École Normale Supérieure de l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

ENSET

ا لمدرسة العليا لأساتذة التعليم التقني المحمدية

جامعة الحسن الثاني بالدار البيضاء

Département de Mathématique et Informatique

II^{ème} année

Filière :

« Ingénierie Informatique : Big Data et Cloud Computing »

II-BDCC

JEE :Bank-Backend

Rapport

KAFANDO Tounwendsida Bertrand

Partie 1 : Objectifs

On souhaite créer une application Web basée sur Spring et Angular qui permet de gérer des comptes bancaires. Chaque compte appartient à un client il existe deux types de comptes : Courant et Epargnes. Chaque Compte peut subir des opérations de types Débit ou crédit.

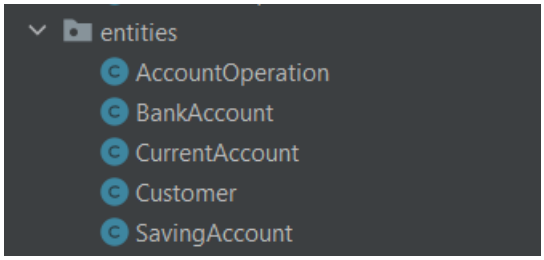
L'application se compose des couches suivantes :

- Couche DAO (Entités JPA et Repositories)
- Couche Service définissant les opérations suivantes :
  - Ajouter des comptes
  - Ajouter des clients
  - Effectuer un débit (Retrait)
  - Effectuer un crédit (Versement)
  - Effectuer un virement
  - Consulter un compte
- La couche DTO
- Mappers (DTO <=>Entities)
- La couche Web (Rest Controllers)

Architectures :

Partie 2 : Réalisation

1-Creation des entités



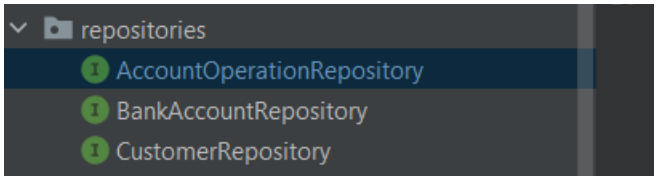
Pour l'héritage j'utilise, j'utilise la stratégie Single Table

```
2 inheritors   bertrand *
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "TYPE",length = 4, discriminatorType= DiscriminatorType.STRING)
@Data @NoArgsConstructor @AllArgsConstructor
public  class BankAccount {
```

```
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@DiscriminatorValue("CA")
public class CurrentAccount extends  BankAccount{
    private double overDraft;
}
```

```
17 usages   bertrand
@Entity
@Data @AllArgsConstructor @NoArgsConstructor
@DiscriminatorValue("SA")
public class SavingAccount extends BankAccount {
    private double interestRate;
}
```

2-Création des JPA repositories



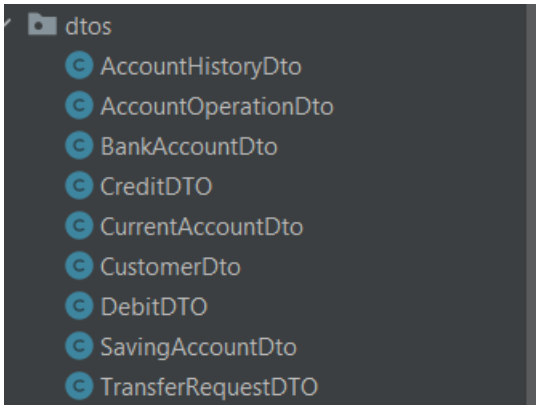
3-Couche service :

Interface service :

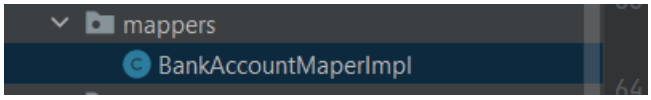
```
public interface BankAccountService {
3 usages  1 implementation  bertrand
CustomerDto saveCustomer(CustomerDto customer);
    1 implementation  bertrand
    CustomerDto updateCustomer(CustomerDto customer);
    1 usage  1 implementation  bertrand
    void deleteCustomer(Long customerId);
    1 usage  1 implementation  bertrand
    SavingAccountDto saveSavingBankAccount(double initialBalance, double interestRate, Long customerId) throws CustomerNotFoundException;
1 usage  1 implementation  bertrand
CurrentAccountDto saveCurrentBankAccount(double initialBalance, double overDraft, Long customerId) throws CustomerNotFoundException;
    2 usages  1 implementation  bertrand
    List<CustomerDto>listCustomers();
    1 usage  1 implementation  bertrand
    CustomerDto getcustomer(Long custometId) throws CustomerNotFoundException;
    1 usage  1 implementation  bertrand
    BankAccountDto getBankAccount(String accountId) throws BankAccountNotFoundException;
3 usages  1 implementation  bertrand
void debit(String accountId, double amount, String description) throws BankAccountNotFoundException, BalanceNotSufficientException;
3 usages  1 implementation  bertrand
void credit(String accountId, double amount, String description) throws BankAccountNotFoundException;
1 usage  1 implementation  bertrand
void transfer(String accountIdsource,String accountIdDestination, double amount) throws BankAccountNotFoundException, BalanceNotSufficientException;
    2 usages  1 implementation  bertrand
    List<BankAccountDto> bankAccountList();
    1 usage  1 implementation  bertrand
    List<AccountOperationDto>accountsHistory(String accountId);
    1 usage  1 implementation  bertrand
    AccountHistoryDto getaccountsHistory(String accountId, int page, int size) throws BankAccountNotFoundException;
    1 usage  1 implementation  bertrand
    List<CustomerDto> searchCustomers(String s);
}
```

