***ECOLE MAROCAINE DES SCIENCES DE L’INGENIEUR***

**Projet Jee**

Présenté par :

**Leuleu Walid**

Spécialité : **Ingénierie informatique et réseaux**

**Application Web pour la gestion d’une banque :**

**Digital Banking**

Sommaire :

I/ Diagramme de cas d’utilisation

*II/ Backend :*

1-Class Account operation :

2-class bank Account :

3-Class current account :

4-class custumer :

5-class saving account :

6-class account history

7-class credit dto :

8-class credit bank account dto :

9-class custumer dto :

10-class debit dto :

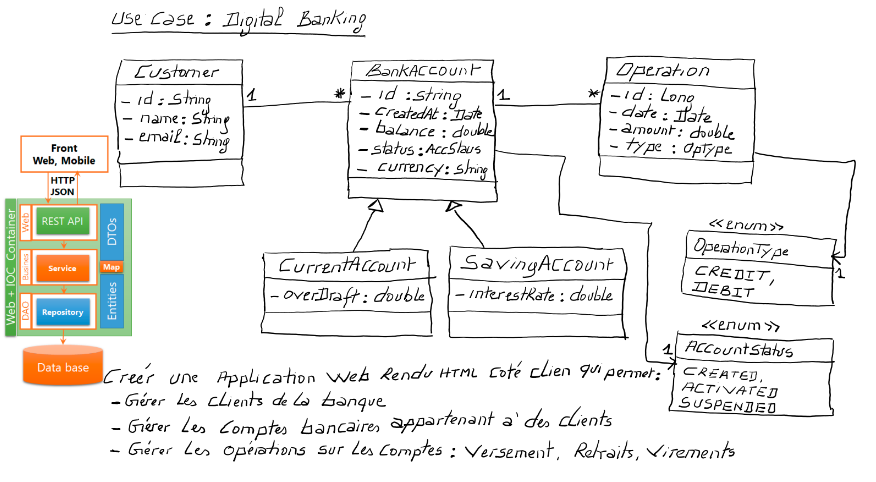
11-saving bank account dto :

12-enum account status :

13-enum operation type :

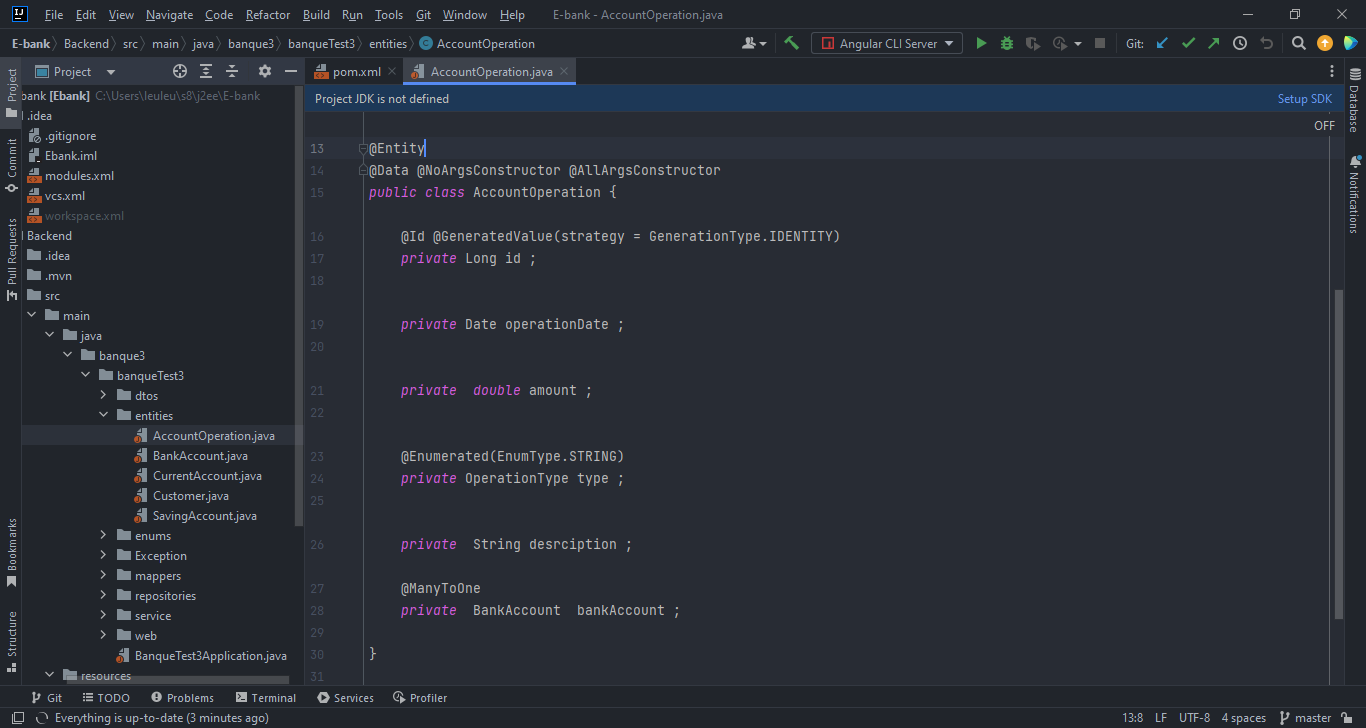
*III/ frontend*

I-/ cas d’utilisation :



II-/Backend :

## 1-Class Account operation :



# 2-class bank Account :

# 

# 3-Class current account :

# 

# Une image contenant capture d’écran, texte, logiciel, Logiciel multimédia Description générée automatiquement4-class custumer :

