Ecole 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

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Compte rendu

- Projet Digital Banking –

« Architecture JEE »

SPRING ET ANGULAR

Réalisé par :

Ikram Berradi

Encadré par :

Mohamed YOUSSFI

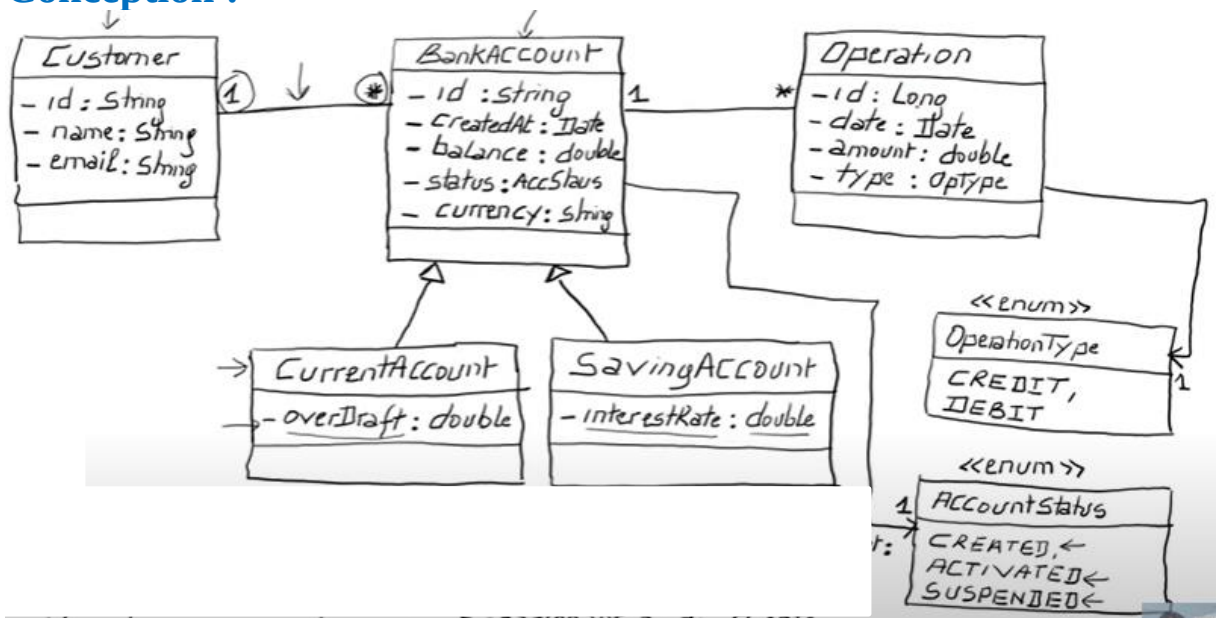
GLSID II-2022/2021

❖ Chair de charge

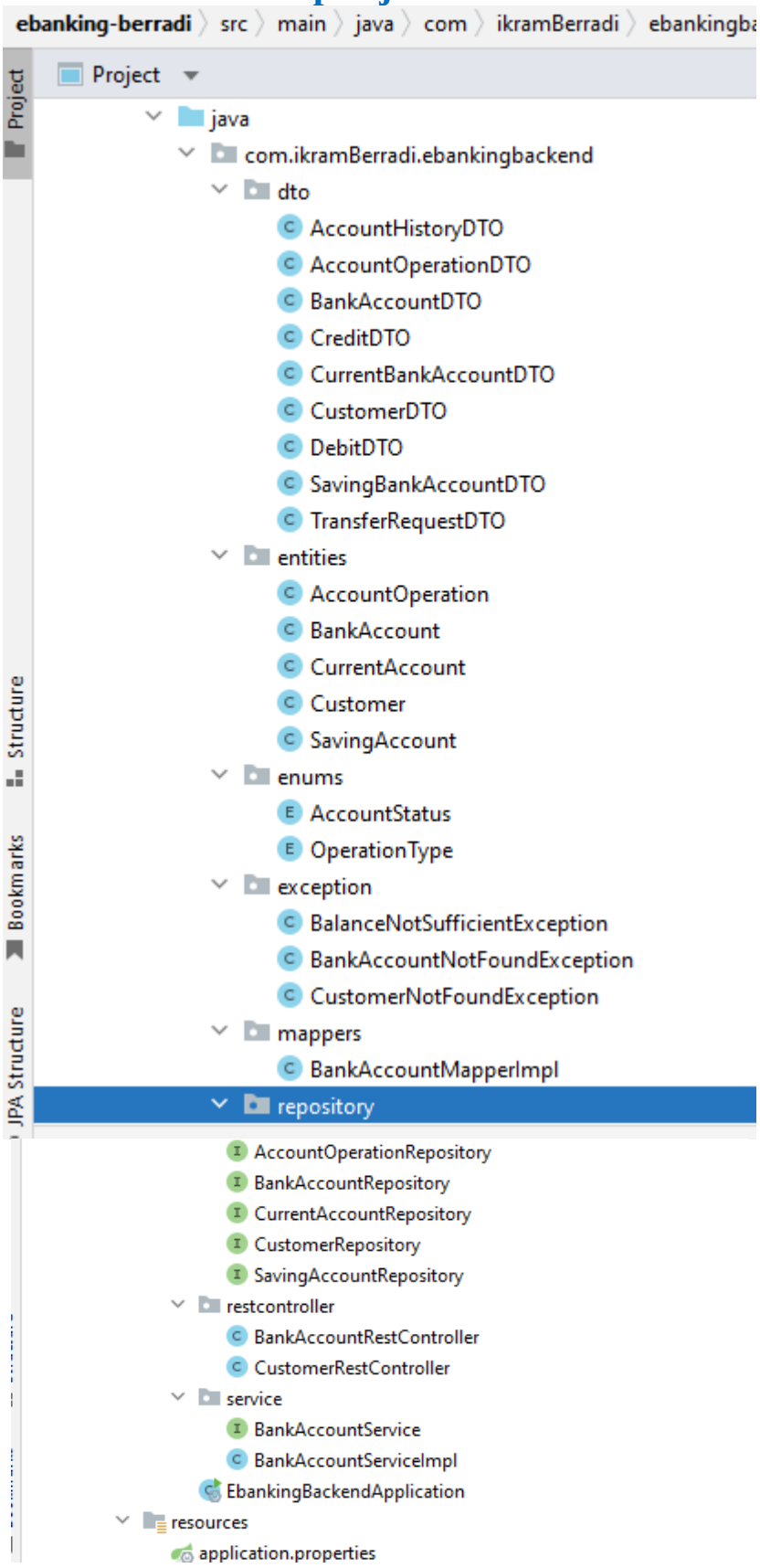
On souhaite créer une application Web basée sur Spring et Angular qui permet de gérer des comptes bancaires. Chaque compte appartient à un client il existe deux types de comptes : Courant et Epargnes. Chaque Compte peut subir des opérations de types Débit ou crédit.
