ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

Projet :

Réalisation d'une plateforme de Banking Digitale avec une architecture JEE

Filière : Ingénierie Informatique et réseaux

Réalisé par : Wahb LAHLALI

Encadre par : Mr. Mohammed Youssfi

Année Universitaire :

2024/2025

INTRODUCTION GENERALE

Afin de compléter mon module JEE, et dans le cadre de on cursus scolaire, le développement d'une application web de Banking Digitale était nécessaire, afin d'acquérir une connaissance solide en ce qui concerne l'architecture JEE, et réaliser un projet en utilisant Java SPRING, avec Spring Security cote BackEnd, et Angular TypeScript en FrontEnd.

Ce rapport englobe la totalité des détails de la réalisation de ce projet.

CHAPITRE 1 : INTRODUCTION AU PROJET

1 – Introduction :

Ce premier chapitre vise à exposer les grandes lignes de mon projet académique intitulé EBANK. J'y préciserai les objectifs que je souhaite atteindre. Cette introduction établira donc les fondations indispensables pour une compréhension approfondie du projet, qui sera détaillée dans les chapitres suivants.

2-Contexte du Projet :

Ce projet académique m'amène à concevoir un système bancaire intelligent, une étape essentielle pour mon Projet JEE. L'objectif est de créer une solution bancaire à la fois innovante et opérationnelle.

3-Objectifs :

Le projet de système bancaire pour EBANK est centré des objectifs principaux :

- **Faciliter la Gestion des Comptes .**
- **Automatiser les Transactions.**
- **Effectuer les opérations : Débit et crédit**.
- **Améliorer l'Efficacité et la Fiabilité du Service .**
- **.....**

4-Conclusion :

Dans ce chapitre, j'ai établi les bases nécessaires pour une compréhension claire du projet.

CHAPITRE 2 : CONCEPTION ET OUTILS DE TRAVAIL

1-Introduction :

Dans cette section, je vais présenter le diagramme des cas d'utilisation du projet, les besoins fonctionnels ainsi que l'architecture adoptée pour ce projet.

2-Besoins fonctionnelles :

