Application de Gestion de Crédits Bancaires - Backend

1. Introduction

Ce projet estune application de gestion de crédits bancaires dont le backend est développée avec Spring Boot. L'application permet de gérer différents types de crédits (Personnel, Immobilier et Professionnel) et leurs remboursements associés.

2. Analyse des Besoins

2.1. Règles Métier

— Un client peut souscrire à plusieurs crédits

— Trois types de crédits distincts : Personnel, Immobilier et Professionnel

— Chaque crédit peut avoir plusieurs remboursements

— Gestion des statuts des crédits (En cours, Accepté, Rejeté)

— Suivi des remboursements (Mensualités et Remboursements anticipés)

2.2. Modèle de Données

Le système est structuré autour de quatre entités principales :

1. **Client** : Représente les informations des clients
2. **Credit** : Classe abstraite représentant les crédits
3. **Remboursement** : Gère les remboursements des crédits
4. **Types de Crédits** : Héritent de la classe Credit (Personnel, Immobilier, Professionnel)

3. Architecture Technique

3.1. Stack Technologique

— Framework : Spring Boot

— Base de données : MySQL

— ORM : Hibernate

— API : REST

3.2. Structure et Diagramme du Projet

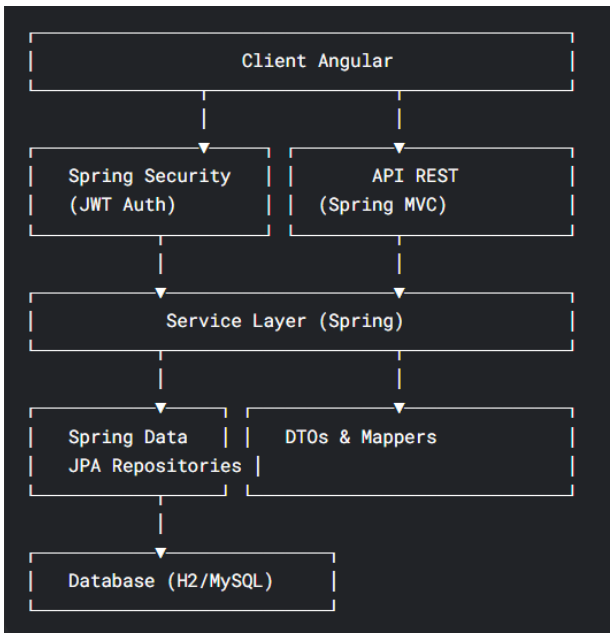

FIGURE 1 – Architecture générale du projet

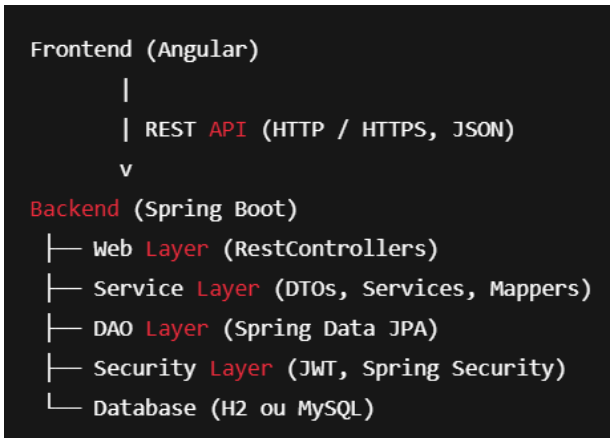

FIGURE 2 – Architecture technique détaillée

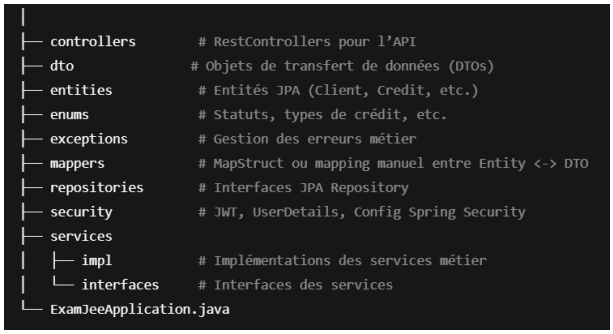

FIGURE 3 – Structure d'un projet Spring Boot

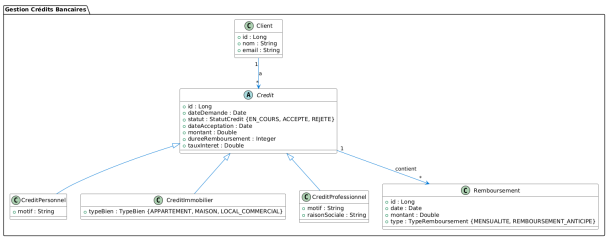

FIGURE 4 – Diagramme de classe des entités

4. Modèle de Données Détaillé

4.1. Entité Client

Attributs :