# 5-class saving account :

# 

# 6-class account history

# 

# 7-class credit dto

# 

# 8-class credit bank account dto :

# 

# 9-class custumer dto :

# 

# 10-class debit dto :

# 

# 11-saving bank account dto :

# 

# 12-enum account status :

# 

# 13-enum operation type :

# 

# 14-enum Operation Type :

# Une image contenant capture d’écran, logiciel, texte, Logiciel multimédia Description générée automatiquement

# 15- Exceptions :

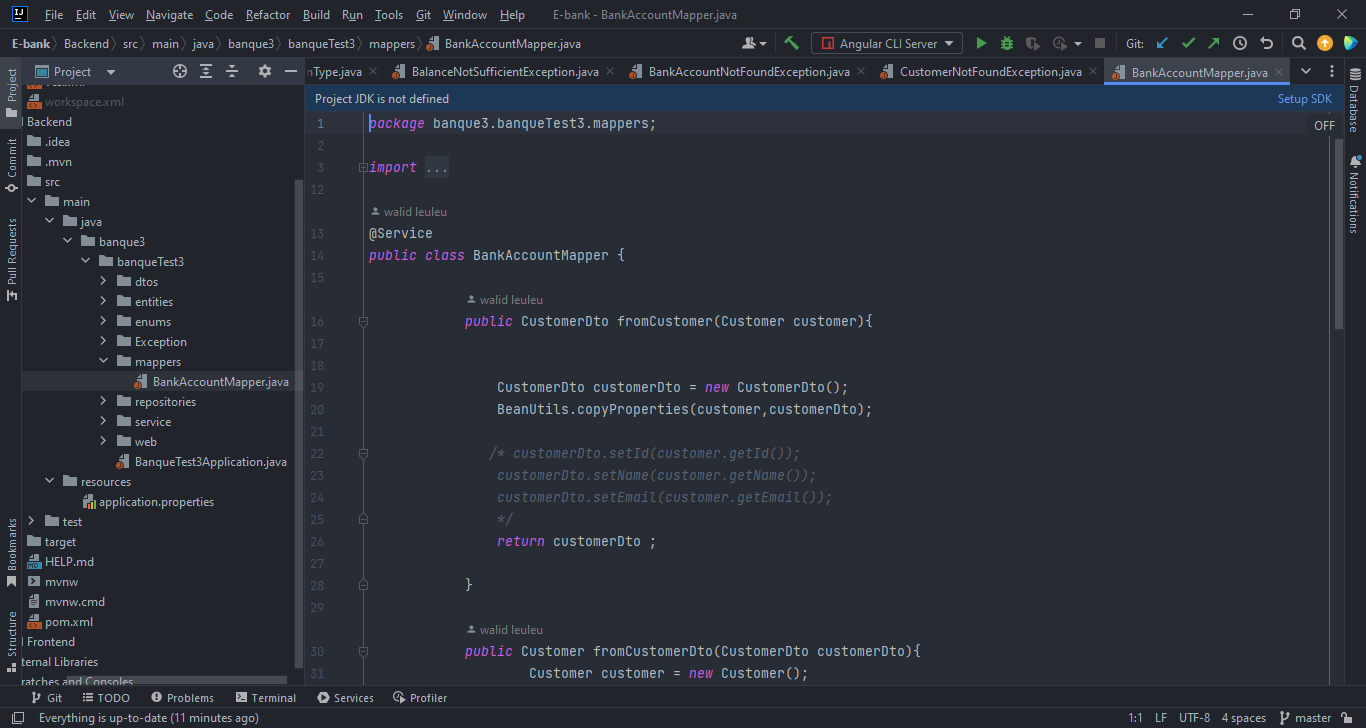
# 

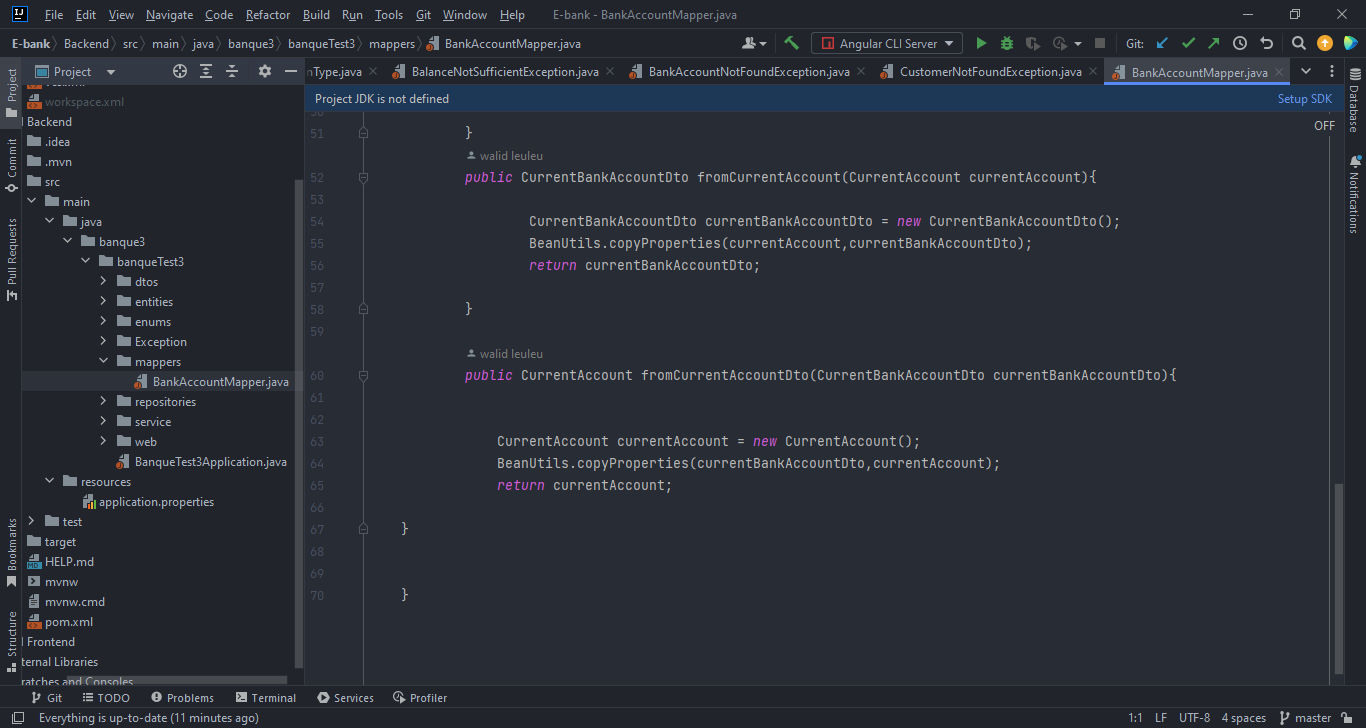
Cette interface est utilisée pour interagir avec la table des clients dans la base de données.