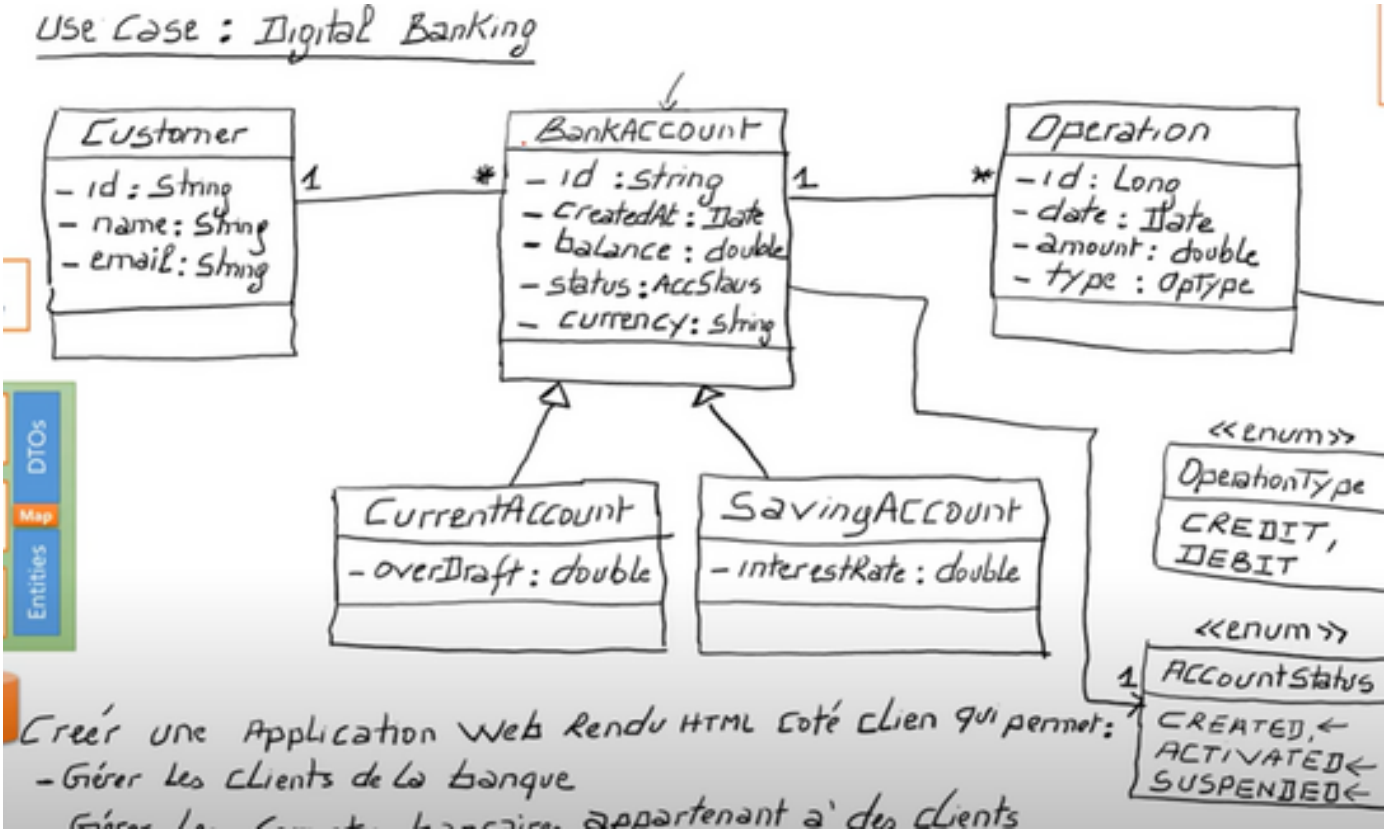
Mon application vise à assurer une gestion efficace d'un service bancaire en ligne, en incluant les fonctionnalités suivantes :

- Gestion des comptes bancaires : Permet d'ajouter et de supprimer des clients, ainsi que de gérer les informations de leurs comptes.
- Opérations bancaires : Permet aux utilisateurs d'effectuer des dépôts, des retraits et des transferts d'argent.
- Historique des opérations : Garde un historique détaillé des transactions pour chaque client.
- Gestion des erreurs : Identifie et résout les problèmes en temps réel pour minimiser les dysfonctionnements.

- Authentification sécurisée des utilisateurs : Assure une autorisation sécurisée pour chaque accès.

Mon application permet aux utilisateurs d'accéder à ces fonctionnalités de manière sécurisée et efficace, facilitant ainsi une gestion optimale de leurs besoins bancaires en ligne.

3-Diagramme Cas d'Utilisation :



Le diagramme de cas d'utilisation illustre l'interaction des utilisateurs avec le système développé et présente le modèle de données pour mon application. Voici les principales entités incluses :

- Customer (Client) : Représente les clients de la banque.

- BankAccount (Compte Bancaire) : Gère les informations de base des comptes bancaires.
- CurrentAccount (Compte Courant) : Spécialisation de BankAccount avec des attributs supplémentaires comme le découvert autorisé.
- SavingAccount (Compte d'Épargne) : Spécialisation de BankAccount avec des attributs supplémentaires comme le taux d'intérêt.
- Operation (Opération) : Représente les transactions effectuées sur les comptes (crédit ou débit).
- Enums (Énumérations) :
- TypeOperation : Définit le type de l'opération (CREDIT, DEBIT).
- AccountStatus : Indique le statut du compte (CREATED, ACTIVATED, SUSPENDED).

