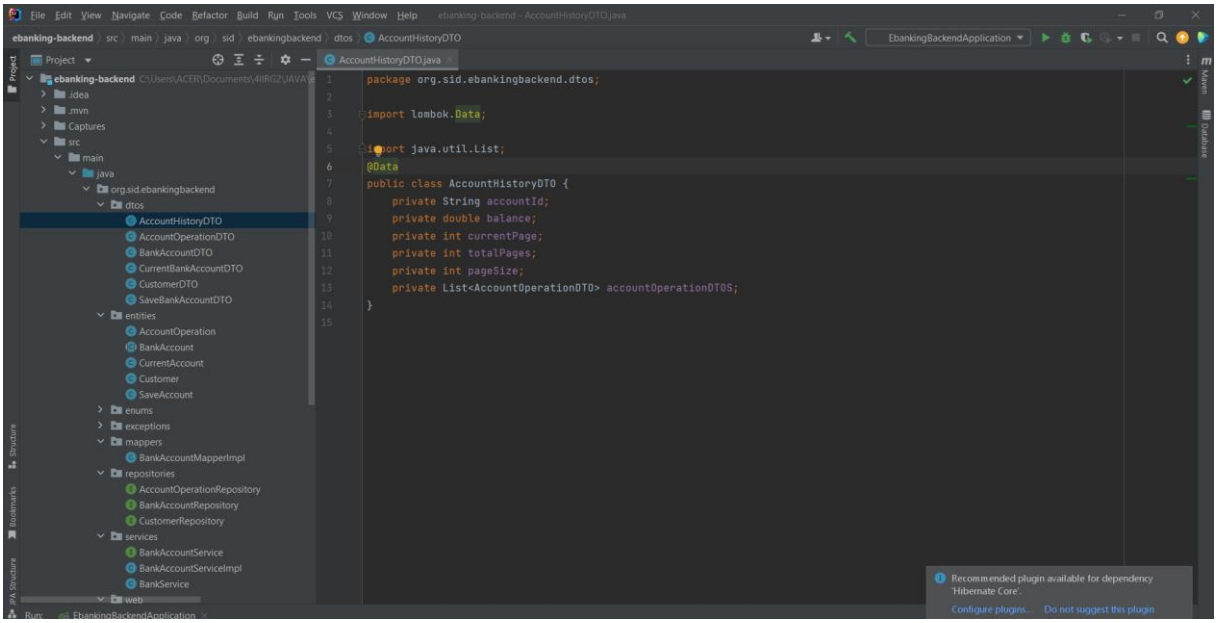
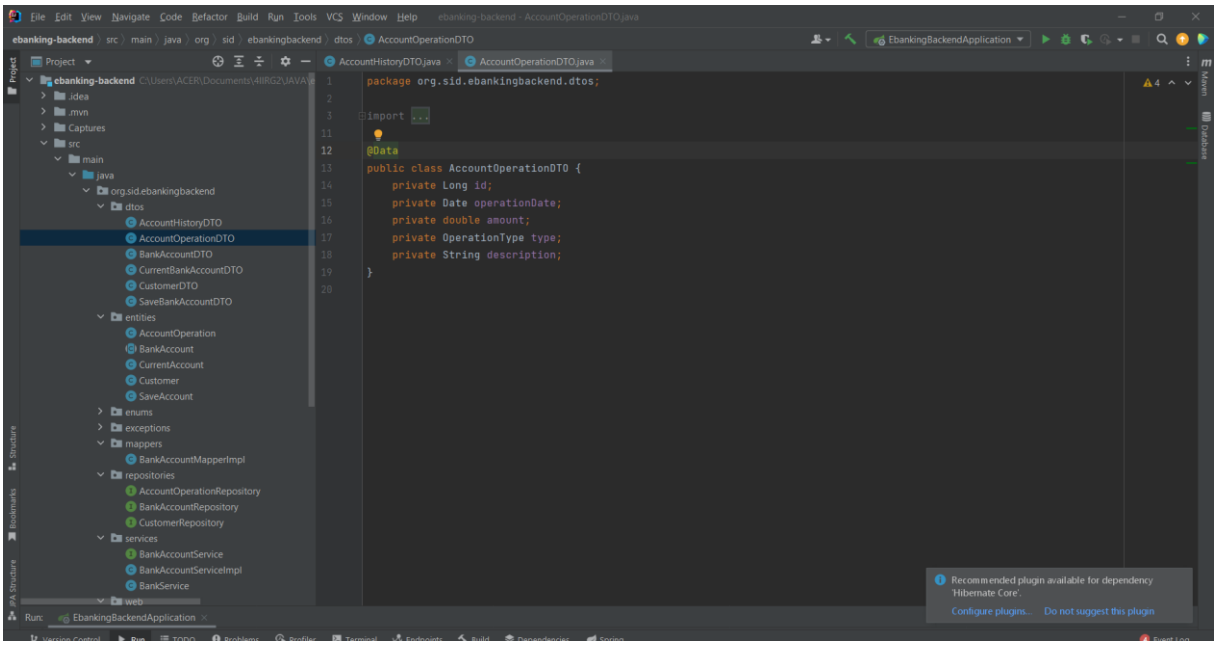
Spring Angular Use case Digital Banking