Elle étend **JpaRepository<Customer, Long>**, ce qui signifie qu'elle est paramétrée pour travailler avec l'entité **Customer** et utilise un identifiant de type **Long**.

Elle définit une méthode personnalisée **findCustomerByName(String kw)** qui utilise une requête JPQL (**@Query**) pour rechercher des clients dont le nom est similaire à un mot-clé spécifié.

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

Description générée automatiquement

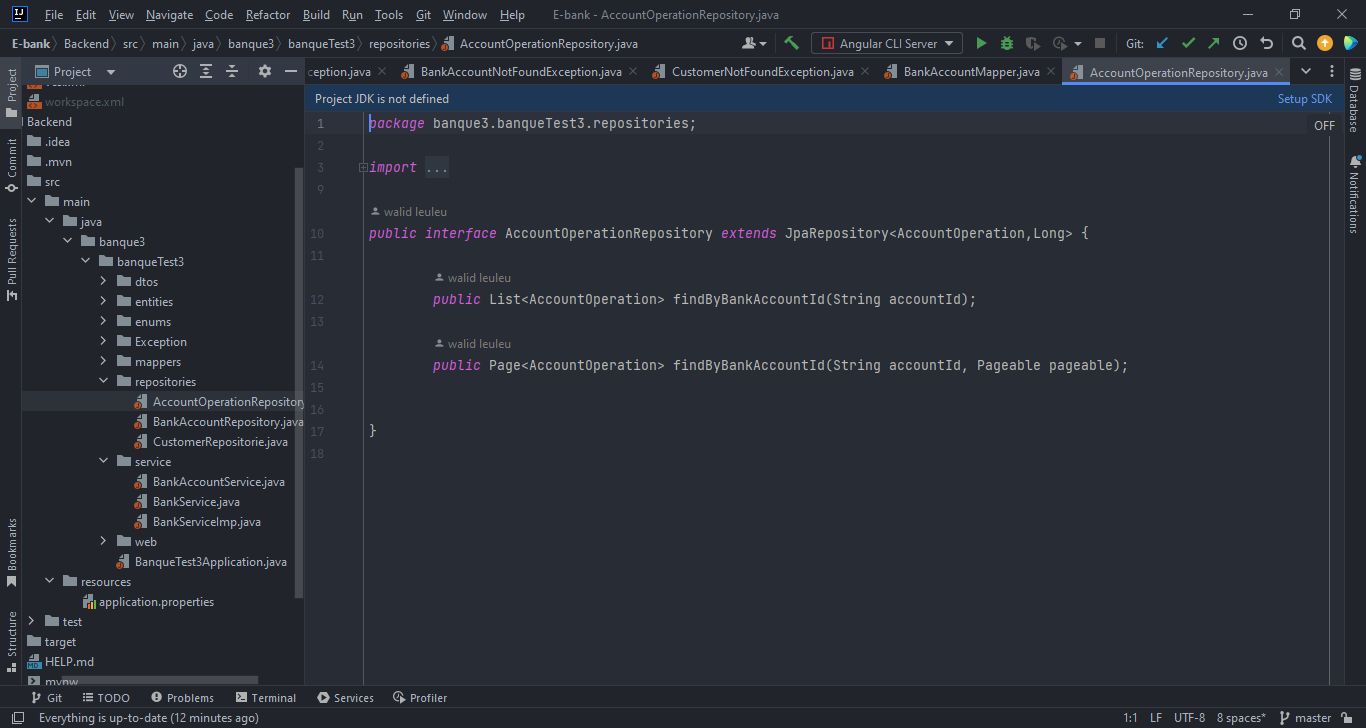
 La classe BankAccountMapper a été développée pour faciliter la conversion entre les entités du domaine et les objets DTO dans le système bancaire. Cette classe joue un rôle clé dans la manipulation des données des comptes bancaires et des clients, en assurant une transition fluide entre les différentes couches de l'application. La classe BankAccountMapper est située dans le package banque3.banqueTest3.mappers et est annotée avec @Service, ce qui indique qu'elle est gérée par le framework Spring. Elle met en œuvre plusieurs méthodes pour effectuer des conversions d'objets, permettant ainsi de passer des entités du domaine aux objets DTO et vice versa.Voici un aperçu des principales méthodes de conversion fournies par cette classe : fromCustomer(Customer customer) : Cette méthode convertit un objet Customer en un objet CustomerDto. Elle utilise la méthode BeanUtils.copyProperties() de Spring pour copier les propriétés de l'objet source vers l'objet cible. Cela facilite la transmission des données du client vers la couche de présentation ou la couche de services. fromCustomerDto(CustomerDto customerDto) : Cette méthode effectue l'opération inverse de la méthode précédente. Elle prend un objet CustomerDto et le convertit en un objet Customer, ce qui permet de restaurer les données du client lorsqu'elles sont reçues depuis la couche de présentation.

fromSavingBankAccount(SavingAccount savingAccount) : Cette méthode convertit un objet SavingAccount en un objet SavingBankAccountDto, facilitant ainsi le transfert des données des comptes d'épargne vers les couches supérieures de l'application.

fromSavingBankAccountDto(SavingBankAccountDto savingBankAccountDto) : Cette méthode effectue l'opération inverse de la méthode précédente. Elle prend un objet SavingBankAccountDto et le convertit en un objet SavingAccount, permettant de restaurer les données des comptes d'épargne à partir des couches supérieures.

fromCurrentAccount(CurrentAccount currentAccount) : Cette méthode convertit un objet CurrentAccount en un objet CurrentBankAccountDto, facilitant ainsi la transmission des données des comptes courants dans le système bancaire.