Chaque client peut posséder plusieurs comptes bancaires, et chaque compte peut être associé à plusieurs opérations. Les comptes bancaires incluent des attributs tels que l'identifiant, la date de création, le solde, le statut et la devise. Les spécialisations (compte courant et compte d'épargne) ajoutent des attributs spécifiques pour répondre aux besoins fonctionnels différents, comme le découvert autorisé pour les comptes courants et le taux d'intérêt pour les comptes d'épargne.

L'application web résultante permettra de gérer efficacement les clients de la banque ainsi que leurs différents comptes bancaires, en facilitant la gestion des opérations courantes telles que les dépôts, les retraits et les transferts d'argent, tout en maintenant un historique détaillé des transactions pour chaque compte.

4-Technologies Utilisées :

Pour le développement de mon projet eBank, j'ai choisi une combinaison d'outils et de technologies robustes afin d'assurer une solution sécurisée, performante et facilement maintenable :

- IntelliJ IDEA : Environnement de développement intégré (IDE) choisi pour sa prise en charge avancée de Java et ses fonctionnalités qui augmentent ma productivité en tant que développeur.
- Java : Langage de programmation principal utilisé pour sa polyvalence et sa fiabilité, particulièrement adapté aux applications d'entreprise.
- Spring Security : Module du framework Spring intégré pour la gestion de la sécurité. Il offre des services complets d'authentification et d'autorisation, assurant ainsi une protection robuste des données sensibles des utilisateurs.
- MySQL : Système de gestion de base de données relationnelle utilisé pour stocker les données de l'application. Choisi pour sa performance, sa fiabilité et sa capacité à gérer efficacement les transactions.
- Swagger : Outil utilisé pour documenter et tester les API REST. Il simplifie la compréhension et l'interaction avec les services web, facilitant ainsi la collaboration entre développeurs et parties prenantes.
- Angular : Framework utilisé pour le développement de l'interface utilisateur front-end. Angular permet de construire des interfaces utilisateur dynamiques et réactives, tout en offrant une architecture modulaire et des performances optimisées.

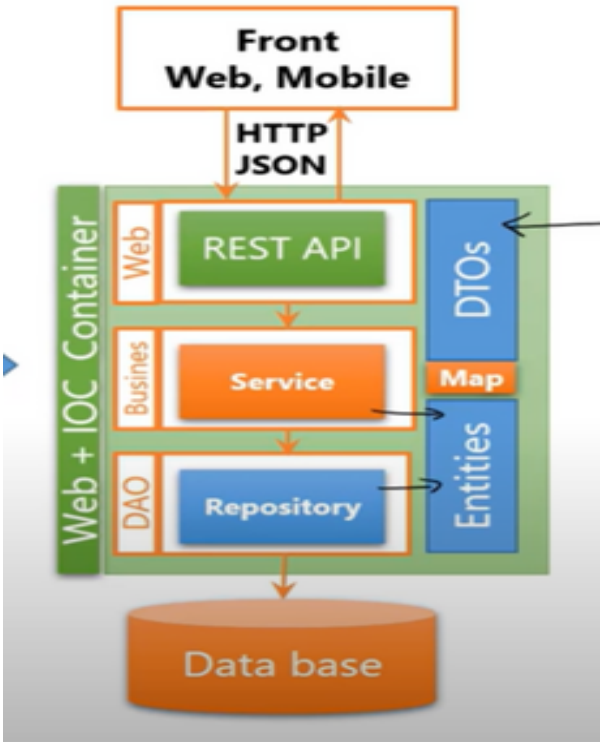
En combinant ces outils, j'établis une base solide pour le développement de l'application eBank. Cette approche garantit la qualité du code, la sécurité des données et des transactions, ainsi que des performances optimales à travers tous les aspects de l'application.

CHAPITRE 3 : ARCHITECTIRE DU PROJET

1-Introduction :

Dans ce chapitre, je vais décrire l'architecture choisie pour le développement de ce projet.

2-Architecture du Projet :



Cette illustration montre les différentes couches de l'application :

- **Interface utilisateur (Frontend) :**

Plateformes : Web, Mobile

Protocole de communication : HTTP et JSON

Envoie des requêtes HTTP à l'API REST et reçoit des réponses au format JSON.