L'application se compose des couches 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)
- Couche sécurité (Spring Security avec JWT)

➢ Conception :



➢ Structure finale de projet



➢ Application.properties

application.properties | CurrentAccount.java | SavingAccount.java | AccountOperationRepository.java | BankAccountRepository.java

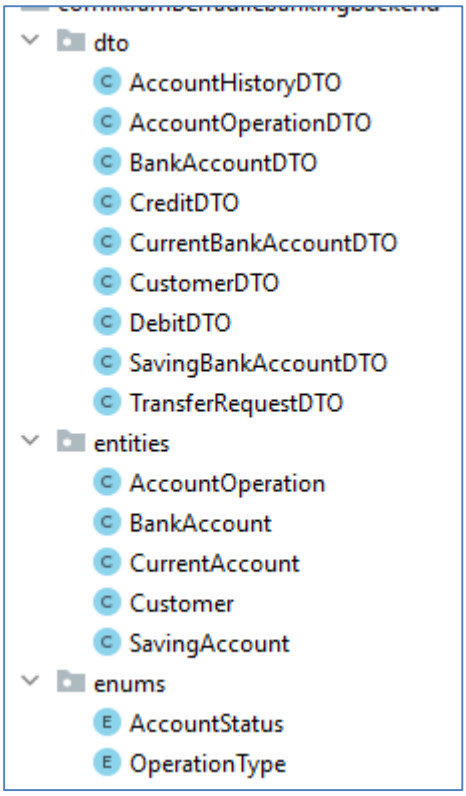
```
#spring.datasource.url=jdbc:h2:mem:bank
#spring.h2.console.enabled=true
server.port=8086
spring.datasource.url=jdbc:mysql://localhost:3306/E-BANK?createDatabaseIfNotExist=true&useSSL=false&serverTi
spring.datasource.username=root
spring.datasource.password=
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
```