Toutes ces fonctions sont implémentées dans une classe.

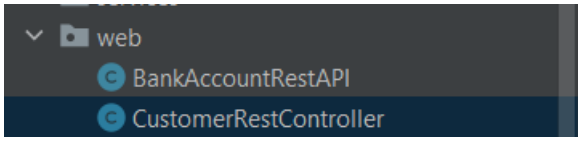
4-DTOs



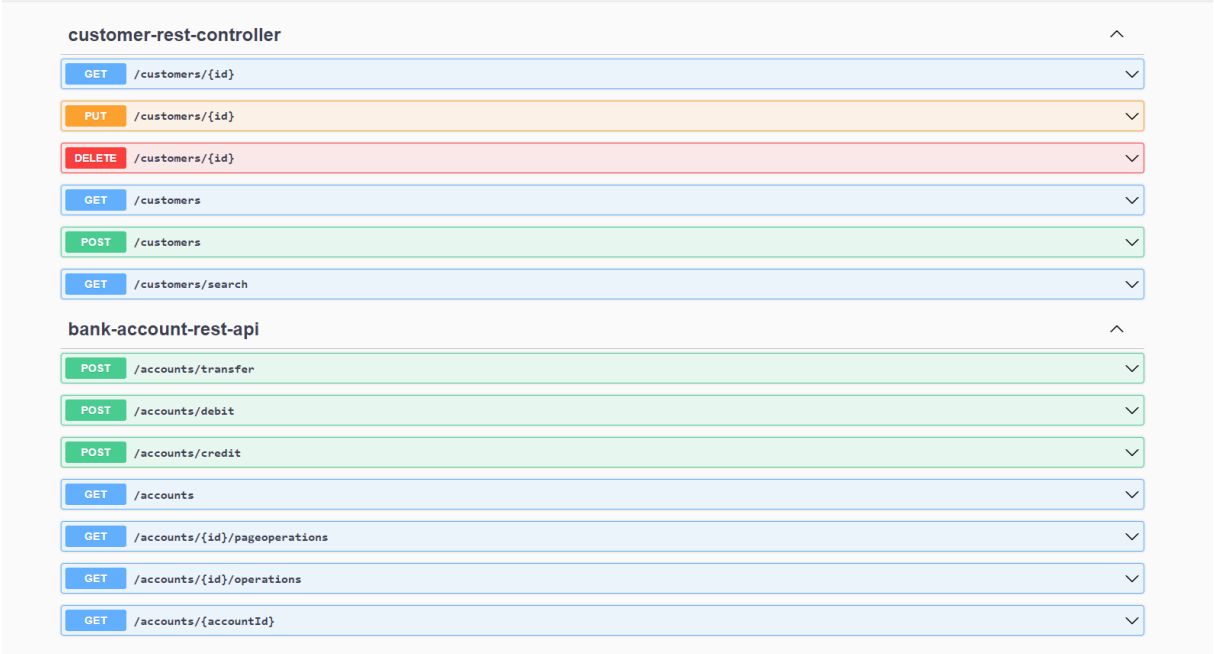
5-Mappers



6-Restful api

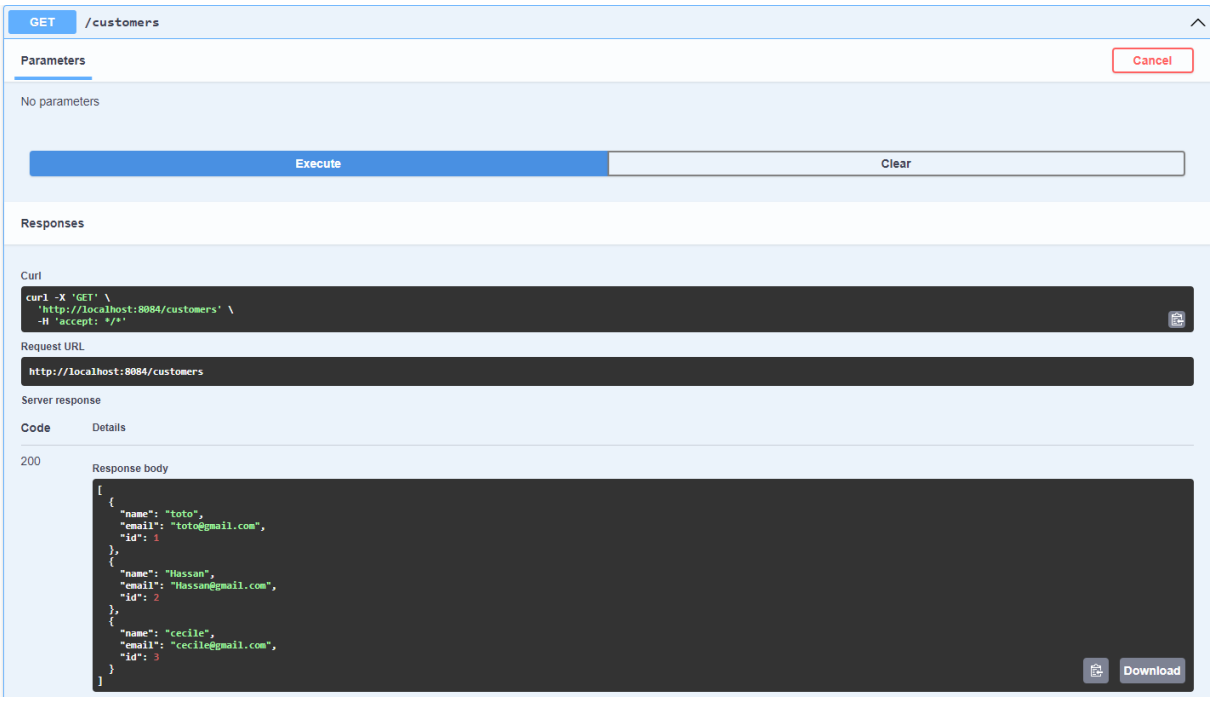


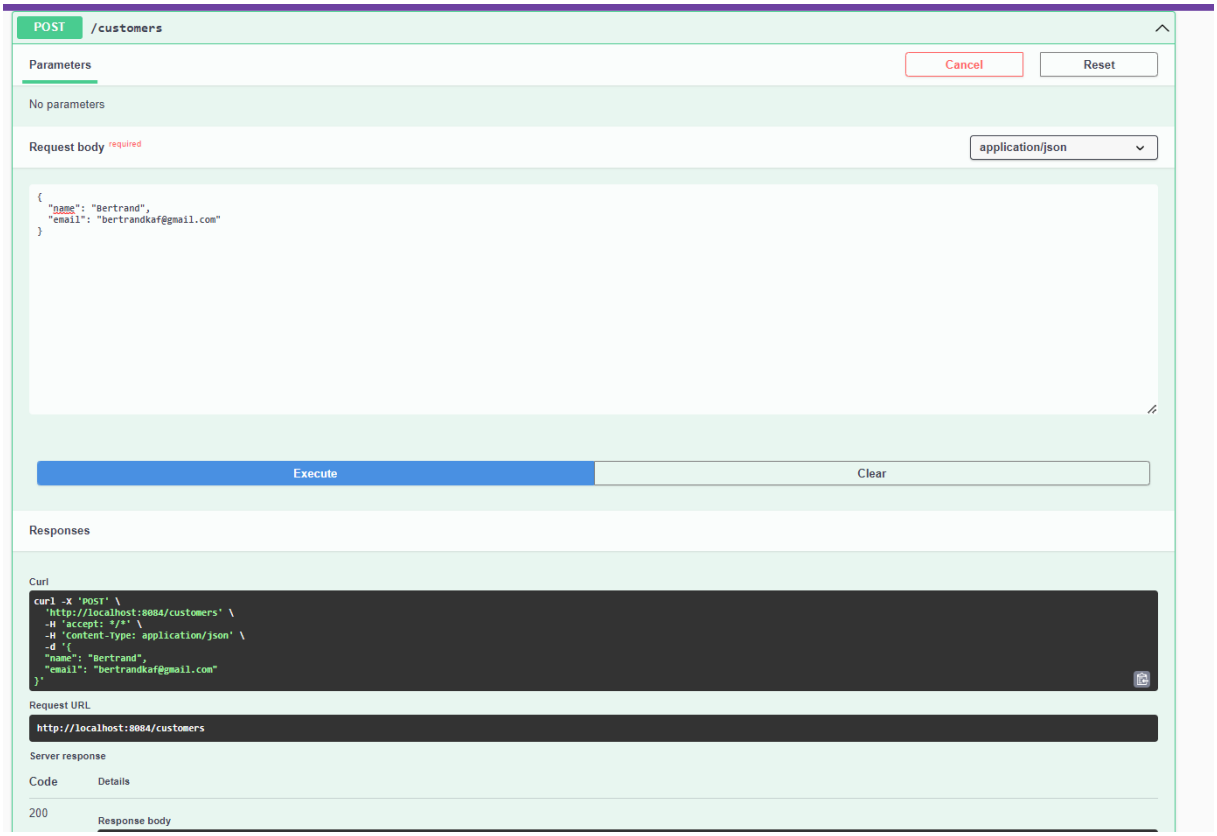
7- fonctionnalités Swagger



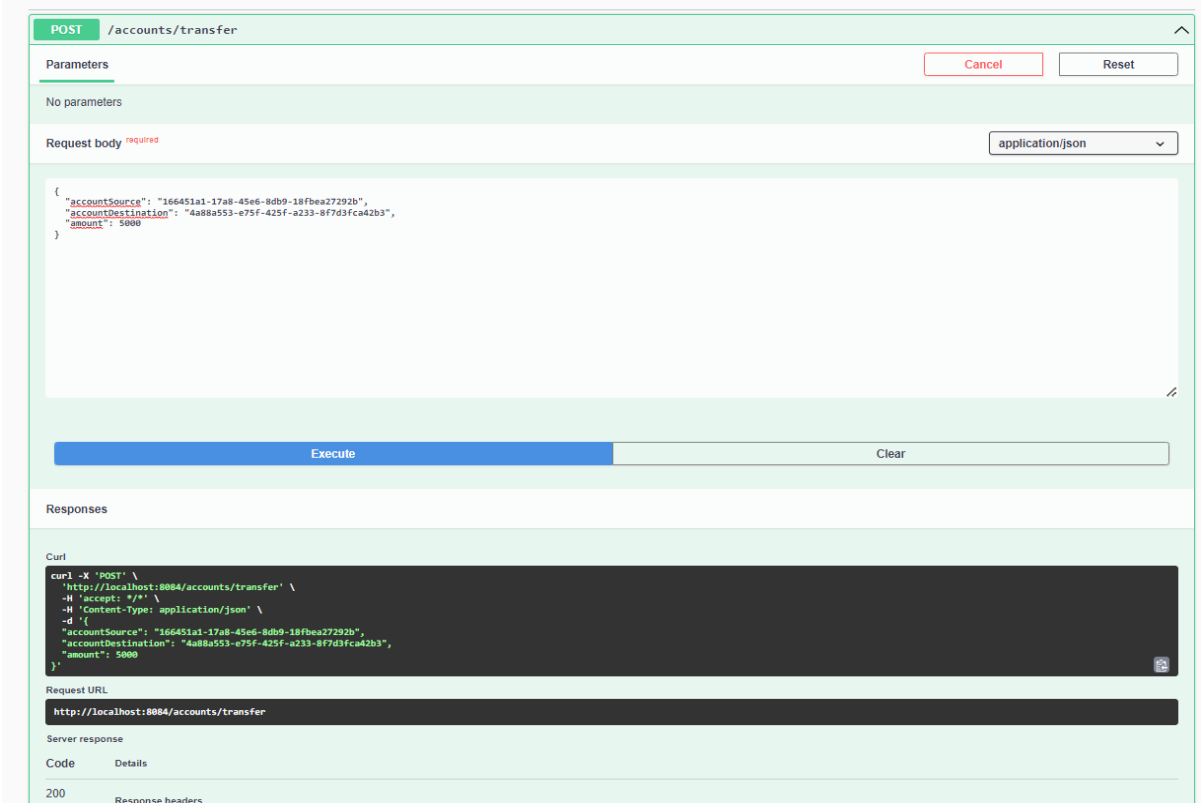
8-Test des fonctions

.....




POST /customers
Parameters
Cancel
Reset
No parameters
Request body required
application/json
{
"name": "Bertrand",
"email": "bertrandkaf@gmail.com"
}
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://localhost:8084/customers' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
"name": "Bertrand",
"email": "bertrandkaf@gmail.com"
}'
Request URL
http://localhost:8084/customers
Server response
Code
Details
200
Response body

Transfert :


POST /accounts/transfer
Parameters
Cancel
Reset
No parameters
Request body required
application/json
{
"accountSource": "166451a1-17a8-45e6-8db9-18fbea27292b",
"accountDestination": "4a88a553-e75f-425f-a233-8f7d3fca42b3",
"amount": 5000
}
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://localhost:8084/accounts/transfer' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
"accountSource": "166451a1-17a8-45e6-8db9-18fbea27292b",
"accountDestination": "4a88a553-e75f-425f-a233-8f7d3fca42b3",
"amount": 5000
}'
Request URL
http://localhost:8084/accounts/transfer
Server response
Code
Details
200
Response headers