Web + Conteneur IOC :

- **API REST :**

Traite les requêtes envoyées par l'interface utilisateur.

Transmet les requêtes à la couche de service.

- **Couche de service :**

Contient la logique métier de l'application.

Reçoit les requêtes de l'API REST et transforme les données en opérations métier.

Appelle la couche DAO pour les opérations de base de données.

Utilise les DTOs pour transférer les données entre les différentes couches.

- **Objet d'accès aux données (DAO) :**

- **Référentiel (Repository) :**

Gère les interactions avec la base de données.

Reçoit les appels de la couche de service pour effectuer des opérations CRUD (Créer, Lire, Mettre à jour, Supprimer) sur les entités.

Utilise les entités pour structurer et stocker les données dans la base de données.

- **Objets de transfert de données (DTOs) :**

Utilisés pour échanger des données entre les différentes couches.

Fournissent une structure de données standardisée pour les échanges entre la couche de service et les autres couches.

- **Entités :**

Représentent les modèles de la base de données.

Utilisées par le référentiel pour les opérations sur la base de données.

- **Mappage (Map) :**

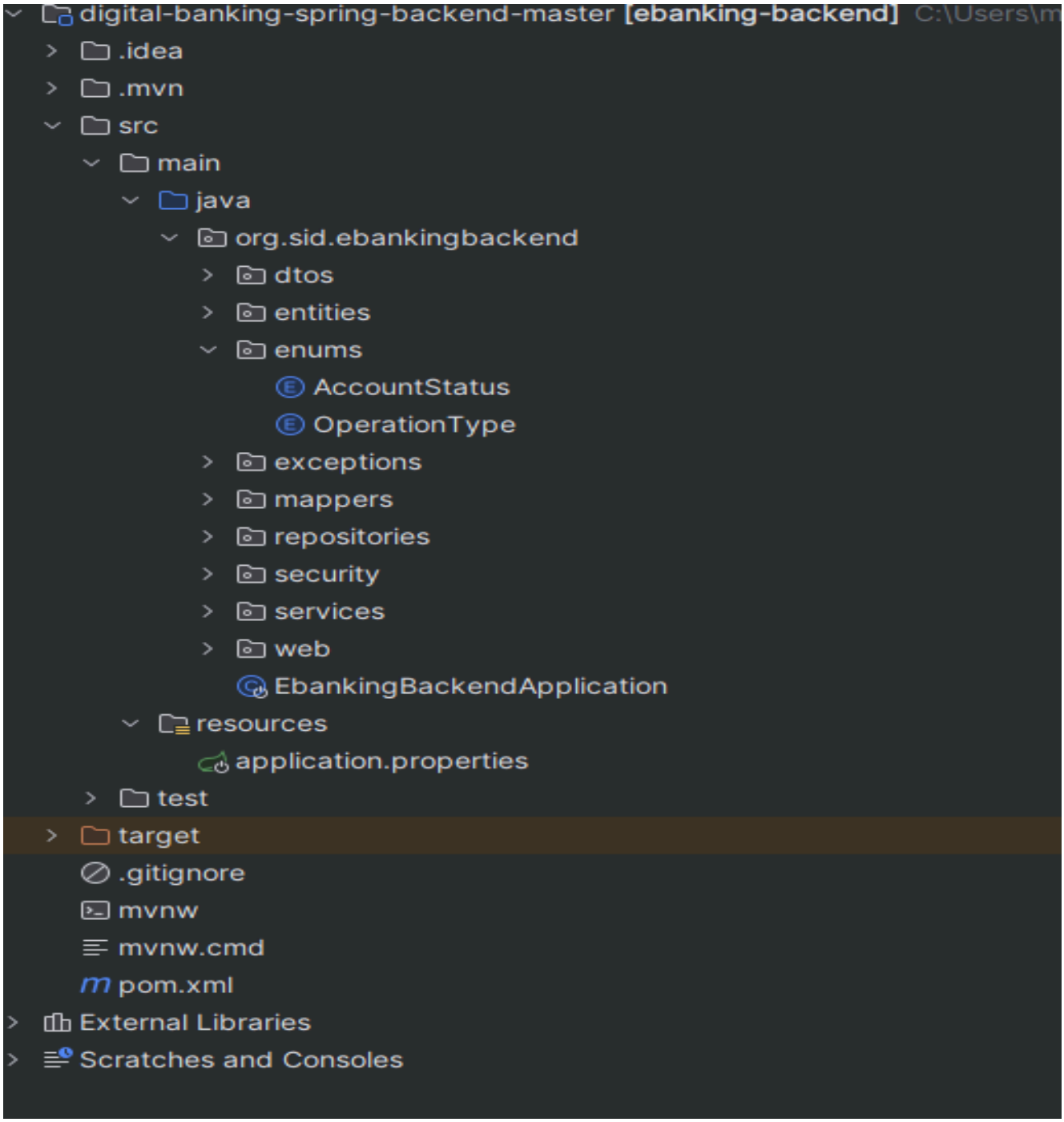
Convertit les données entre les DTOs et les entités.

