmation orientée objet (encapsulation, héritage, polymorphisme). Enfin, vous mapperez ces classes Java sur des tables de base de données à l'aide de JPA, en définissant les relations entre les entités (un à un, un à plusieurs, etc.)."

## 5 Dockerisation de l'application et de la base de données

Vous encapsulerez votre application Java et votre base de données dans des conteneurs Docker, créant ainsi des environnements d'exécution isolés et reproductibles. Vous rédigerez un Dockerfile pour chaque service, définissant les instructions nécessaires à la construction de l'image Docker. Vous utiliserez ensuite Docker Compose pour orchestrer ces différents conteneurs et définir les relations entre eux. Cette approche vous permettra de déployer votre application de manière facile et portable sur différentes plateformes."

## 6 Développement de l'API REST avec JAX-RS

Vous développerez une API RESTful complète pour manipuler les utilisateurs, groupes, rôles et canaux de votre application Discord. Vous implémenterez les méthodes HTTP standards (GET, POST, PUT, DELETE) pour effectuer les opérations CRUD (Créer, Lire, Mettre à jour, Supprimer) sur ces entités. Vous utiliserez Jackson pour sérialiser et désérialiser les données au format JSON, assurant ainsi une interopérabilité avec d'autres applications. A terme l'application sera déployée avec OpenLiberty.

## 7 Interaction avec Discord via Discord4J

En vous appuyant sur la puissante bibliothèque Discord4J https://discord4j.com/, vous allez créer un bot Discord personnalisé capable d'interagir de manière dynamique avec votre serveur.

Synchronisation avec votre base de données: Le bot sera conçu pour maintenir une cohérence entre les données de votre base de données et les rôles attribués aux utilisateurs sur Discord. Par exemple,

si un utilisateur acquiert un nouveau statut dans votre base de données, le bot lui attribuera automatiquement le rôle correspondant sur Discord.

Commandes personnalisées: Vous commencerez par implémenter une commande basique /sayhello pour vous familiariser avec la création et le traitement des commandes. Par la suite, vous pourrez étendre cette fonctionnalité pour créer des commandes plus complexes, telles que la gestion des rôles, la recherche d'informations ou encore l'organisation de jeux.

Événements Discord: Le bot sera capable de réagir à différents événements sur votre serveur, comme l'ajout d'un nouveau membre, la création d'un nouveau canal, etc. Cela vous permettra de créer des fonctionnalités plus riches et personnalisées.