Package dtos

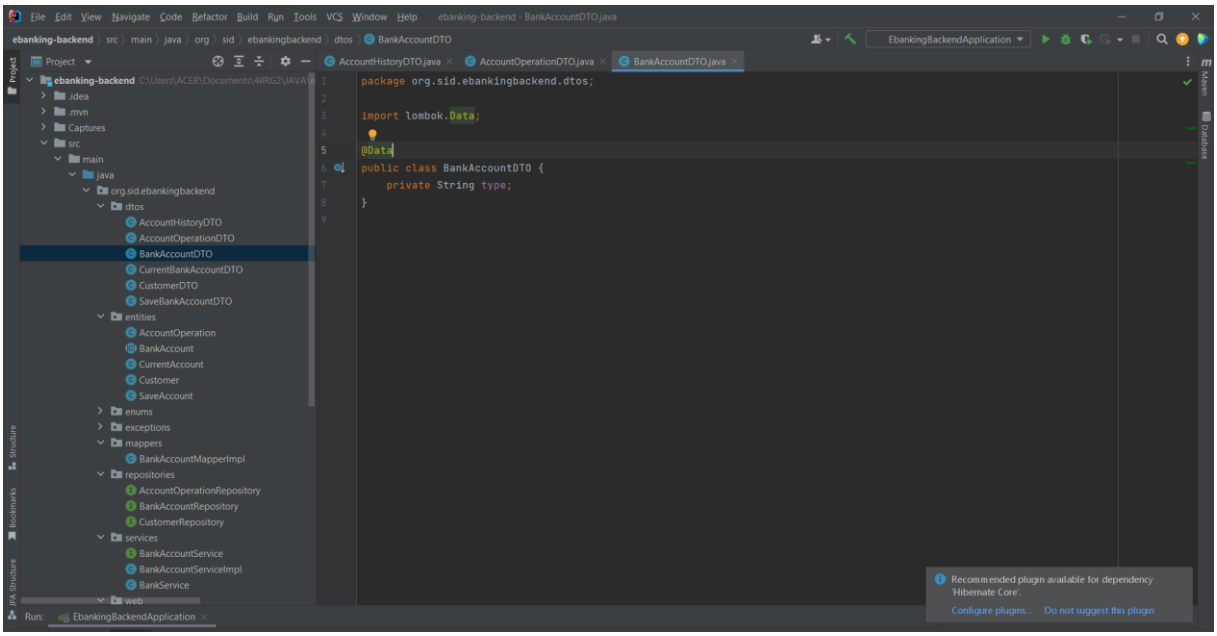
AccountHistoryDTO.java



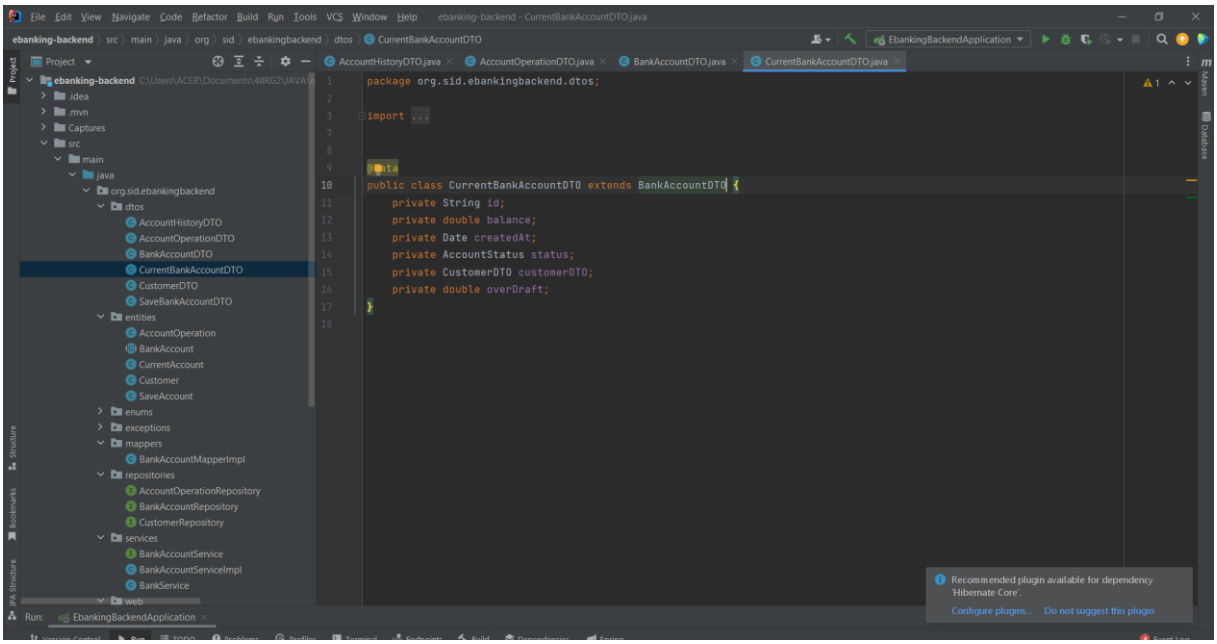
AccountOperationDTO.java



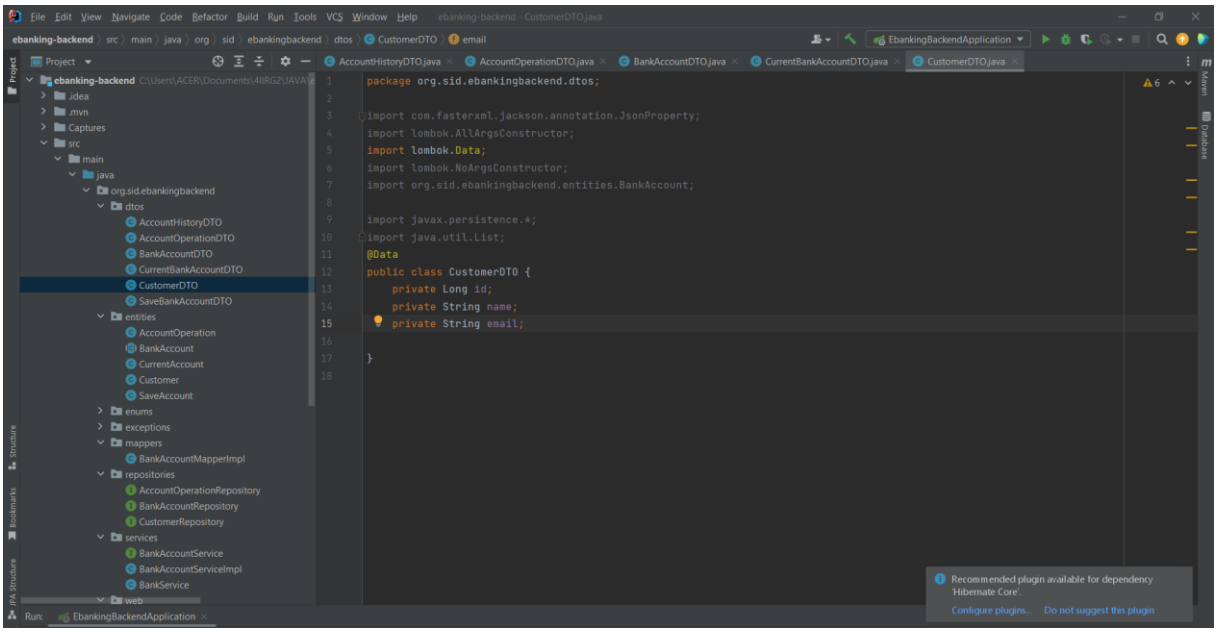
BankAccountDTO.java



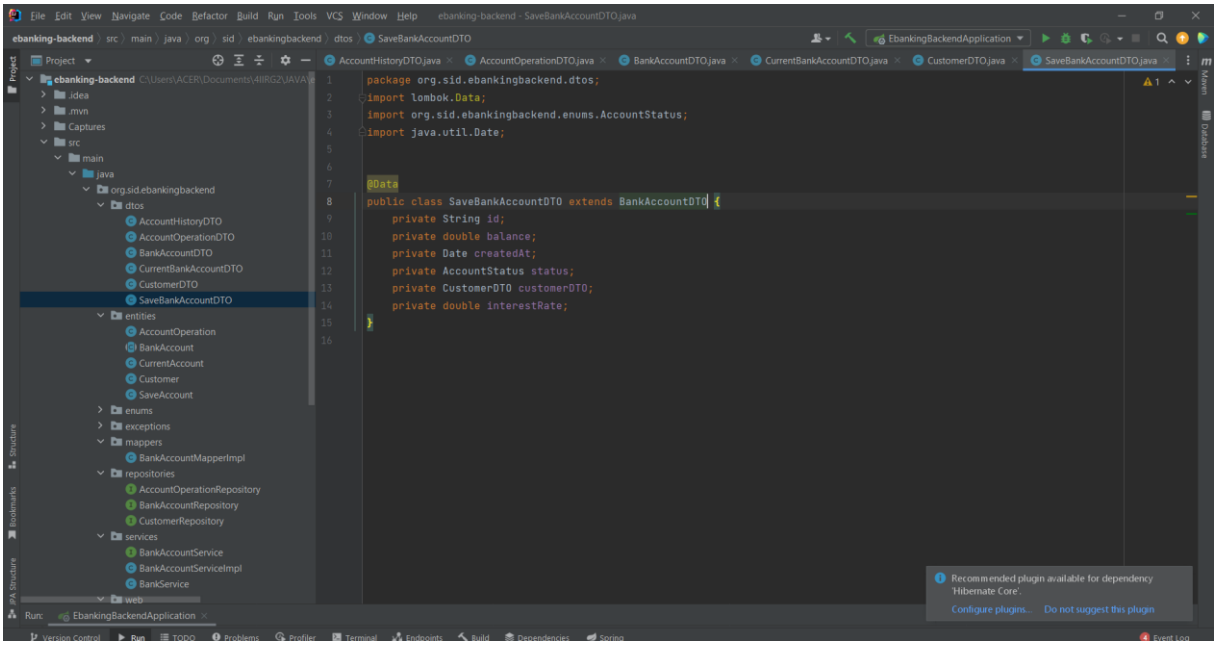
CurrentBankAccountDTO.java



CustomerDTO.java

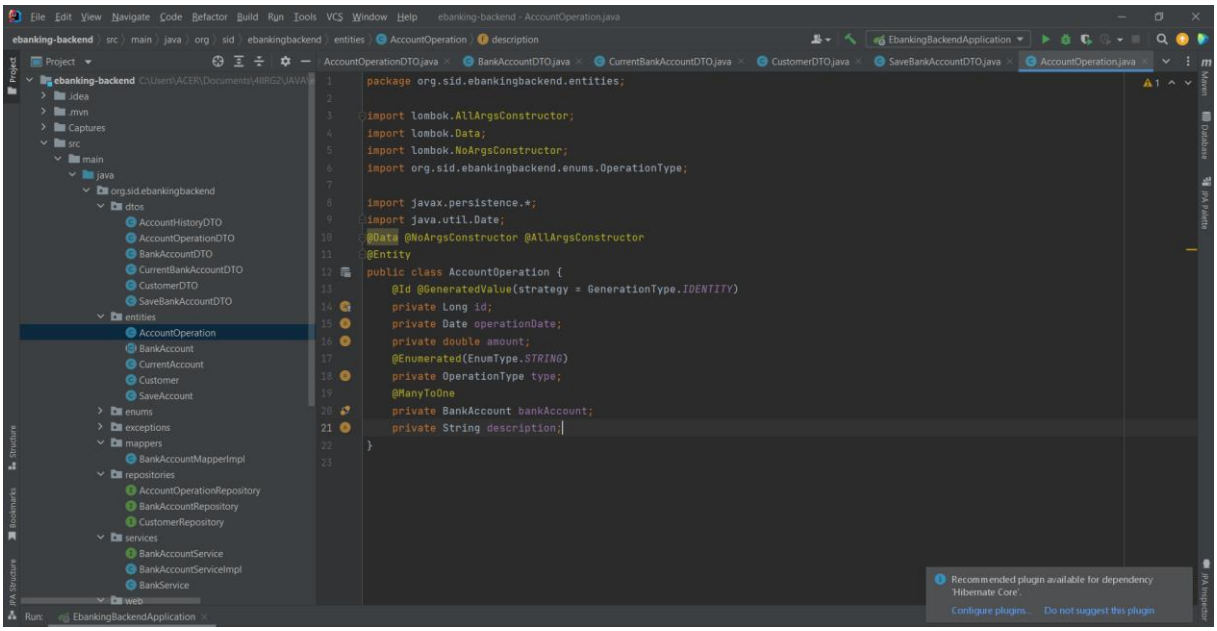


SaveBankAccountDTO.java

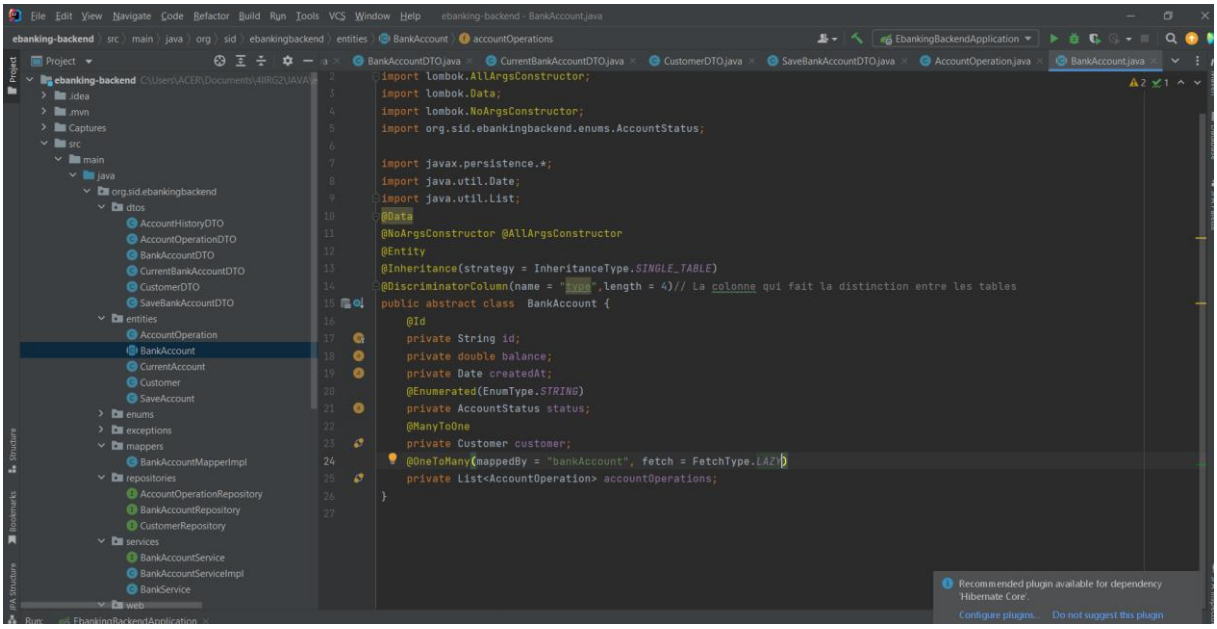


Package entities

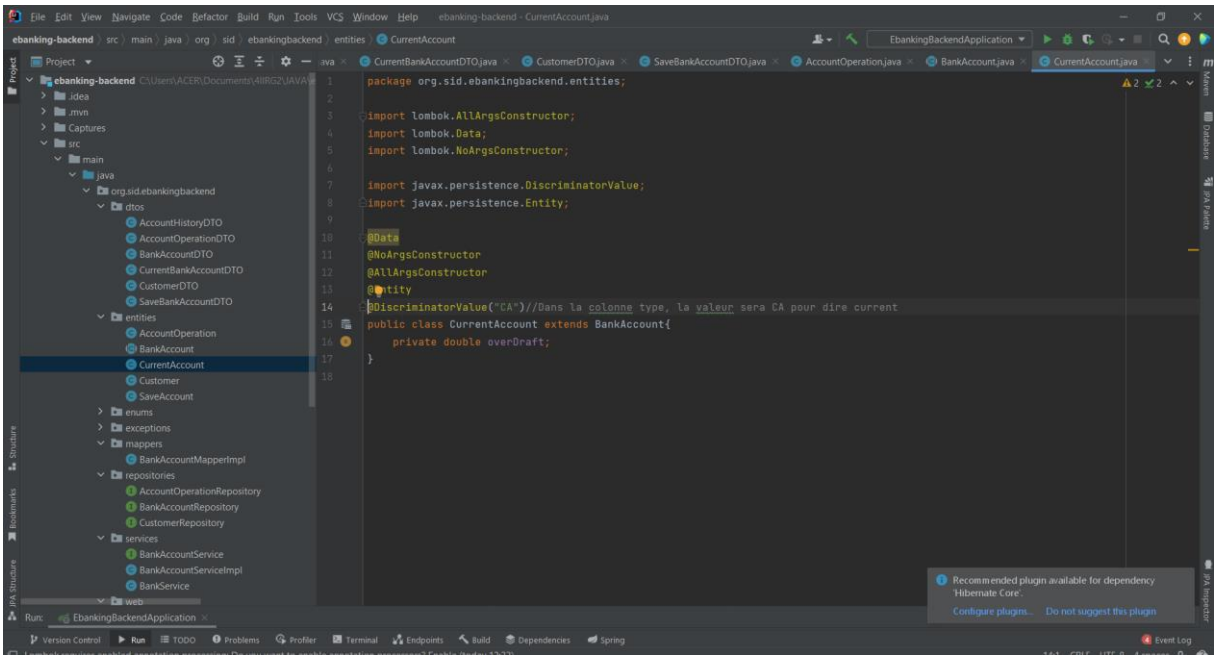
AccountOperation.java



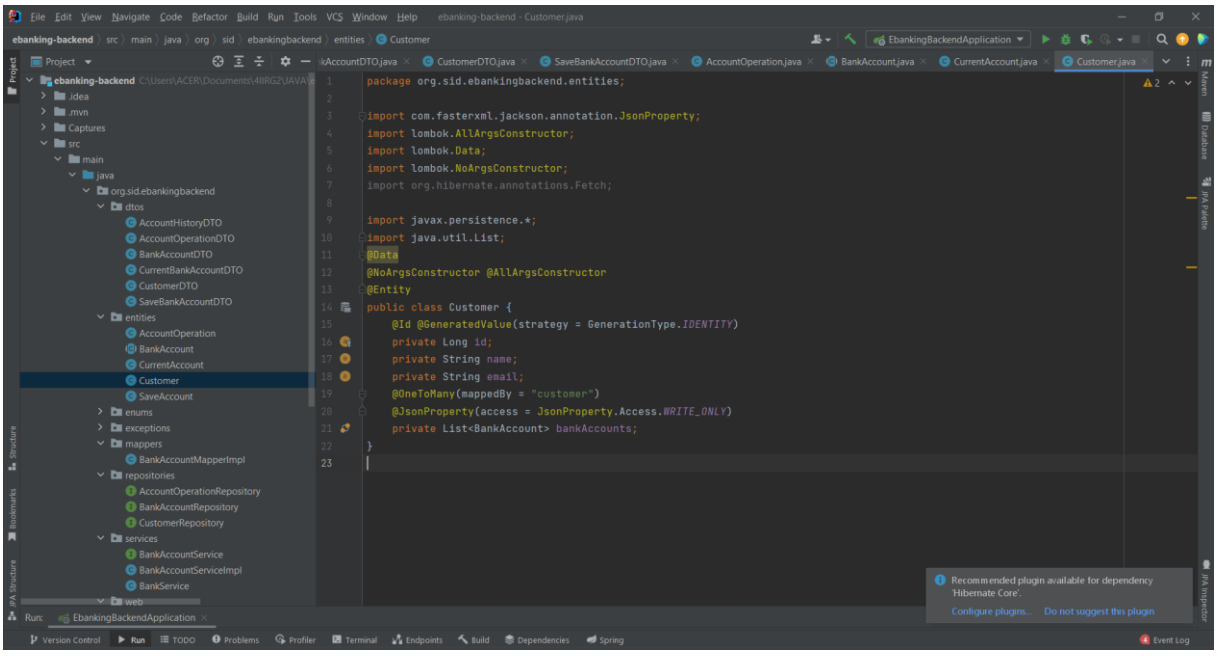
BankAccount.java



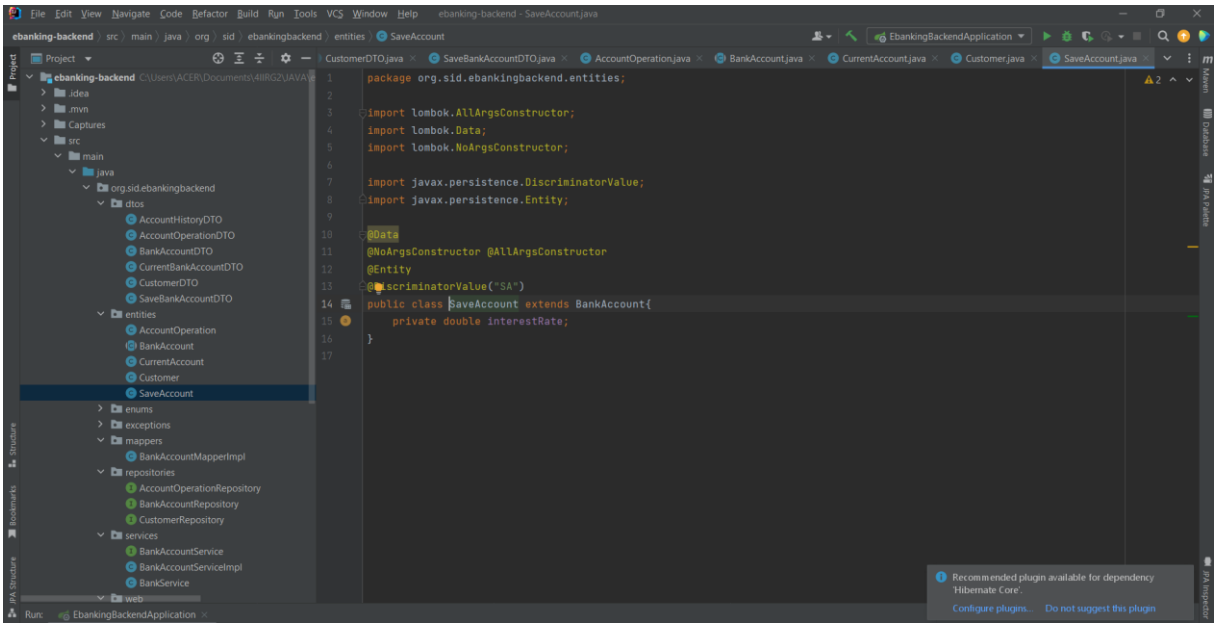
CurrentAccount.java



Customer.java

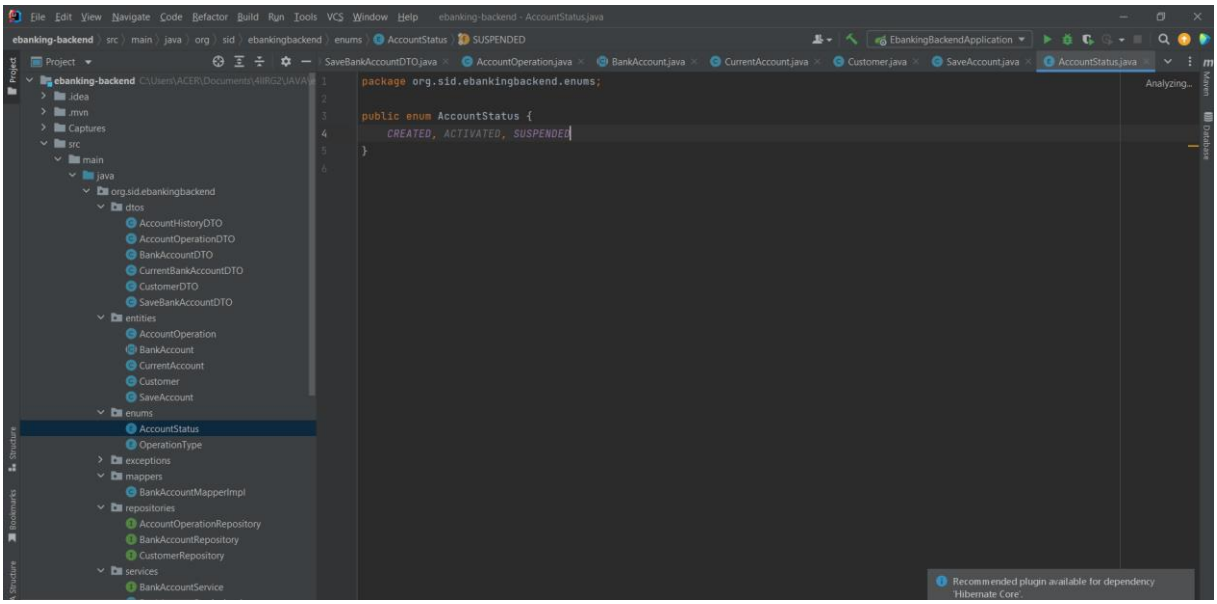


SaveAccount.java

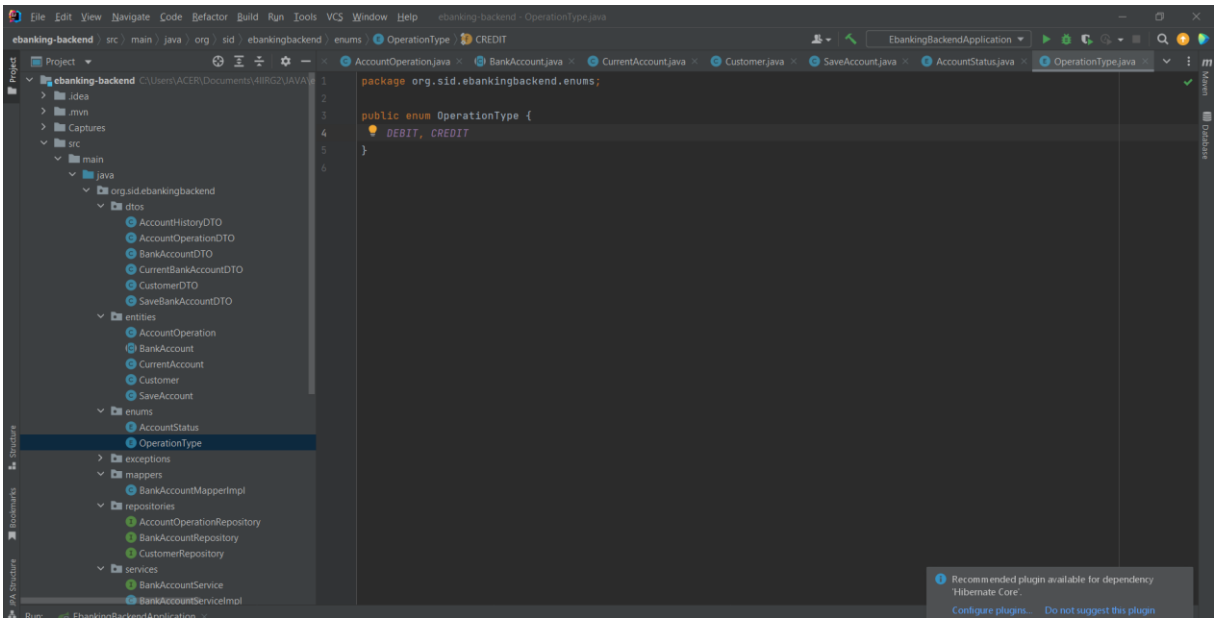


Package enums

AccountStatus.java

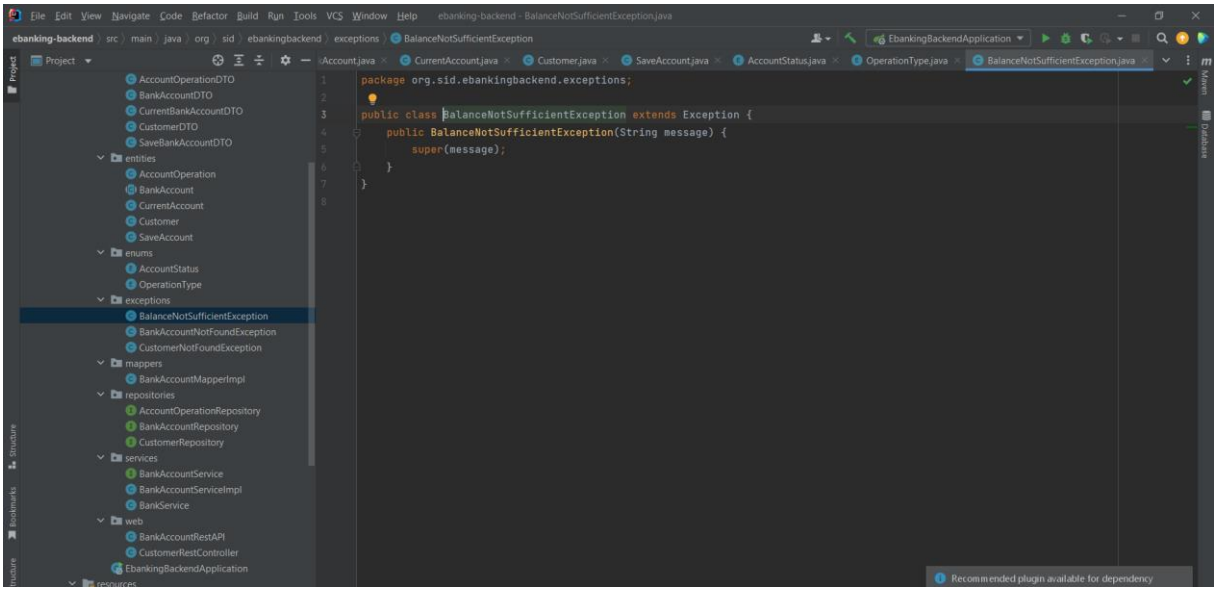


OperationType.java

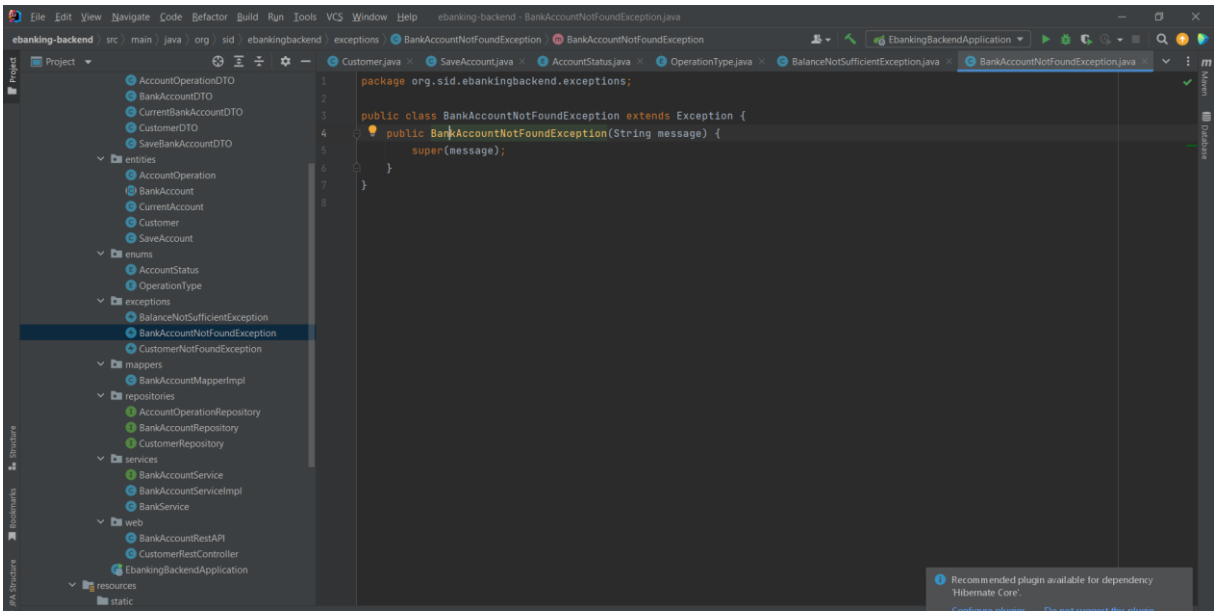


Package exceptions