Veille à ce que les données soient correctement formatées pour chaque couche, facilitant ainsi la communication entre la couche de service et le référentiel.

- **Base de données :**

Stocke les données de manière persistante.

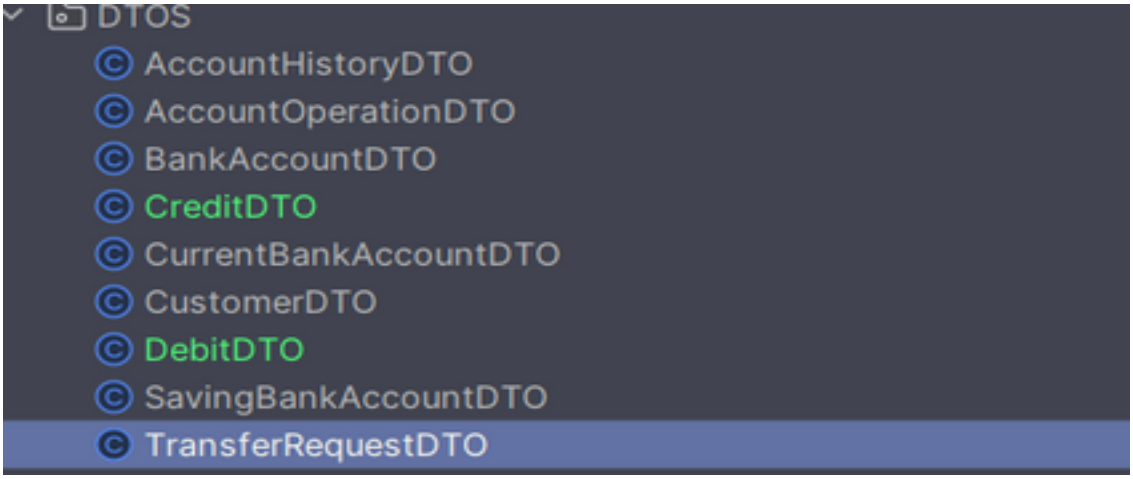
Gérée par le référentiel pour les opérations d'accès et de modification des données.