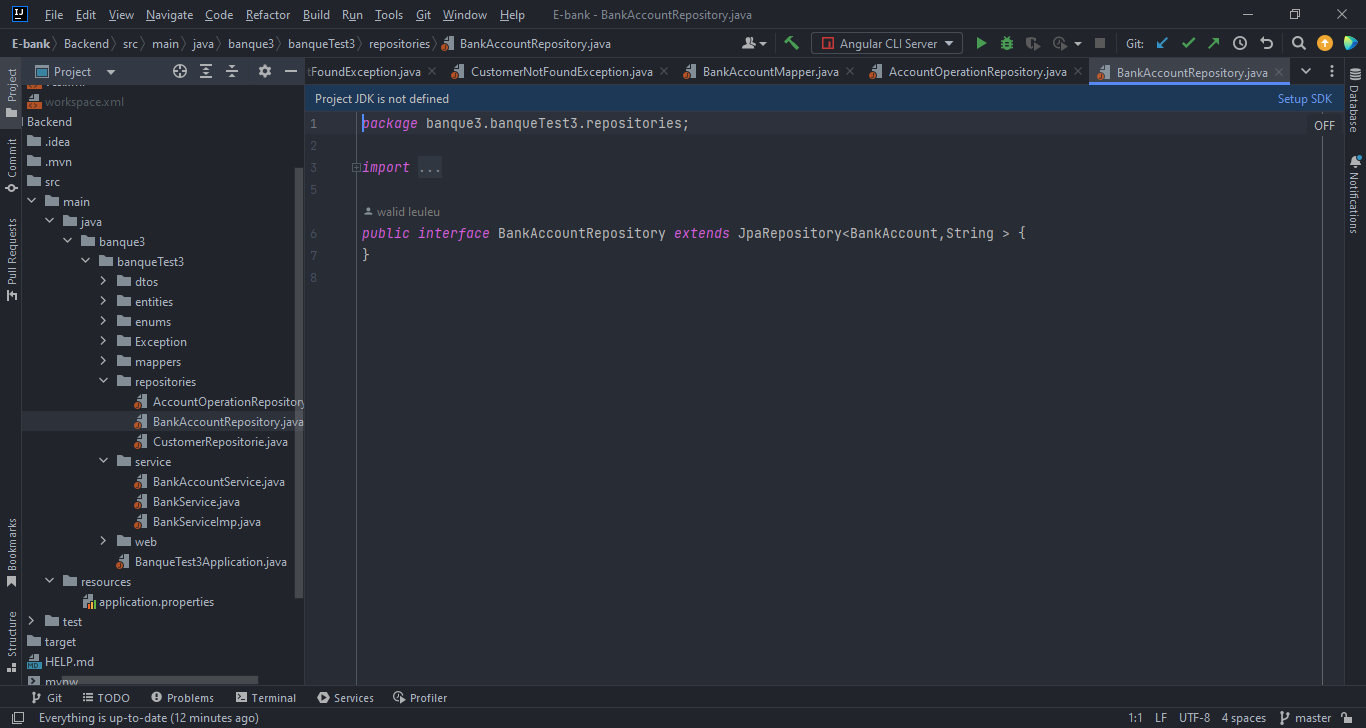
fromCurrentAccountDto(CurrentBankAccountDto currentBankAccountDto) : Cette méthode effectue l'opération inverse de la méthode précédente. Elle convertit un objet CurrentBankAccountDto en un objet CurrentAccount, ce qui permet de restaurer les données des comptes courants à partir des couches supérieures de l'application.



Cette interface est utilisée pour interagir avec la table des opérations de compte dans la base de données.

Elle étend **JpaRepository<AccountOperation, Long>**, ce qui signifie qu'elle est paramétrée pour travailler avec l'entité **AccountOperation** et utilise un identifiant de type **Long**.