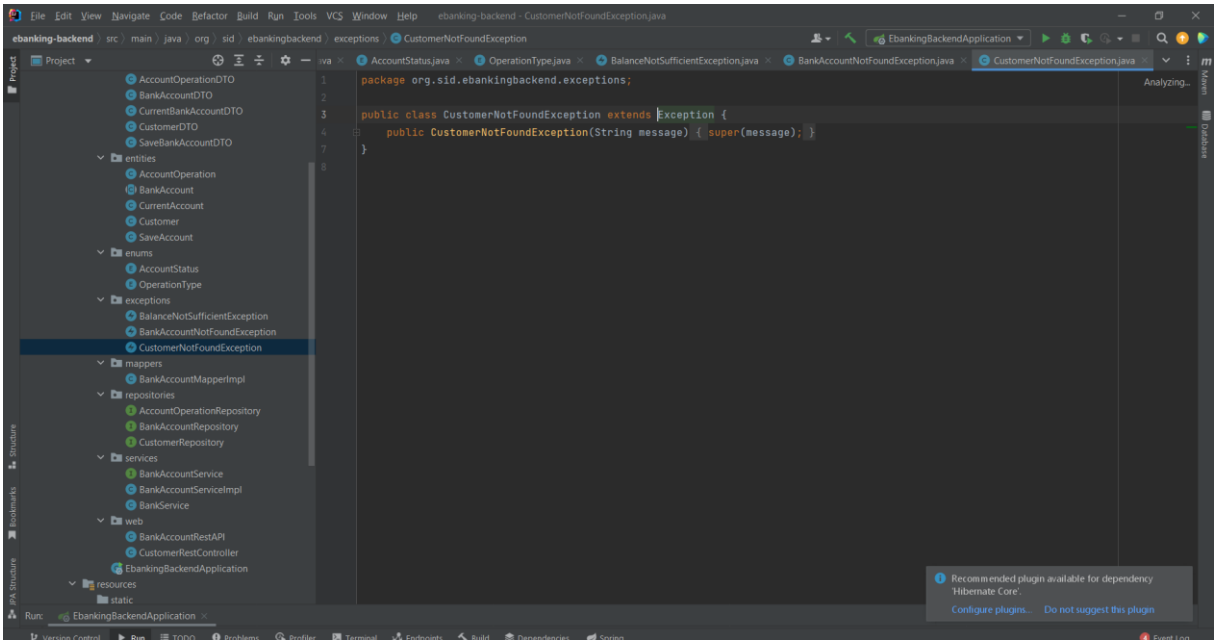
BalanceNotSufficientException.java



BankAccountNotFoundException.java

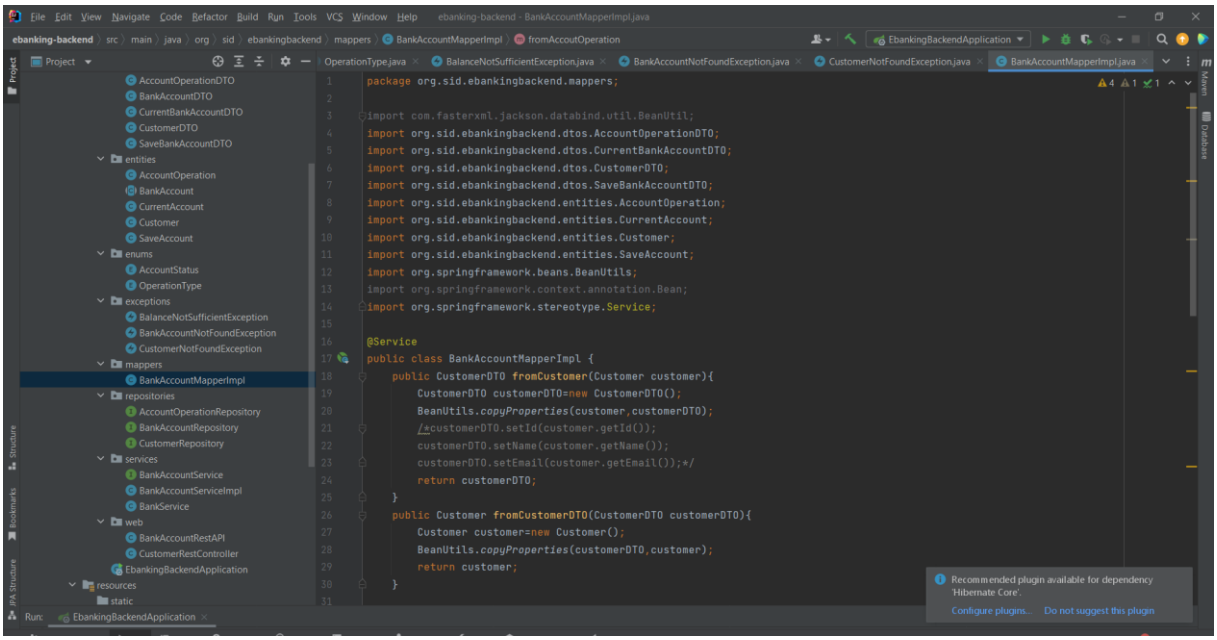


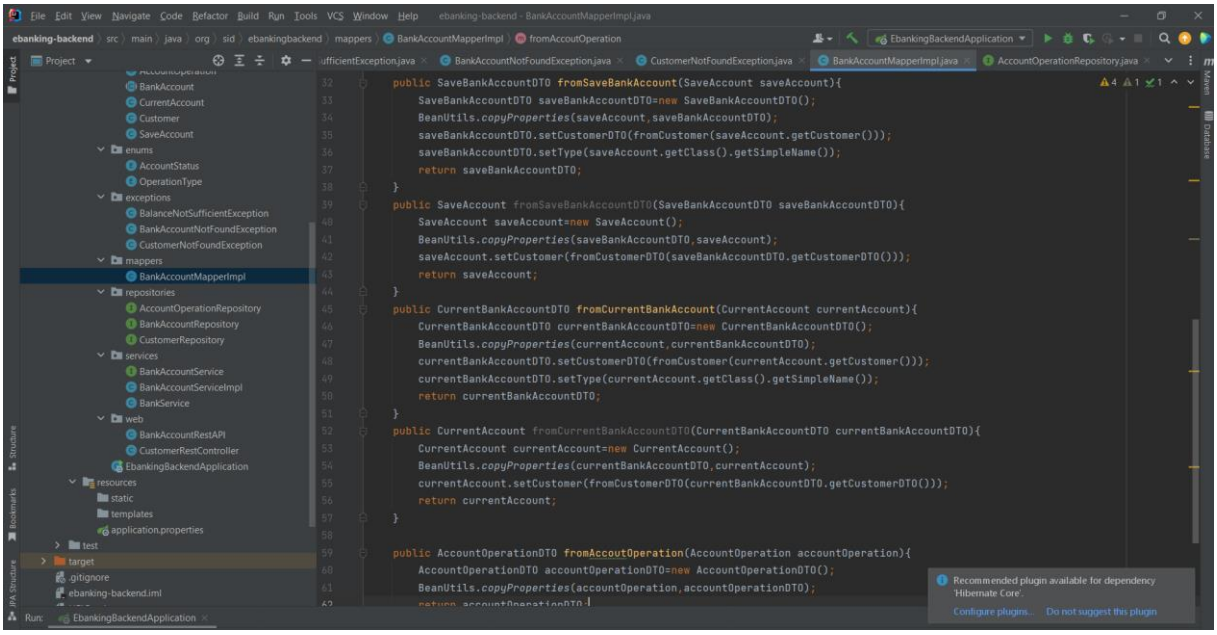
CustomerNotFoundException.java

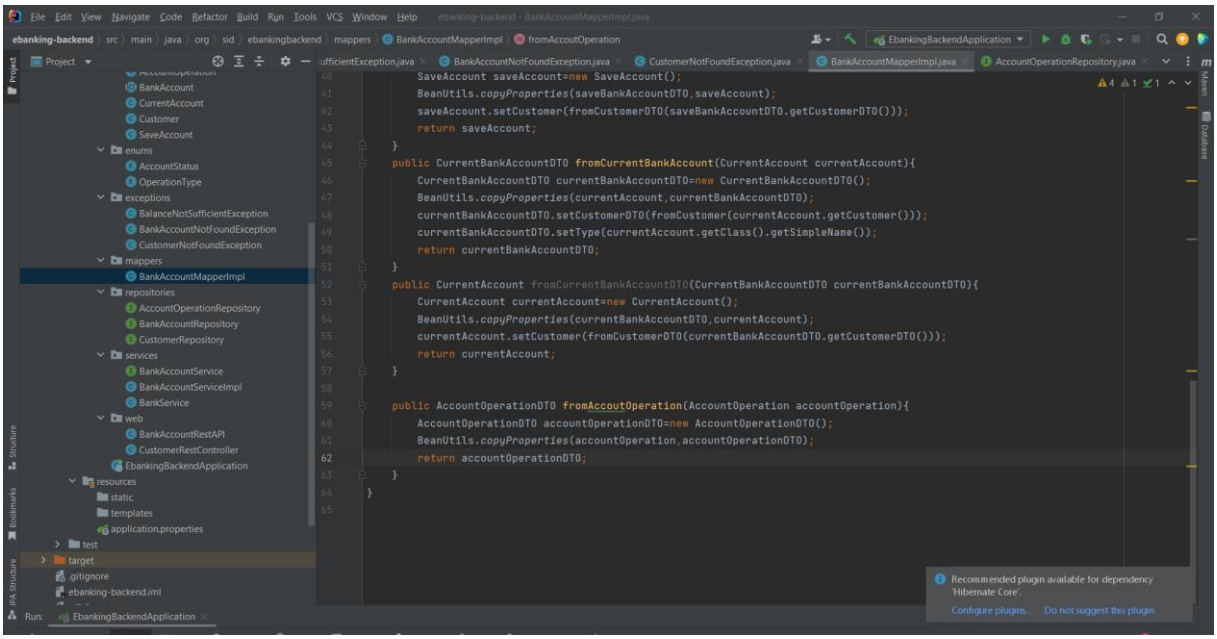


Package mappers

BankAccountMapperImpl.java

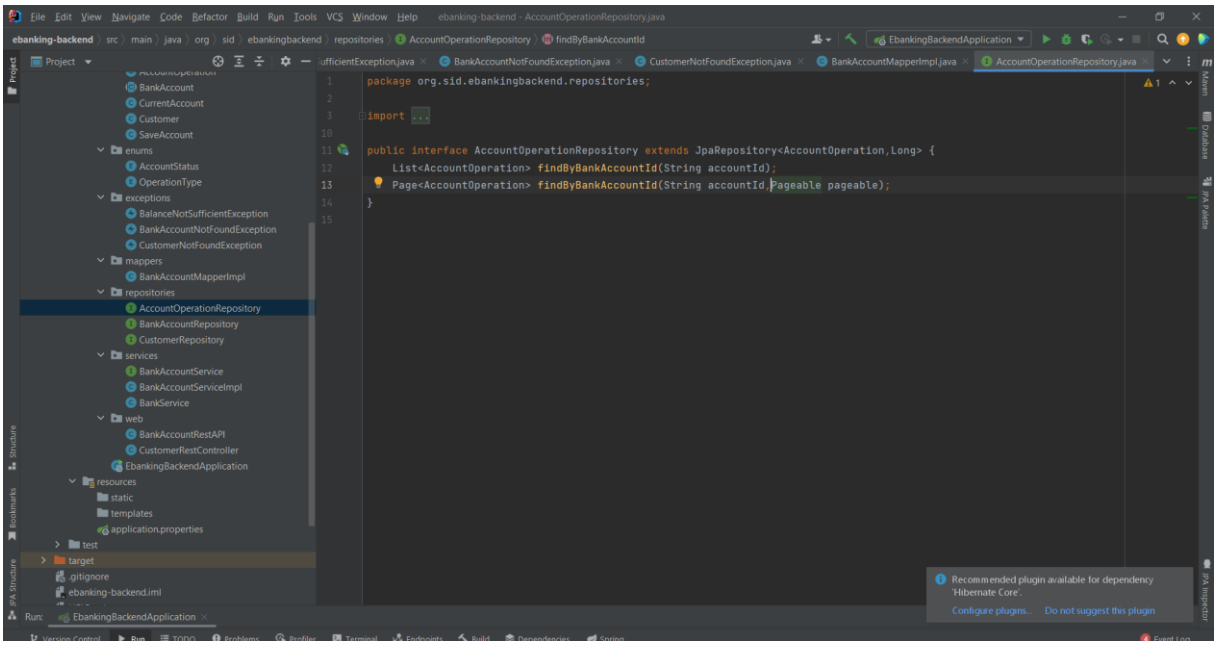

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
ebanking-backend - BankAccountMapperImpl.java
ebanking-backend › src › main › java › org › sid › ebankingbackend › mappers › BankAccountMapperImpl › fromAccoutOperation
EbankingBackendApplication
Project
OperationType.java
BalanceNotSufficientException.java
BankAccountNotFoundException.java
CustomerNotFoundException.java
BankAccountMapperImpl.java
AccountOperationDTO
BankAccountDTO
CurrentBankAccountDTO
CustomerDTO
SaveBankAccountDTO
entities
AccountOperation
BankAccount
CurrentAccount