- id (Long) : Identifiant unique
- nom (String) : Nom du client
- email (String) : Adresse email

Relations :

- One-to-Many avec Credit

4.2. Entité Credit (Abstraite)

Attributs :

- id (Long)
- dateDemande (LocalDate)
- statut (Enum) : En cours, Accepté, Rejeté
- dateAcceptation (LocalDate)
- montant (BigDecimal)
- dureeRemboursement (Integer)
- tauxInteret (BigDecimal)

Relations :

- Many-to-One avec Client
- One-to-Many avec Remboursement

4.3. Types de Crédits

4.3.1. CreditPersonnel

- Hérite de Credit
- Attribut : motif (String)

4.3.2. CreditImmobilier

- Hérite de Credit
- Attribut : typeBien (Enum : Appartement, Maison, Local Commercial)

4.3.3. CreditProfessionnel

— Hérite de Credit

— Attributs : motif (String), raisonSociale (String)

4.4. Entité Remboursement

Attributs :

— id (Long)

— date (LocalDate)

— montant (BigDecimal)

— type (Enum) : Mensualité ou Remboursement anticipé

Relations :

— Many-to-One avec Credit

5. Stratégie de Persistance

5.1. Mapping Objet-Relationnel

— Utilisation de JPA/Hibernate

— Stratégie d'héritage : `SINGLE_TABLE`

— Colonne discriminante : `type_credit`

5.2. Configuration de la Base de Données

— SGBD : MySQL

— Mode de génération du schéma : `update`

— Configuration via `application.properties`

6. Couche d'Accès aux Données

6.1. Repositories

— **ClientRepository** : Gestion des clients

— **CreditRepository** : Gestion des crédits

— **RemboursementRepository** : Gestion des remboursements

6.2. Méthodes Principales

— Recherche par identifiant

— Recherche des crédits par client

— Recherche des remboursements par crédit

7. Couche Service

7.1. Services Principaux

— **CreditService**
— **ClientService**
— **RemboursementService**

7.2. Fonctionnalités Implémentées

— Création de crédits
— Validation des données
— Calcul des remboursements
— Gestion des statuts

8. API REST

8.1. Points d'Entrée

— `/clients` : Gestion des clients
— `/credits` : Gestion des crédits
— `/remboursements` : Gestion des remboursements

8.2. Opérations Disponibles

— **GET** : Récupération des données
— **POST** : Création de nouvelles entités
— **PUT** : Mise à jour des entités
— **DELETE** : Suppression d'entités

8.3. Détails des Contrôleurs

8.3.1. ClientController

— `GET /clients`
— `GET /clients/{id}`
— `POST /clients`
— `PUT /clients/{id}`
— `DELETE /clients/{id}`
— `GET /clients/{id}/credits`

8.3.2. CreditController

- `GET /credits`
- `GET /credits/{id}`
- `POST /credits`
- `PUT /credits/{id}`
- `DELETE /credits/{id}`
- `GET /credits/{id}/remboursements`
- `PUT /credits/{id}/status`
- `POST /credits/{id}/simulate`

8.3.3. RemboursementController

- `GET /remboursements`
- `GET /remboursements/{id}`
- `POST /remboursements`
- `PUT /remboursements/{id}`
- `GET /remboursements/credit/{creditId}`
- `GET /remboursements/status/{status}`

9. Sécurité et Validation

9.1. Validation des Données

- Validation des montants
- Vérification des dates
- Contrôle des statuts

9.2. Gestion des Erreurs

- Messages d'erreur personnalisés
- Codes HTTP appropriés
- Logging des exceptions

10. Installation et Configuration

10.1. Prérequis

- Java 17 ou supérieur
- Maven
- MySQL

10.2. Configuration

1. Cloner le repository
2. Configurer la base de données dans `application.properties`
3. Exécuter `mvn clean install`
4. Lancer l'application avec `mvn spring-boot:run`