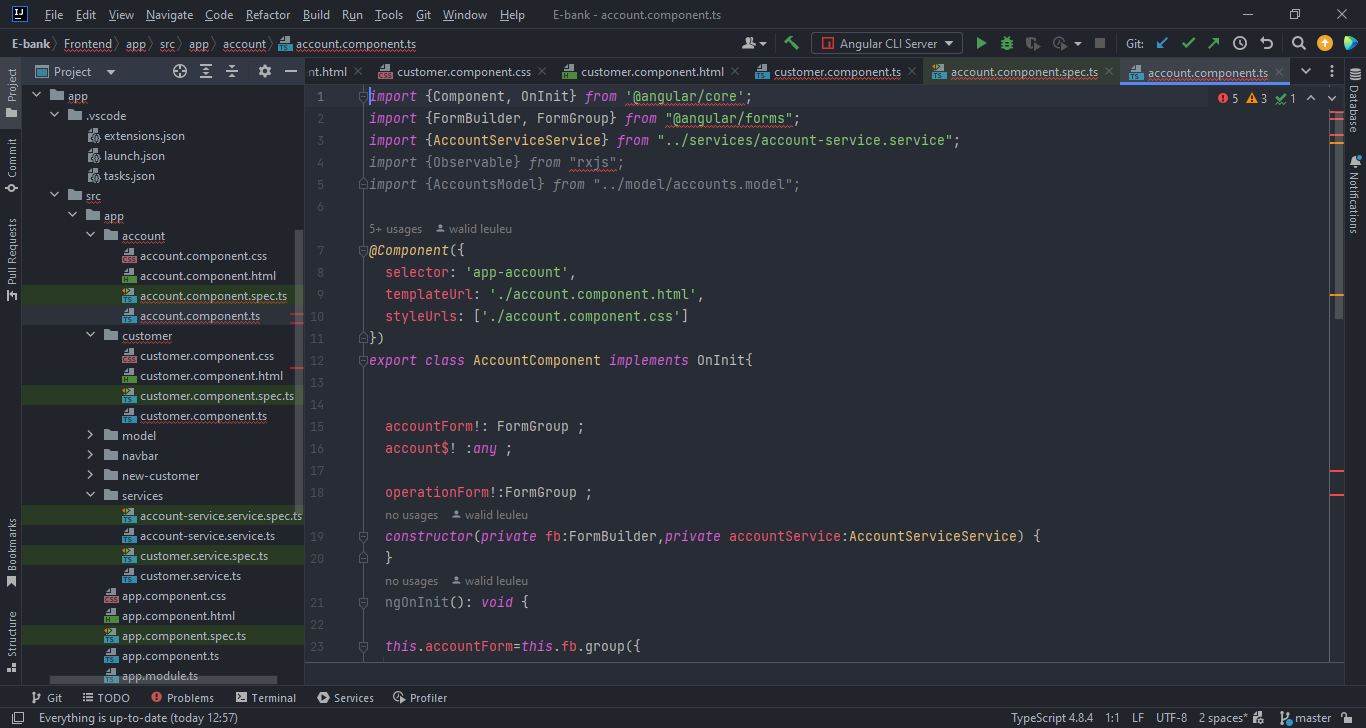
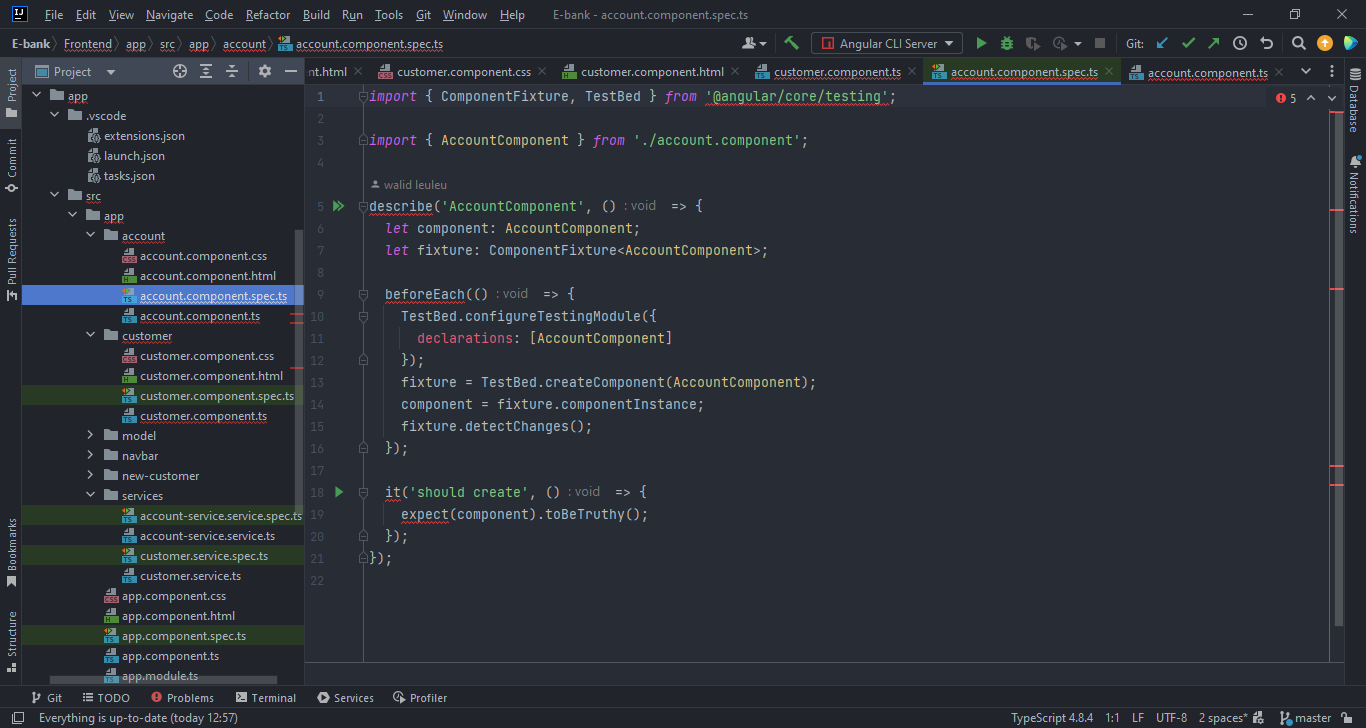
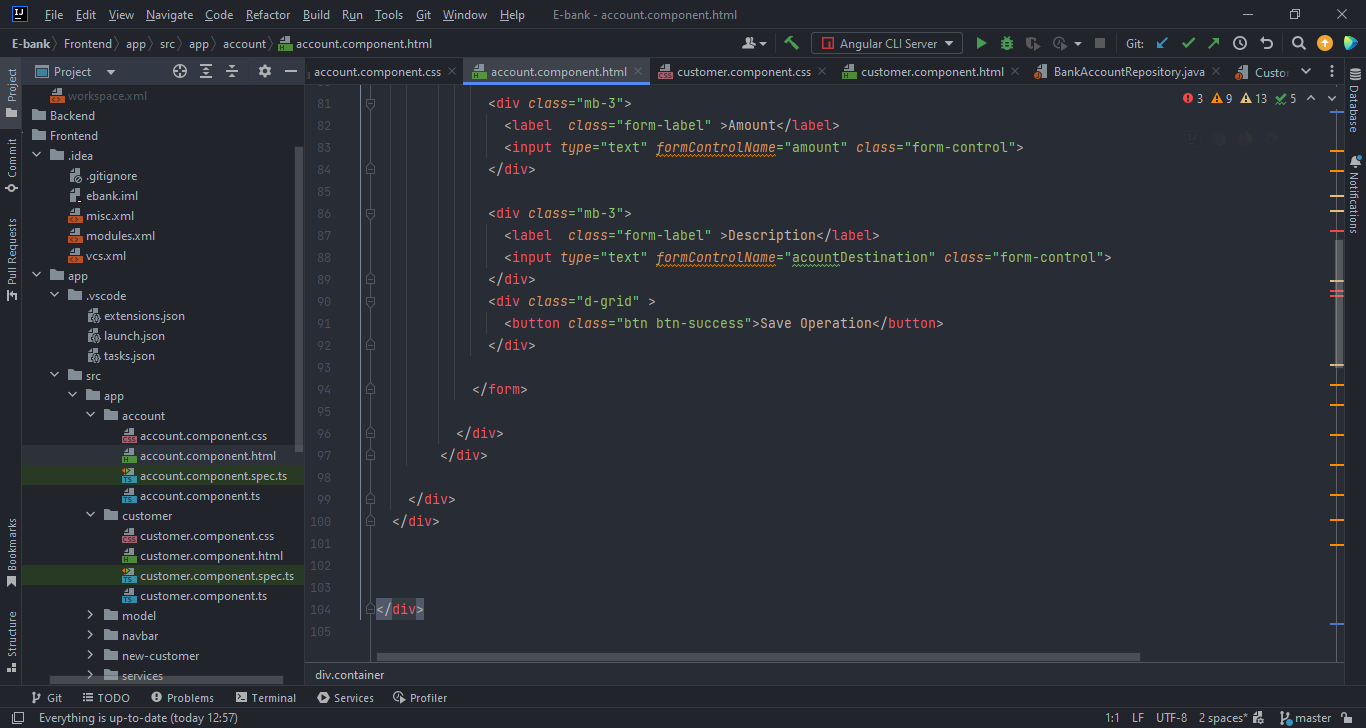
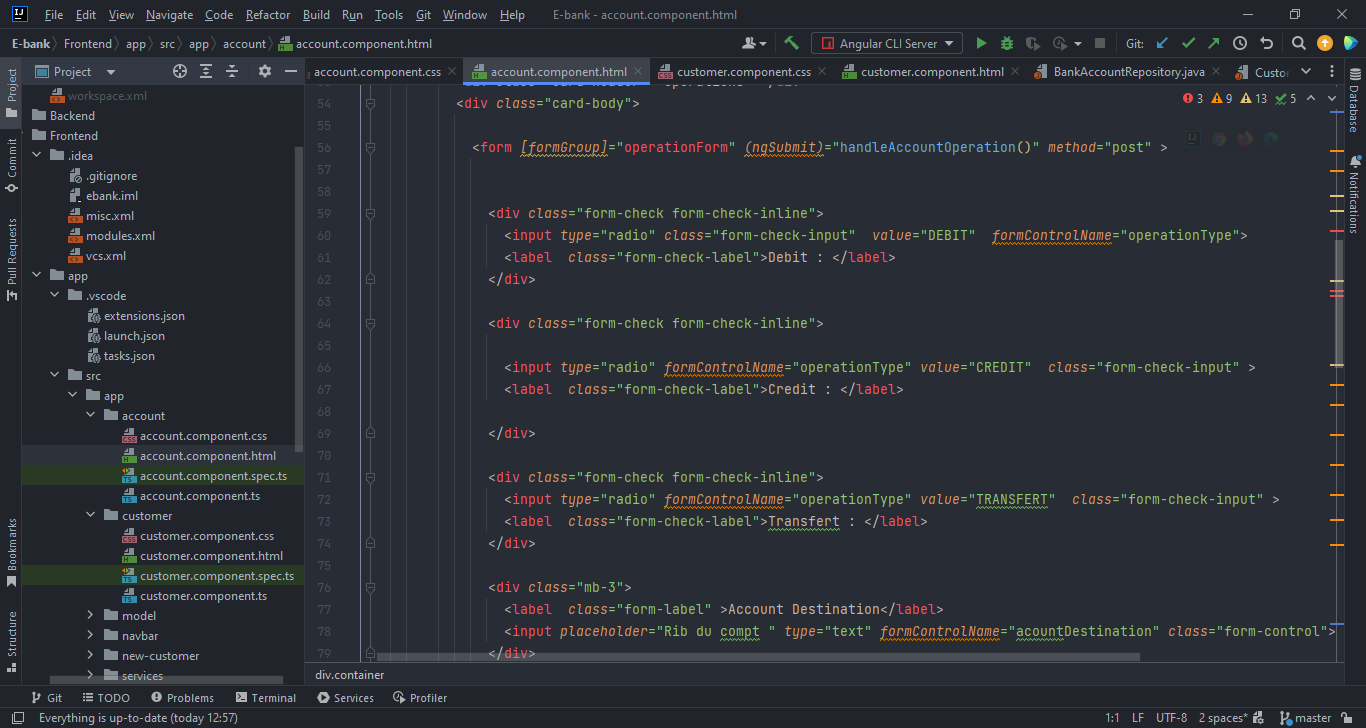