Customer
SaveAccount
enums
AccountStatus
OperationType
exceptions
BalanceNotSufficientException
BankAccountNotFoundException
CustomerNotFoundException
mappers
BankAccountMapperImpl
repositories
AccountOperationRepository
BankAccountRepository
CustomerRepository
services
BankAccountService
BankAccountServiceImpl
BankService
web
BankAccountRestAPI
CustomerRestController
EbankingBackendApplication
resources
static
1 package org.sid.ebankingbackend.mappers;
2
3 import com.fasterxml.jackson.databind.util.BeanUtil;
4 import org.sid.ebankingbackend.dtos.AccountOperationDTO;
5 import org.sid.ebankingbackend.dtos.CurrentBankAccountDTO;
6 import org.sid.ebankingbackend.dtos.CustomerDTO;
7 import org.sid.ebankingbackend.dtos.SaveBankAccountDTO;
8 import org.sid.ebankingbackend.entities.AccountOperation;
9 import org.sid.ebankingbackend.entities.CurrentAccount;
10 import org.sid.ebankingbackend.entities.Customer;
11 import org.sid.ebankingbackend.entities.SaveAccount;
12 import org.springframework.beans.BeanUtils;
13 import org.springframework.context.annotation.Bean;
14 import org.springframework.stereotype.Service;
15
16 @Service
17 public class BankAccountMapperImpl {
18 public CustomerDTO fromCustomer(Customer customer){
19 CustomerDTO customerDTO=new CustomerDTO();
20 BeanUtils.copyProperties(customer,customerDTO);
21 /*customerDTO.setId(customer.getId());
22 customerDTO.setName(customer.getName());
23 customerDTO.setEmail(customer.getEmail());*/
24 return customerDTO;
25 }
26 public Customer fromCustomerDTO(CustomerDTO customerDTO){
27 Customer customer=new Customer();
28 BeanUtils.copyProperties(customerDTO,customer);
29 return customer;
30 }
31
Recommended plugin available for dependency 'Hibernate Core'.
Configure plugins... Do not suggest this plugin
Run: EbankingBackendApplication