➢ Application Spring Boot

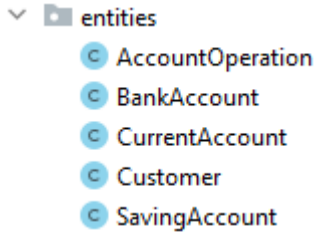
EbankingBackendApplication.java

```
@SpringBootApplication
public class EbankingBackendApplication {

    public static void main(String[] args) { SpringApplication.run(EbankingBackendApplication.class, args);

    //@Bean
    CommandLineRunner start(BankAccountService bankAccountService) {
        return args -> {
            Stream.of("ikram", "sara", "salma").forEach(name -> {
                CustomerDTO customer = new CustomerDTO();
                customer.setName(name);
                customer.setEmail(name+"@gmail.com");
                bankAccountService.saveCustomer(customer);
            });
            bankAccountService.listCustomers().forEach(customer -> {
                try {
                    bankAccountService.saveCurrentBankAccount( initialBalance: Math.random()*9000,  overDraft: 9000, c
                    bankAccountService.saveSavingBankAccount( initialBalance: Math.random()*120000,  interestRate: 5.5,

                } catch (CustomerNotFoundException e) {
                    e.printStackTrace();
                }
            });
```

➢ Couche DAO



- **Entités JPA :**



- # AccountOperation

```java
@Entity
public class AccountOperation {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private Date operationDate;
    private double amount;

    @Enumerated(EnumType.STRING)
    private OperationType type;
    @ManyToOne
    private BankAccount bankAccount;

    private String description;

    public AccountOperation(Long id, Date operationDate, double amount, OperationType type, BankAccoun
            String description) {
        this.id = id;
        this.operationDate = operationDate;
        this.amount = amount;
        this.type = type;
        this.bankAccount = bankAccount;
        this.description = description;
```

BankAccount

BankAccount.java

```java
package com.ikramBerradi.ebankingbackend.entities;

import ...

@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "TYPE", length = 4, discriminatorType = DiscriminatorType.STRIN
public class BankAccount {

    @Id
    private String id;
    private double balance;
    private Date createdAt;

    @Enumerated(EnumType.STRING)
    private AccountStatus status;
    @ManyToOne
    private Customer customer;
    @OneToMany(mappedBy = "bankAccount", fetch = FetchType.LAZY)
    private List<AccountOperation> accountOperations;


    public BankAccount(String id, double balance, Date createdAt, AccountStatus status, Cus
            List<AccountOperation> accountOperations) {
        super();
```

Current Account

CurrentAccount.java

```java
import java.util.List;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

import com.ikramBerradi.ebankingbackend.enums.AccountStatus;

@Entity
@DiscriminatorValue("CA")
public class CurrentAccount extends BankAccount {

    private double overDraft; //decouvert
    public CurrentAccount(String id, double balance, Date createdAt, AccountSt
            List<AccountOperation> accountOperations, double overDraft) {
        super(id, balance, createdAt, status, customer, accountOperations);
        this.overDraft = overDraft;
    }
    public CurrentAccount() { super(); }
    public CurrentAccount(String id, double balance, Date createdAt, AccountSt
            List<AccountOperation> accountOperations) {
        super(id, balance, createdAt, status, customer, accountOperations);
    }


    public double getOverDraft() { return overDraft; }
```

Customer

Customer.java

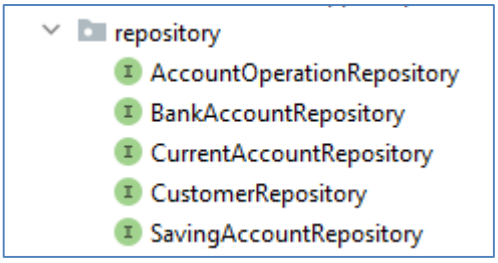
```java
import javax.persistence.OneToMany;

import com.fasterxml.jackson.annotation.JsonProperty;

@Entity
public class Customer {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;

    @OneToMany(mappedBy = "customer")
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<BankAccount> bankAccounts;


    public Customer(Long id, String name, String email, List<BankAccount>
        super();
        this.id = id;
        this.name = name;
        this.email = email;
        this.bankAccounts = bankAccounts;
    }
```