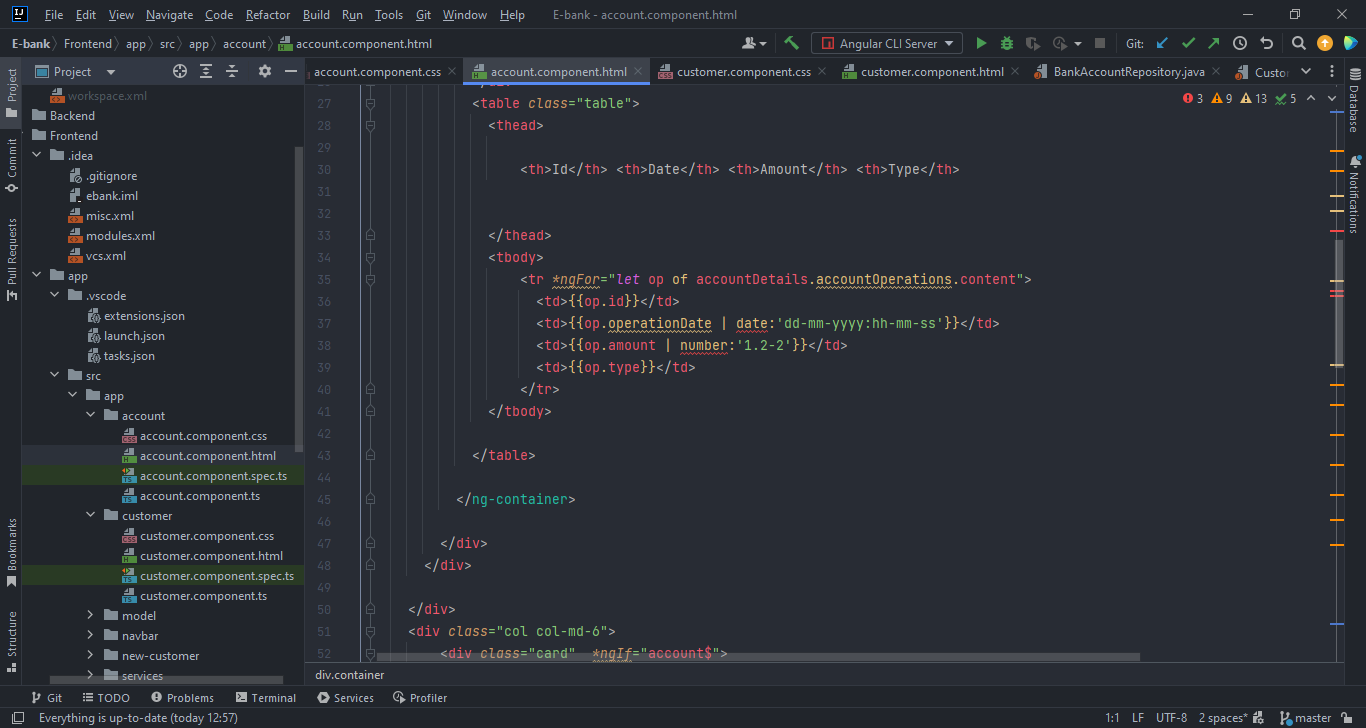
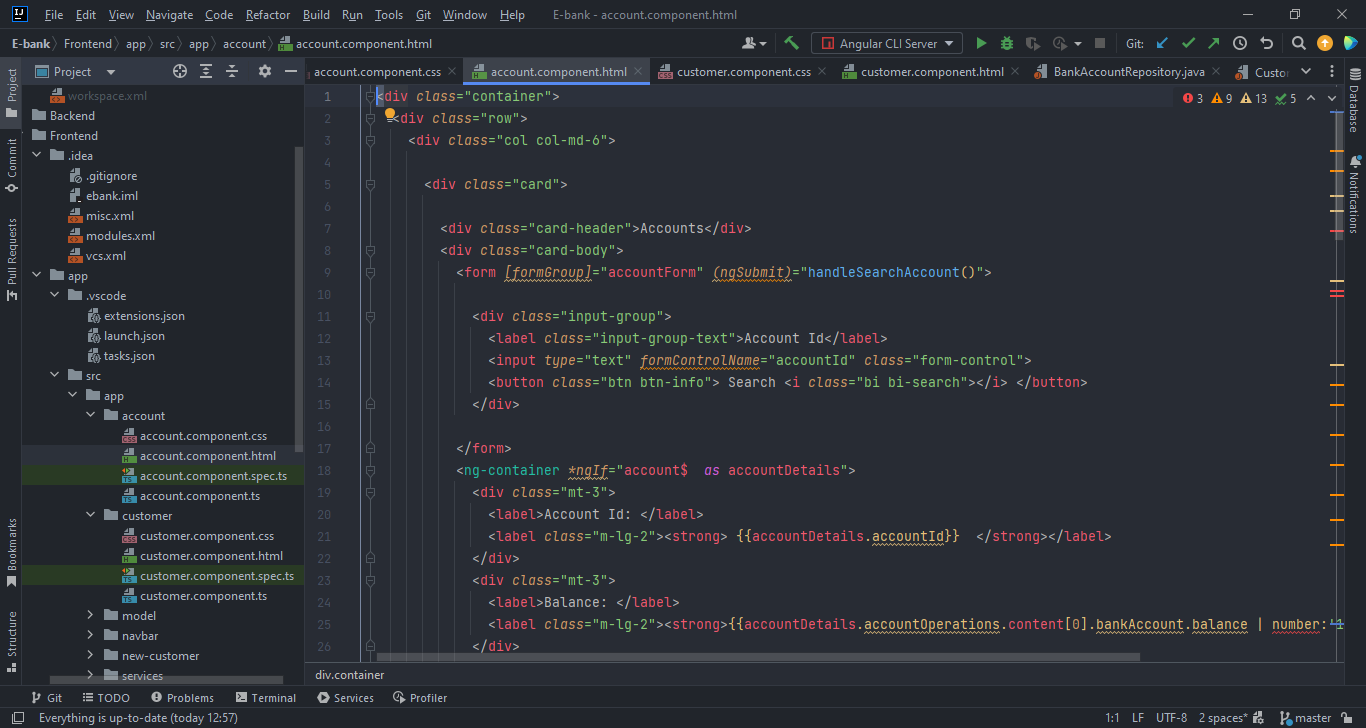
Elle définit des méthodes pour rechercher des opérations de compte en fonction de l'identifiant du compte (**findByBankAccountId(String accountId)**) et pour paginer les résultats (**findByBankAccountId(String accountId, Pageable pageable)**).