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
ebanking-backend - BankAccountMapperImpl.java
ebanking-backend › src › main › java › org › sid › ebankingbackend › mappers › BankAccountMapperImpl › fromAccoutOperation
EbankingBackendApplication
Project
CustomerNotFoundException.java
BankAccountMapperImpl.java
AccountOperationRepository.java
BankAccount
CurrentAccount
Customer
SaveAccount
enums
AccountStatus
OperationType
exceptions
BalanceNotSufficientException
BankAccountNotFoundException
CustomerNotFoundException
mappers
BankAccountMapperImpl
repositories
AccountOperationRepository
BankAccountRepository
CustomerRepository
services
BankAccountService
BankAccountServiceImpl
BankService
web
BankAccountRestAPI
CustomerRestController
EbankingBackendApplication
resources
static
templates
application.properties
test
target
.gitignore
ebanking-backend.iml
32 public SaveBankAccountDTO fromSaveBankAccount(SaveAccount saveAccount){
33 SaveBankAccountDTO saveBankAccountDTO=new SaveBankAccountDTO();
34 BeanUtils.copyProperties(saveAccount,saveBankAccountDTO);
35 saveBankAccountDTO.setCustomerDTO(fromCustomer(saveAccount.getCustomer()));
36 saveBankAccountDTO.setType(saveAccount.getClass().getSimpleName());
37 return saveBankAccountDTO;
38 }
39 public SaveAccount fromSaveBankAccountDTO(SaveBankAccountDTO saveBankAccountDTO){
40 SaveAccount saveAccount=new SaveAccount();
41 BeanUtils.copyProperties(saveBankAccountDTO,saveAccount);
42 saveAccount.setCustomer(fromCustomerDTO(saveBankAccountDTO.getCustomerDTO()));
43 return saveAccount;
44 }
45 public CurrentBankAccountDTO fromCurrentBankAccount(CurrentAccount currentAccount){
46 CurrentBankAccountDTO currentBankAccountDTO=new CurrentBankAccountDTO();
47 BeanUtils.copyProperties(currentAccount,currentBankAccountDTO);
48 currentBankAccountDTO.setCustomerDTO(fromCustomer(currentAccount.getCustomer()));
49 currentBankAccountDTO.setType(currentAccount.getClass().getSimpleName());
50 return currentBankAccountDTO;
51 }
52 public CurrentAccount fromCurrentBankAccountDTO(CurrentBankAccountDTO currentBankAccountDTO){
53 CurrentAccount currentAccount=new CurrentAccount();
54 BeanUtils.copyProperties(currentBankAccountDTO,currentAccount);
55 currentAccount.setCustomer(fromCustomerDTO(currentBankAccountDTO.getCustomerDTO()));
56 return currentAccount;
57 }
58
59 public AccountOperationDTO fromAccoutOperation(AccountOperation accountOperation){
60 AccountOperationDTO accountOperationDTO=new AccountOperationDTO();
61 BeanUtils.copyProperties(accountOperation,accountOperationDTO);
Recommended plugin available for dependency 'Hibernate Core'.
Configure plugins... Do not suggest this plugin
Run: EbankingBackendApplication