- **SavingAccount**

SavingAccount.java

```java
package com.ikramBerradi.ebankingbackend.entities;

import ...

@Entity
@DiscriminatorValue("SA")
public class SavingAccount extends BankAccount {

    private double interestRate; //taux d'interet



    public SavingAccount(String id, double balance, Date createdAt, AccountStatus status, Customer
            List<AccountOperation> accountOperations, double interestRate) {
        super(id, balance, createdAt, status, customer, accountOperations);
        this.interestRate = interestRate;
    }
    public SavingAccount() {
        super();

    }
    public SavingAccount(String id, double balance, Date createdAt, AccountStatus status, Customer
            List<AccountOperation> accountOperations) {
        super(id, balance, createdAt, status, customer, accountOperations);

```

➢ Interfaces DAO basées sur Spring Data



AccountOperationRepository.java

```java
package com.ikramBerradi.ebankingbackend.repository;

import ...

public interface AccountOperationRepository extends JpaRepository<AccountOperation, Long> {
    List<AccountOperation> findByBankAccountId(String accountId);

    Page<AccountOperation> findByBankAccountIdOrderByOperationDateDesc(String accountId, Pageable pageable
}
```

BankAccountRepository.java

```java
package com.ikramBerradi.ebankingbackend.repository;

import ...

public interface BankAccountRepository extends JpaRepository<BankAccount, String> {

}
```

CurrentAccountRepository.java

```java
package com.ikramBerradi.ebankingbackend.repository;

import ...

public interface CurrentAccountRepository extends JpaRepository<CurrentAccount, String> {

}
```

CustomerRepository.java

```java
package com.ikramBerradi.ebankingbackend.repository;

import ...

public interface CustomerRepository extends JpaRepository<Customer, Long> {

    @Query("select c from Customer c where c.name like :kw")
    List<Customer> findByNameContains(@Param("kw") String keywords);
}
```

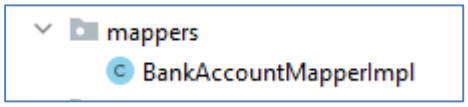
SavingAccountRepository.java

```java
package com.ikramBerradi.ebankingbackend.repository;

import ...

public interface SavingAccountRepository extends JpaRepository<SavingAccount, String> {

}
```

➢ Test de la couche DAO

id	email	name
1	ikram@gmail.com	ikram
2	omayma@gmail.com	omayma
3	sara@gmail.com	sara

➢ Mappers



BankAccountMapperImpl.java

```java
package com.ikramBerradi.ebankingbackend.mappers;

import ...

@Service
public class BankAccountMapperImpl {

    public CustomerDTO fromCustomer(Customer customer) {
        CustomerDTO customerDTO = new CustomerDTO();
        BeanUtils.copyProperties(customer, customerDTO);
        return customerDTO;
    }

    public Customer fromCustomerDTO(CustomerDTO customerDTO) {
        Customer customer = new Customer();
        BeanUtils.copyProperties(customerDTO, customer);
        return customer;
    }

    public SavingBankAccountDTO fromSavingBankAccount(SavingAccount savingAccount) {
        SavingBankAccountDTO savingBankAccountDTO = new SavingBankAccountDTO();
        BeanUtils.copyProperties(savingAccount, savingBankAccountDTO);
        savingBankAccountDTO.setCustomerDTO(fromCustomer(savingAccount.getCustomer()));
        savingBankAccountDTO.setType(savingAccount.getClass().getSimpleName());
        return savingBankAccountDTO;
```

➢ La couche métier

L’interface BanckAccountservice

BankAccountService.java

```
1  package com.ikramBerradi.ebankingbackend.service;
2
3  import ...
14
15 public interface BankAccountService {
16
17     CustomerDTO saveCustomer(CustomerDTO customerDTO);
18
19     CustomerDTO updateCustomer(CustomerDTO customerDTO);
20
21     List<CustomerDTO> listCustomers();
22
23     void deleteCustomer(Long customerId);
24
25     CustomerDTO getCustomer(Long customerId) throws CustomerNotFoundException;
26
27     CurrentBankAccountDTO saveCurrentBankAccount(double initialBalance, double overDraft, L
28
29     SavingBankAccountDTO saveSavingBankAccount(double initialBalance, double interestRate, 
30
31     BankAccountDTO getBankAccount(String accountId) throws BankAccountNotFoundException;
32
33     void debit(String accountId, double amount, String description) throws BankAccountNotFo
34
35     void credit(String accountId, double amount, String description) throws BankAccountNotF
```