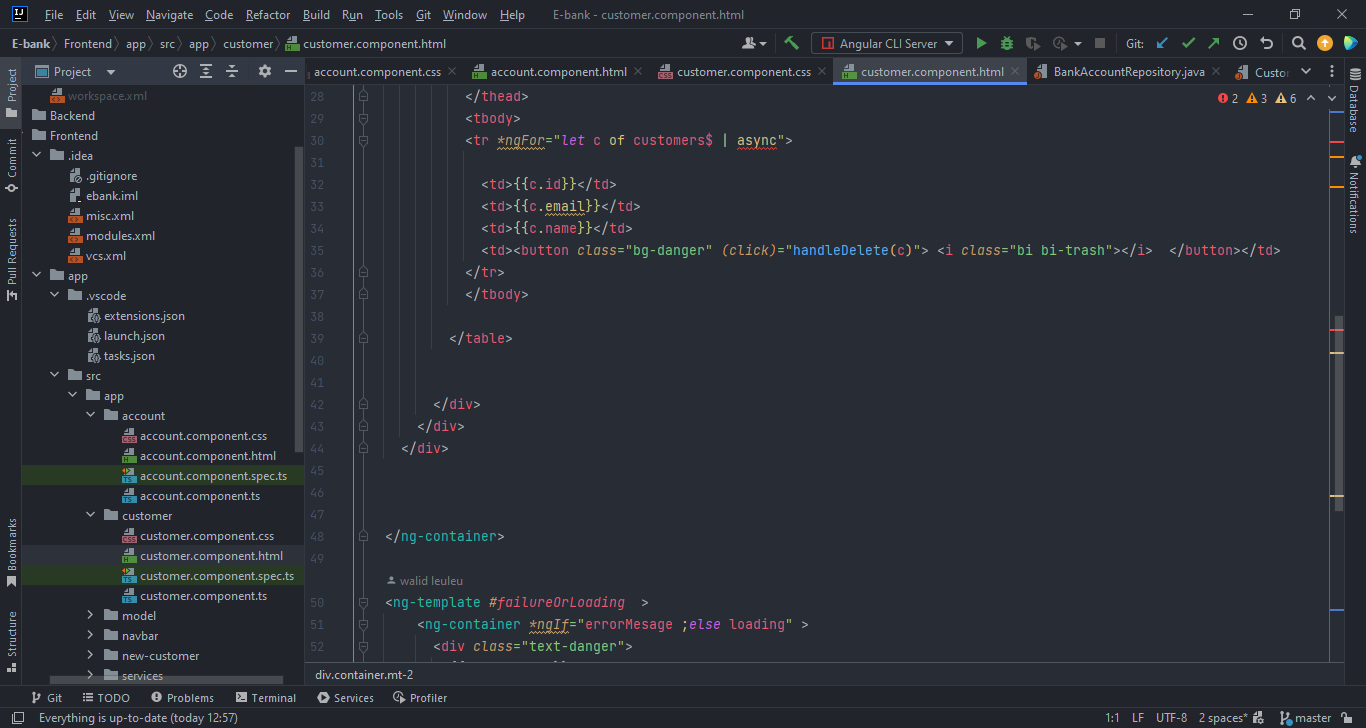
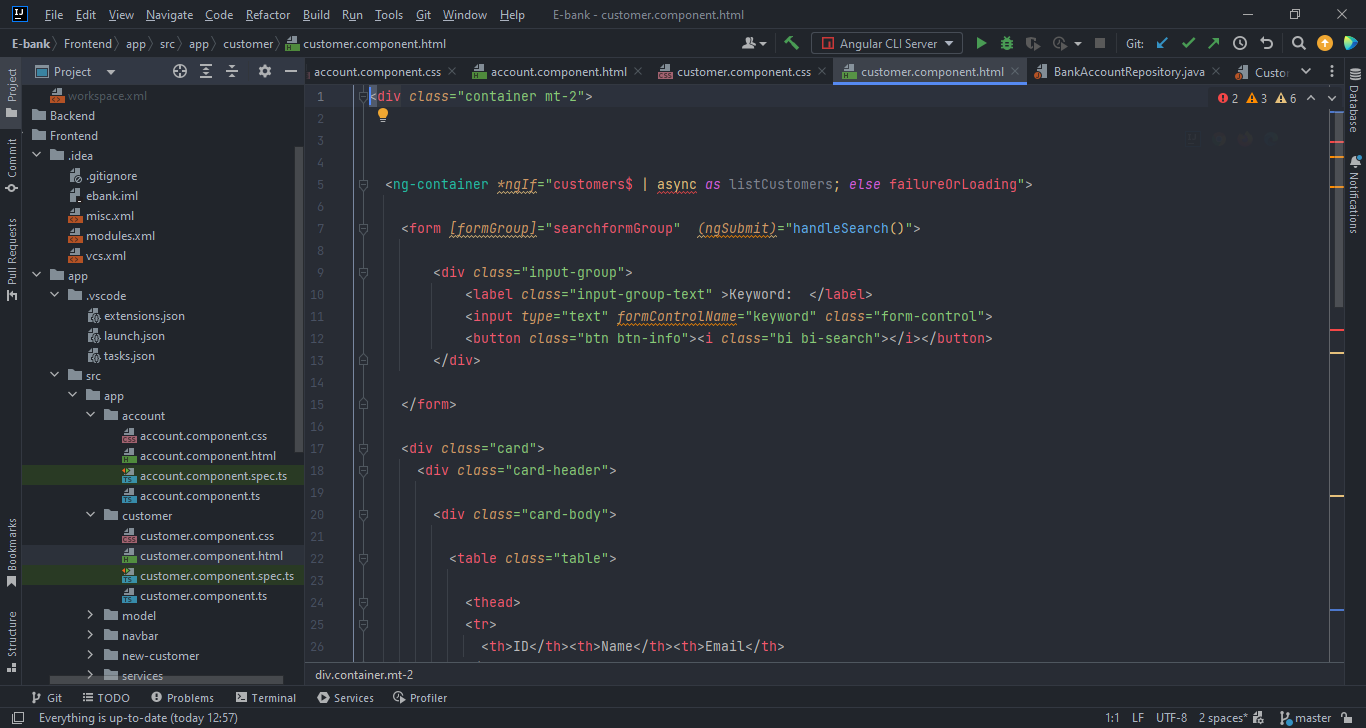


Cette interface est utilisée pour interagir avec la table des comptes bancaires dans la base de données.

Elle étend **JpaRepository<BankAccount, String>**, ce qui signifie qu'elle est paramétrée pour travailler avec l'entité **BankAccount** et utilise un identifiant de type **String**.

# II-frontend :



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

Description générée automatiquement