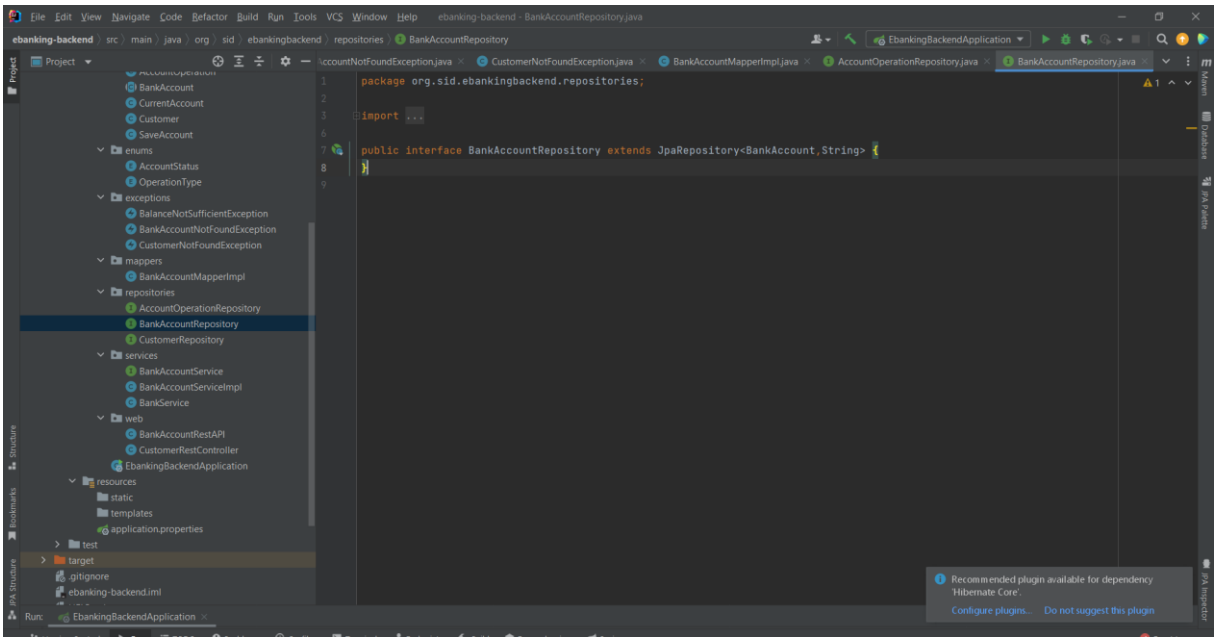


Package repositories

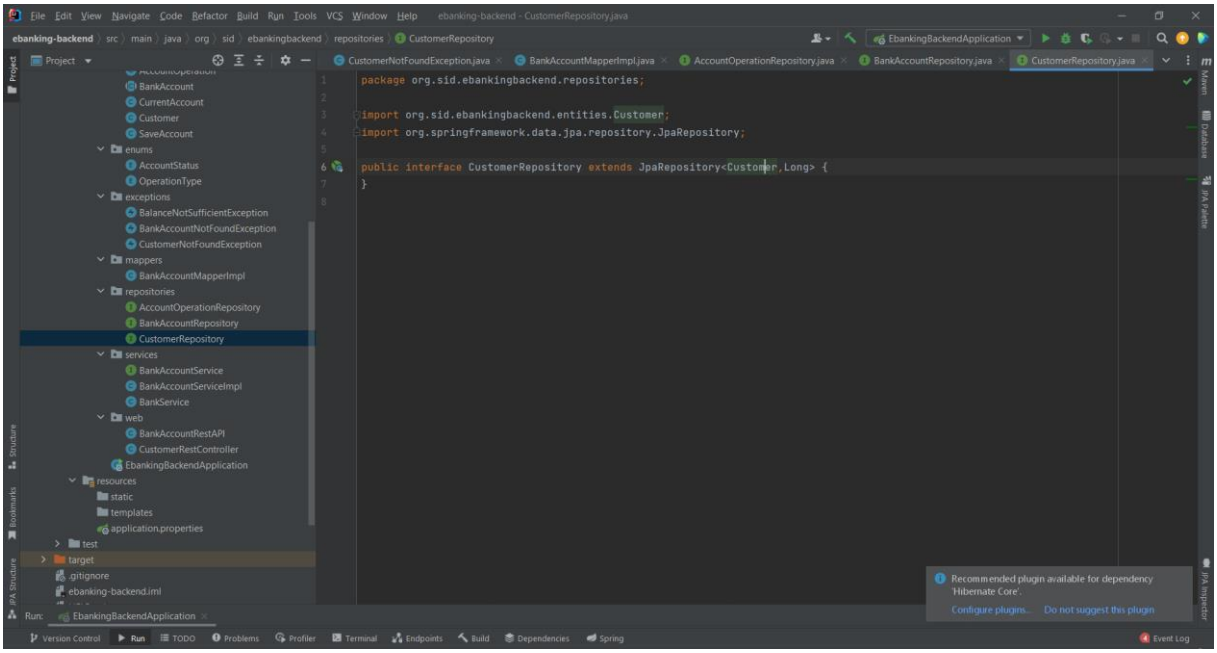
AccountOperationRepository.java



BankAccountRepository.java

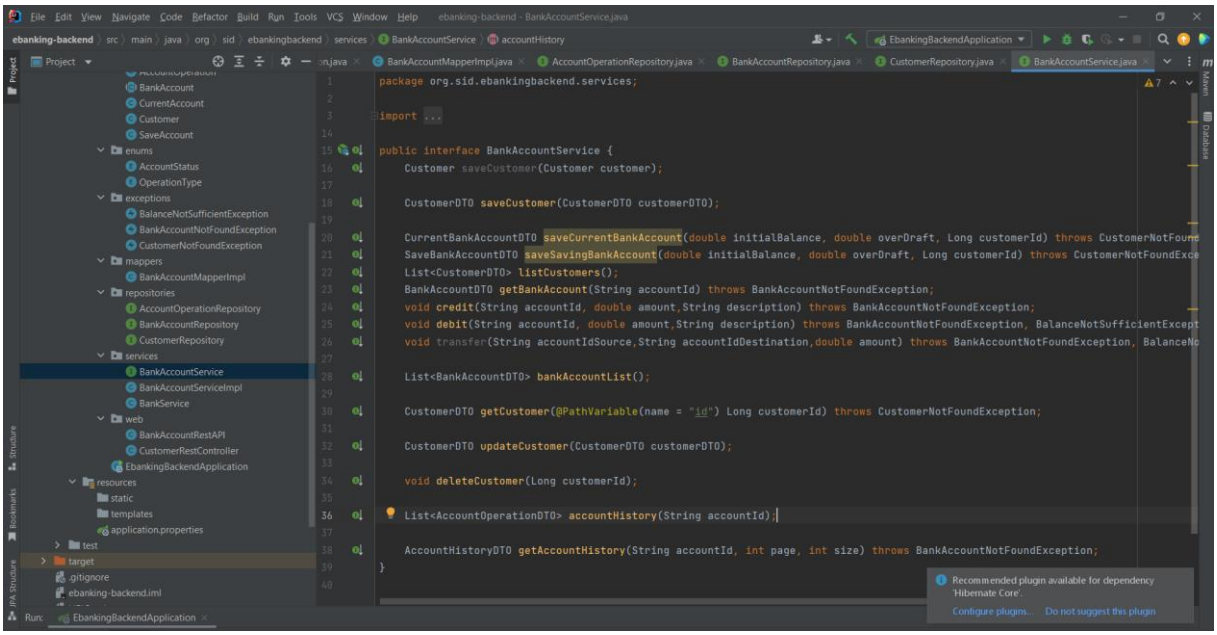


CustomerRepository.java



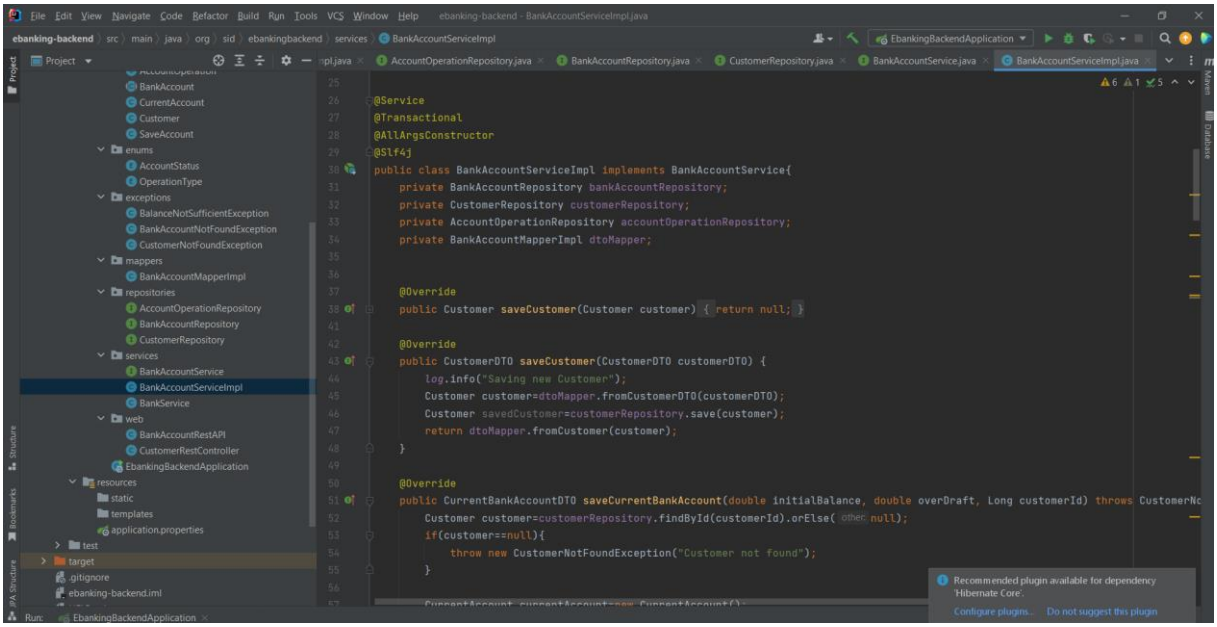
Package services

BankAccountService.java

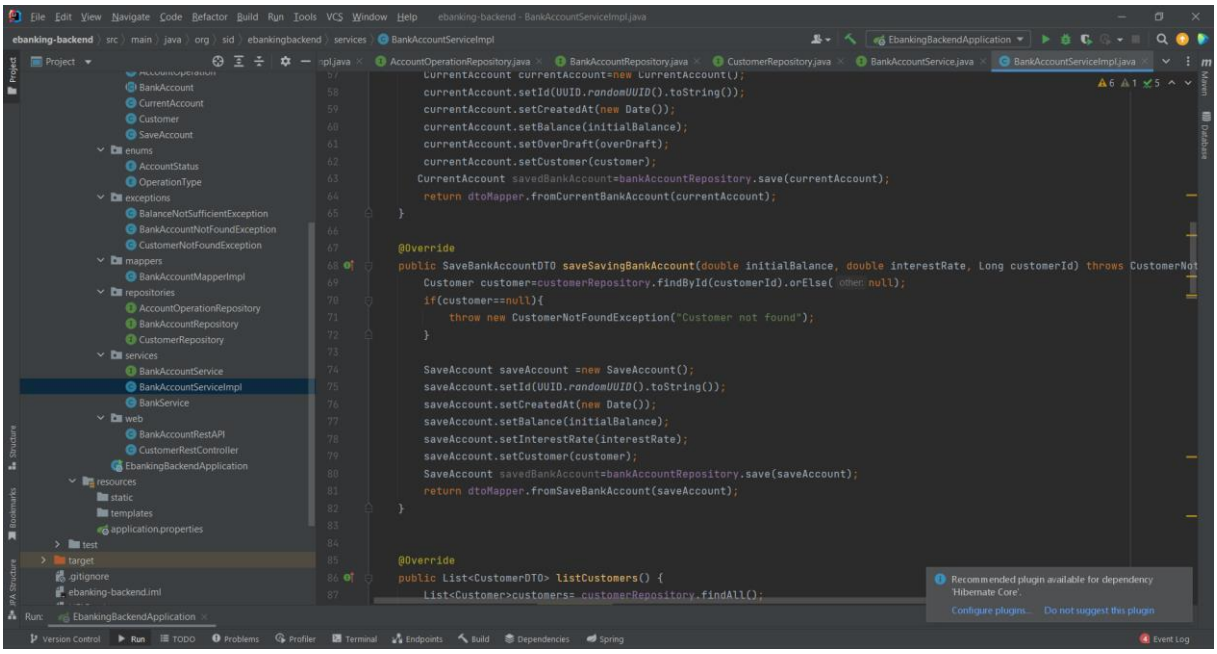


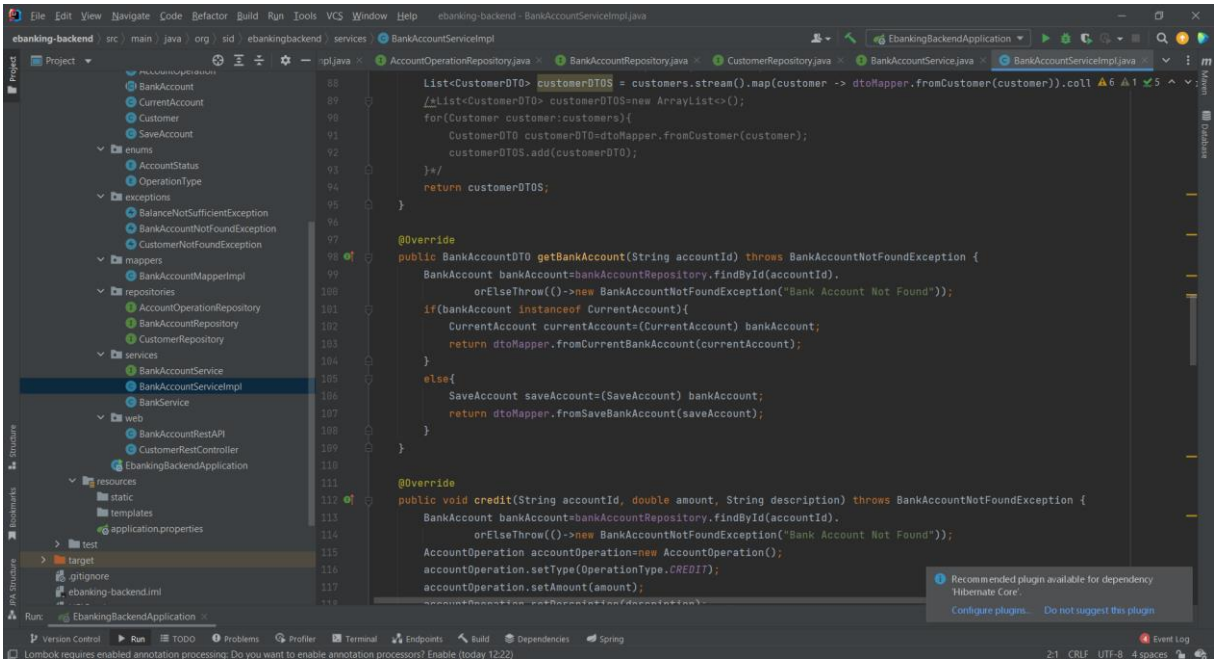
```java
package org.sid.ebankingbackend.services;

import ...

public interface BankAccountService {
    Customer saveCustomer(Customer customer);

    CustomerDTO saveCustomer(CustomerDTO customerDTO);

    CurrentBankAccountDTO saveCurrentBankAccount(double initialBalance, double overDraft, Long customerId) throws CustomerNotFound
    SaveBankAccountDTO saveSavingBankAccount(double initialBalance, double overDraft, Long customerId) throws CustomerNotFoundExce
    List<CustomerDTO> listCustomers();
    BankAccountDTO getBankAccount(String accountId) throws BankAccountNotFoundException;
    void credit(String accountId, double amount,String description) throws BankAccountNotFoundException;
    void debit(String accountId, double amount,String description) throws BankAccountNotFoundException, BalanceNotSufficientExcept
    void transfer(String accountIdSource,String accountIdDestination,double amount) throws BankAccountNotFoundException, BalanceNo

    List<BankAccountDTO> bankAccountList();

    CustomerDTO getCustomer(@PathVariable(name = "id") Long customerId) throws CustomerNotFoundException;

    CustomerDTO updateCustomer(CustomerDTO customerDTO);

    void deleteCustomer(Long customerId);

    List<AccountOperationDTO> accountHistory(String accountId);

    AccountHistoryDTO getAccountHistory(String accountId, int page, int size) throws BankAccountNotFoundException;
}
```