➢ Une implémentation de l'interface

BankAccountServiceImpl.java

```java
package com.ikramBerradi.ebankingbackend.service;

import ...

@Transactional
@Service
public class BankAccountServiceImpl implements BankAccountService {

    Logger log = LoggerFactory.getLogger(this.getClass().getName());

    private CustomerRepository customerRepository;

    private BankAccountRepository bankAccountRepository;

    private AccountOperationRepository accountOperationRepository;

    private BankAccountMapperImpl dtoMapper;



    public BankAccountServiceImpl(CustomerRepository customerRepository,
            BankAccountRepository bankAccountRepository,
            AccountOperationRepository accountOperationRepository,
            BankAccountMapperImpl  dtoMapper) {

```

➢ La base de données

➢ Bank_account

type	id	balance	created_at	status	over_draft	interest_rate	customer_id
CA	2dc4f2b3-834e-4d13-99d1-b47a272fba20	25890.257905260973	2022-05-21 13:20:31	CREATED	9000	NULL	3
SA	56869cc1-b246-4417-bb6c-38d2f68f2e98	50594.55782918615	2022-05-21 13:20:31	CREATED	NULL	5.5	1
CA	908582af-a5fd-4c02-8b7a-19c4d0ec5c64	66717.98416579244	2022-05-21 13:20:31	CREATED	9000	NULL	2
SA	d06d7265-78eb-42d6-a00a-9d4ec9c2a239	71208.99401059456	2022-05-21 13:20:31	CREATED	NULL	5.5	2
CA	db482f7f-565f-4cb8-bb14-1e4a4fc9e235	48620.90844868612	2022-05-21 13:20:31	CREATED	9000	NULL	1
SA	dde1a9ad-c99e-41df-a883-01452dda9a12	26376.07926238281	2022-05-21 13:20:31	CREATED	NULL	5.5	3

➢ Custommer

id	email	name
1	ikram@gmail.com	ikram
2	omayma@gmail.com	omayma
3	sara@gmail.com	sara

➢ Account_Operation

id	amount	description	operation_date	type	bank_account_id
1	11198.66614158947	*NULL*	2022-05-21 13:20:31	DEBIT	2dc4f2b3-834e-4d13-99d1-b47a272fba20
2	7802.191702622459	*NULL*	2022-05-21 13:20:31	DEBIT	2dc4f2b3-834e-4d13-99d1-b47a272fba20
3	7168.147362434192	*NULL*	2022-05-21 13:20:31	DEBIT	2dc4f2b3-834e-4d13-99d1-b47a272fba20
4	3869.578216631743	*NULL*	2022-05-21 13:20:31	DEBIT	2dc4f2b3-834e-4d13-99d1-b47a272fba20
5	9552.803959809515	*NULL*	2022-05-21 13:20:31	DEBIT	2dc4f2b3-834e-4d13-99d1-b47a272fba20
6	459.93907133907675	*NULL*	2022-05-21 13:20:31	DEBIT	2dc4f2b3-834e-4d13-99d1-b47a272fba20
7	4191.372657660584	*NULL*	2022-05-21 13:20:31	DEBIT	2dc4f2b3-834e-4d13-99d1-b47a272fba20
8	114.64090010062434	*NULL*	2022-05-21 13:20:31	CREDIT	2dc4f2b3-834e-4d13-99d1-b47a272fba20
9	3101.814258588716	*NULL*	2022-05-21 13:20:31	CREDIT	2dc4f2b3-834e-4d13-99d1-b47a272fba20
10	4514.860378893848	*NULL*	2022-05-21 13:20:31	CREDIT	2dc4f2b3-834e-4d13-99d1-b47a272fba20
11	756.6929154866959	*NULL*	2022-05-21 13:20:31	CREDIT	56869cc1-b246-4417-bb6c-38d2f68f2e98
12	3651.456029388622	*NULL*	2022-05-21 13:20:32	DEBIT	56869cc1-b246-4417-bb6c-38d2f68f2e98
13	985.2738737975	*NULL*	2022-05-21 13:20:32	CREDIT	56869cc1-b246-4417-bb6c-38d2f68f2e98
14	8489.903260496738	*NULL*	2022-05-21 13:20:32	CREDIT	56869cc1-b246-4417-bb6c-38d2f68f2e98