3-Structure du Projet :

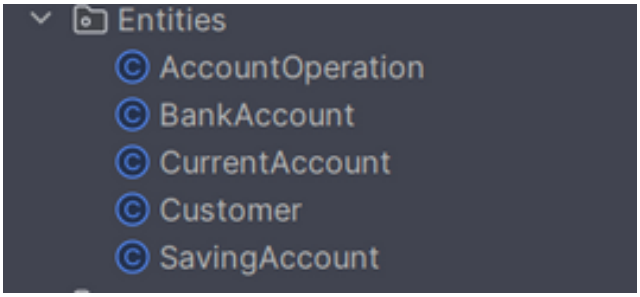
Cette figure représente la structure du projet , comme indique l'image j'ai un ensemble de package bien définis chaque package représente une couche de l'architecture :

1. **DTOs :**



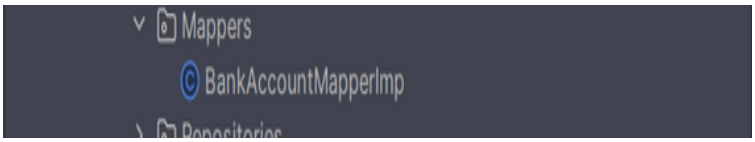
Le package DTOs (Data Transfer Objects) dans le projet eBank facilite le transfert de données entre les différentes couches de l'application, notamment entre la couche de service et la couche de présentation. Les DTOs sont des objets simples, sans logique métier, qui encapsulent les données pour les opérations de lecture et d'écriture. Ils permettent de séparer les modèles de données internes des données exposées aux clients, réduisant ainsi le couplage et augmentant la sécurité en contrôlant les données accessibles.

2 :Entités :



Le package entities dans le projet eBank est un élément essentiel de la couche de persistance de l'application. Il comprend les classes qui représentent les modèles de données, correspondant directement aux tables de la base de données. Chaque entité est une classe annotée avec des annotations JPA (Java Persistence API), définissant la relation entre la classe et la table de la base de données, ainsi que les relations entre les différentes entités (par exemple, les relations one-to-many ou many-to-many).

3-Mappers

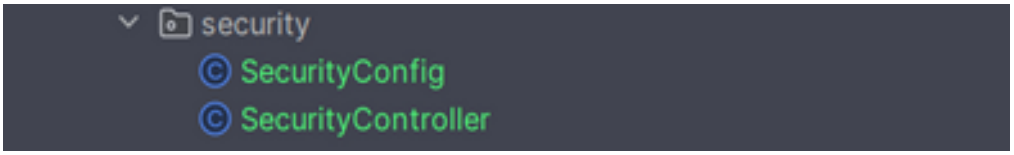


L e package mappers dans le projet eBank est essentiel pour convertir les objets entre les différentes couches de l'application. Ce package contient des classes et des interfaces qui mappent les entités de la base de données (provenant du package entities) vers des DTOs (Data Transfer Objects) et inversement.

4-Repositories

Le package repositories dans le projet eBank est essentiel pour la couche de persistance de l'application. Il contient des interfaces définissant les méthodes d'accès aux données et les opérations CRUD (Create, Read, Update, Delete) sur les entités.

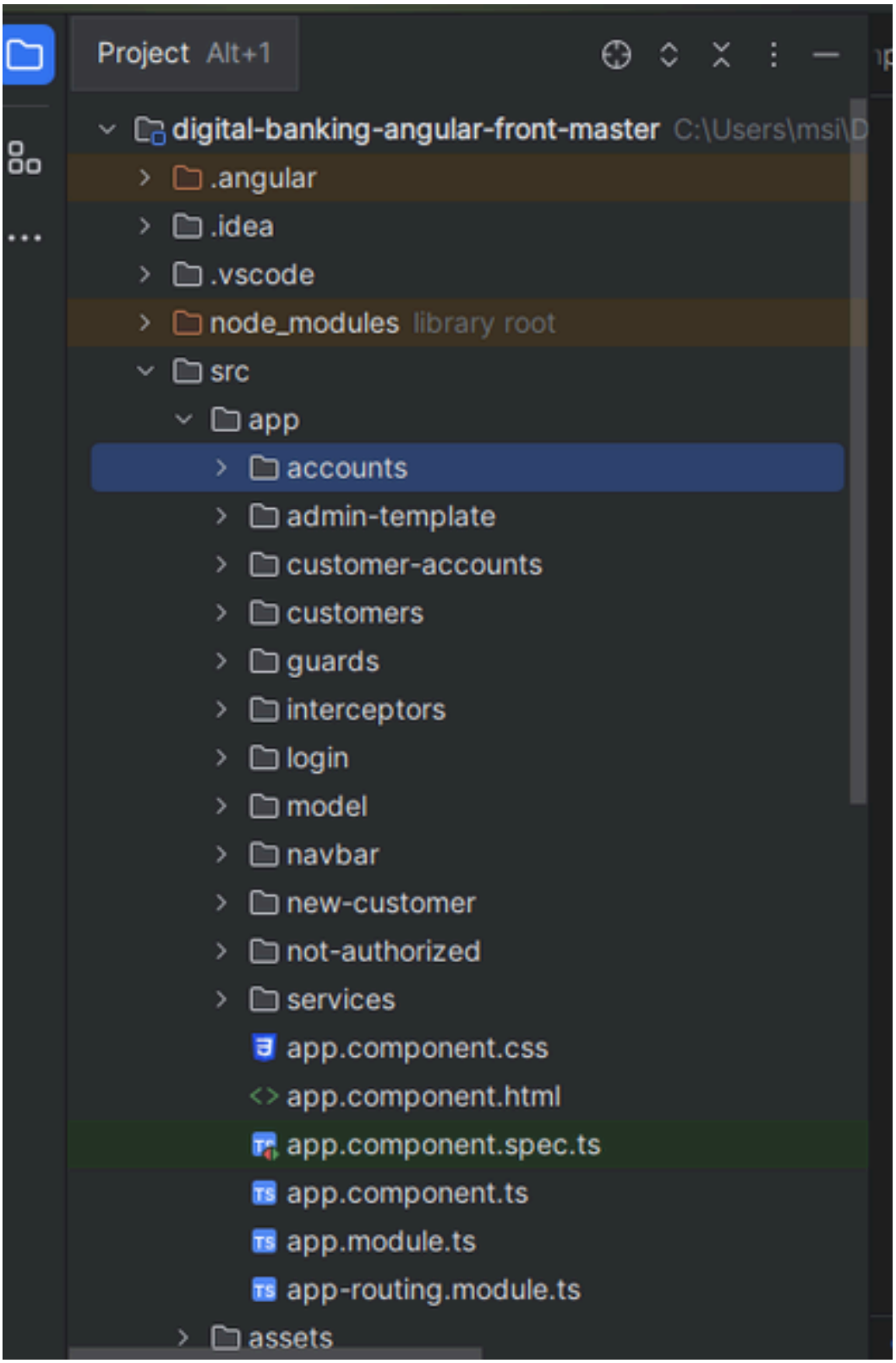
5- Securite :



Le package security dans le projet eBank est consacré à la configuration et à la gestion de la sécurité de l'application. Il contient principalement deux classes : SecurityConfig et SecurityController.

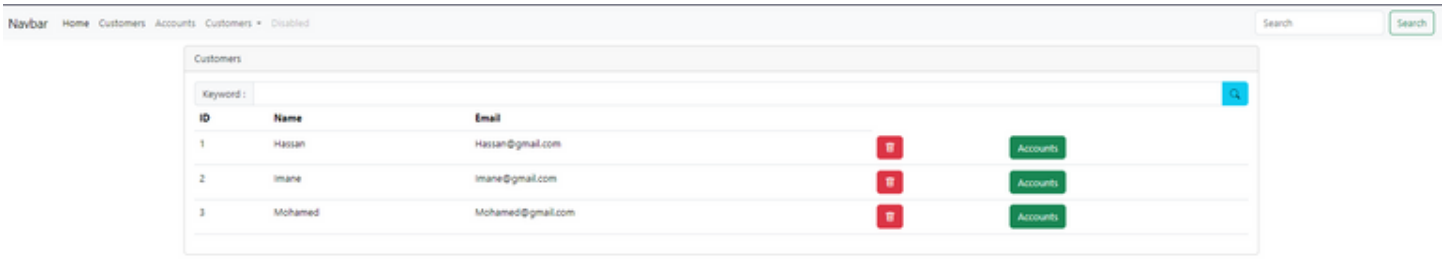
- SecurityConfig : Cette classe configure la sécurité de l'application en utilisant Spring Security. Elle définit les règles de sécurité, y compris les stratégies d'authentification et d'autorisation, les configurations des filtres de sécurité et les paramètres de session. En centralisant ces configurations, SecurityConfig permet de sécuriser efficacement les endpoints de l'application, garantissant que seuls les utilisateurs autorisés peuvent accéder aux ressources protégées.
- SecurityController : Cette classe agit comme un contrôleur REST pour gérer les requêtes liées à la sécurité, telles que l'authentification des utilisateurs, l'inscription et potentiellement la gestion des rôles et des permissions. SecurityController fournit des endpoints spécifiques pour les actions de sécurité, facilitant ainsi l'interaction entre le client et les services de sécurité de l'application.

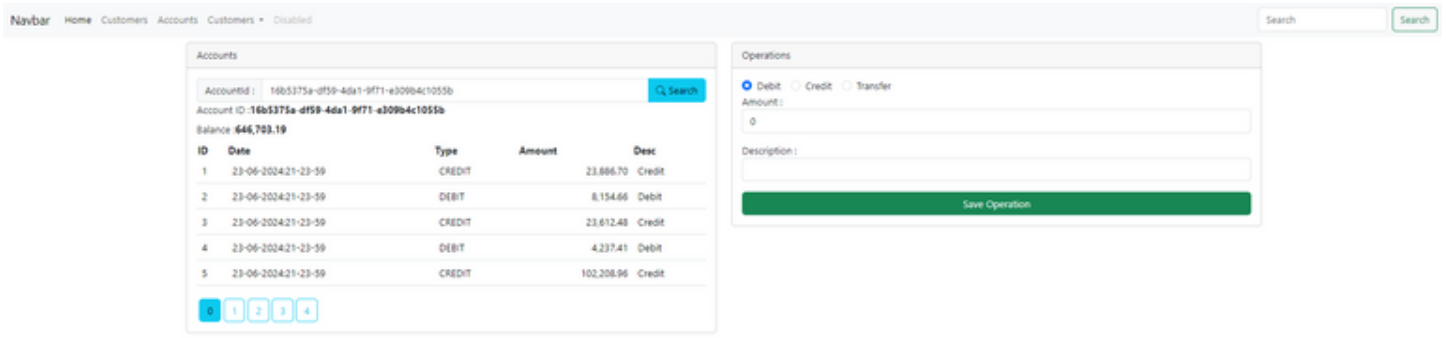
4-La Partie Front-End :



Cette organisation indique que le projet est structuré en différents modules ou composants de fonctionnalités comme accounts, customers et navbar. Chaque module contient ses propres fichiers de styles, de templates HTML et de scripts TypeScript. De plus, il y a un répertoire model qui contient des fichiers TypeScript définissant les modèles de données utilisés dans l'application.

Chapitre 4 : Les Interfaces de E-Bank





Conclusion :

Ce rapport a exploré les aspects essentiels de notre application Angular, en mettant en évidence la structure et les fonctionnalités des éléments constitutifs. Nous avons détaillé comment ces éléments facilitent la création d'une interface utilisateur modulaire et réutilisable, ce qui améliore la maintenabilité de l'application.

L'interface pour visualiser les transactions par comptes bancaires permet une gestion efficace des opérations grâce à une pagination intuitive et des options pour initier de nouvelles actions. De plus, l'interface dédiée à l'ajout de nouveaux clients offre une expérience utilisateur fluide en assurant la validité des données via des formulaires interactifs et des validations intégrées.

Ces éléments, grâce à leur architecture bien conçue, jouent un rôle crucial dans l'amélioration de l'expérience utilisateur et la solidité de notre application.