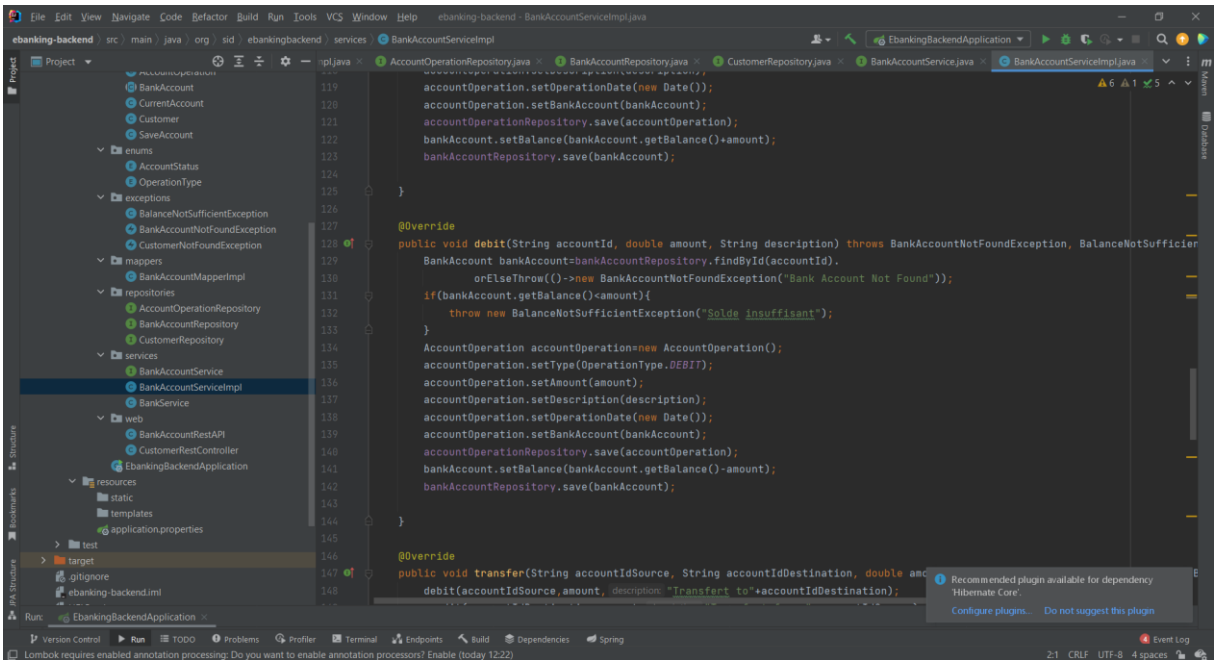
BankAccountServiceImpl.java



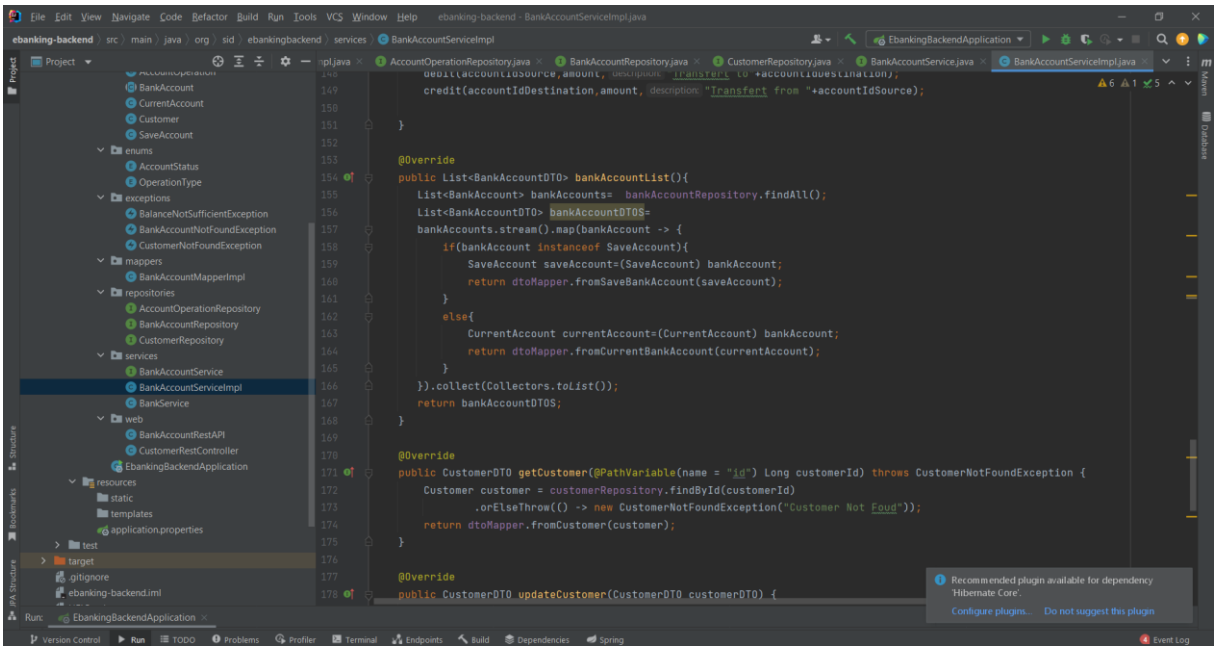
```java
@Service
@Transactional
@AllArgsConstructor
@Slf4j
public class BankAccountServiceImpl implements BankAccountService{
    private BankAccountRepository bankAccountRepository;
    private CustomerRepository customerRepository;
    private AccountOperationRepository accountOperationRepository;
    private BankAccountMapperImpl dtoMapper;


    @Override
    public Customer saveCustomer(Customer customer) { return null; }

    @Override
    public CustomerDTO saveCustomer(CustomerDTO customerDTO) {
        log.info("Saving new Customer");
        Customer customer=dtoMapper.fromCustomerDTO(customerDTO);
        Customer savedCustomer=customerRepository.save(customer);
        return dtoMapper.fromCustomer(customer);
    }

    @Override
    public CurrentBankAccountDTO saveCurrentBankAccount(double initialBalance, double overDraft, Long customerId) throws CustomerNo
        Customer customer=customerRepository.findById(customerId).orElse( other: null);
        if(customer==null){
            throw new CustomerNotFoundException("Customer not found");
        }
```


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
ebanking-backend - BankAccountServiceImpl.java
ebanking-backend › src › main › java › org › sid › ebankingbackend › services › BankAccountServiceImpl
EbankingBackendApplication
Project
AccountOperationRepository.java
BankAccountRepository.java
CustomerRepository.java
BankAccountService.java
BankAccountServiceImpl.java
BankAccount
CurrentAccount
Customer
SaveAccount
enums
AccountStatus
OperationType
exceptions
BalanceNotSufficientException
BankAccountNotFoundException
CustomerNotFoundException
mappers
BankAccountMapperImpl
repositories
AccountOperationRepository
BankAccountRepository
CustomerRepository
services
BankAccountService
BankAccountServiceImpl
BankService
web
BankAccountRestAPI
CustomerRestController
EbankingBackendApplication
resources
static
templates
application.properties
test
target
.gitignore
ebanking-backend.iml
CurrentAccount currentAccount=new CurrentAccount();
currentAccount.setId(UUID.randomUUID().toString());
currentAccount.setCreatedAt(new Date());
currentAccount.setBalance(initialBalance);
currentAccount.setOverDraft(overDraft);
currentAccount.setCustomer(customer);
CurrentAccount savedBankAccount=bankAccountRepository.save(currentAccount);
return dtoMapper.fromCurrentBankAccount(currentAccount);
}
@Override
public SaveBankAccountDTO saveSavingBankAccount(double initialBalance, double interestRate, Long customerId) throws CustomerNot
Customer customer=customerRepository.findById(customerId).orElse(other: null);
if(customer==null){
throw new CustomerNotFoundException("Customer not found");
}
SaveAccount saveAccount =new SaveAccount();
saveAccount.setId(UUID.randomUUID().toString());
saveAccount.setCreatedAt(new Date());
saveAccount.setBalance(initialBalance);
saveAccount.setInterestRate(interestRate);
saveAccount.setCustomer(customer);
SaveAccount savedBankAccount=bankAccountRepository.save(saveAccount);
return dtoMapper.fromSaveBankAccount(saveAccount);
}
@Override
public List<CustomerDTO> listCustomers() {
List<Customer>customers= customerRepository.findAll();
Recommended plugin available for dependency 'Hibernate Core'.
Configure plugins... Do not suggest this plugin
Run: EbankingBackendApplication
Version Control Run TODO Problems Profiler Terminal Endpoints Build Dependencies Spring
Event Log

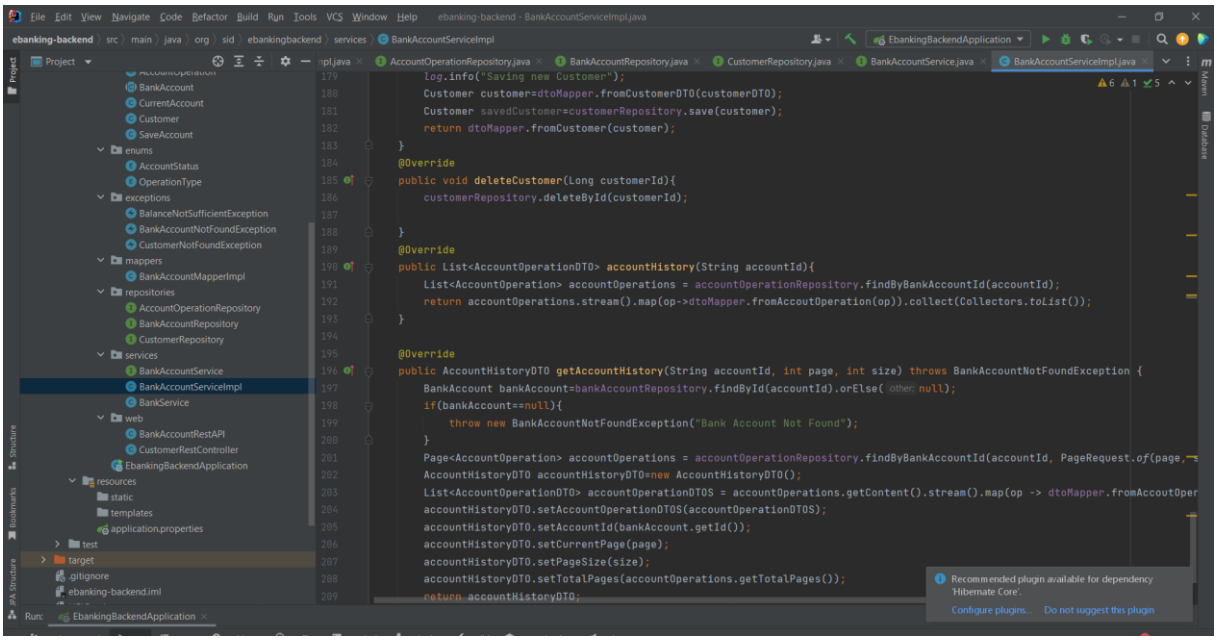

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
ebanking-backend - BankAccountServiceImpl.java
ebanking-backend › src › main › java › org › sid › ebankingbackend › services › BankAccountServiceImpl
EbankingBackendApplication
Project
AccountOperationRepository.java
BankAccountRepository.java
CustomerRepository.java
BankAccountService.java
BankAccountServiceImpl.java
BankAccount
CurrentAccount
Customer
SaveAccount
enums
AccountStatus
OperationType
exceptions
BalanceNotSufficientException
BankAccountNotFoundException
CustomerNotFoundException
mappers
BankAccountMapperImpl
repositories
AccountOperationRepository
BankAccountRepository
CustomerRepository
services
BankAccountService
BankAccountServiceImpl
BankService
web
BankAccountRestAPI
CustomerRestController
EbankingBackendApplication
resources
static
templates
application.properties
test
target
.gitignore
ebanking-backend.iml
List<CustomerDTO> customerDTOS = customers.stream().map(customer -> dtoMapper.fromCustomer(customer)).coll
/*List<CustomerDTO> customerDTOS=new ArrayList<>();
for(Customer customer:customers){
CustomerDTO customerDTO=dtoMapper.fromCustomer(customer);
customerDTOS.add(customerDTO);
}*/
return customerDTOS;
}
@Override
public BankAccountDTO getBankAccount(String accountId) throws BankAccountNotFoundException {
BankAccount bankAccount=bankAccountRepository.findById(accountId).
orElseThrow(()->new BankAccountNotFoundException("Bank Account Not Found"));
if(bankAccount instanceof CurrentAccount){
CurrentAccount currentAccount=(CurrentAccount) bankAccount;
return dtoMapper.fromCurrentBankAccount(currentAccount);
}
else{
SaveAccount saveAccount=(SaveAccount) bankAccount;
return dtoMapper.fromSaveBankAccount(saveAccount);
}
}
@Override
public void credit(String accountId, double amount, String description) throws BankAccountNotFoundException {
BankAccount bankAccount=bankAccountRepository.findById(accountId).
orElseThrow(()->new BankAccountNotFoundException("Bank Account Not Found"));
AccountOperation accountOperation=new AccountOperation();
accountOperation.setType(OperationType.CREDIT);
accountOperation.setAmount(amount);
Recommended plugin available for dependency 'Hibernate Core'.
Configure plugins... Do not suggest this plugin
Run: EbankingBackendApplication
Version Control Run TODO Problems Profiler Terminal Endpoints Build Dependencies Spring
Event Log
Lombok requires enabled annotation processing: Do you want to enable annotation processors? Enable (today 12:22)
2:1 CRLF UTF-8 4 spaces

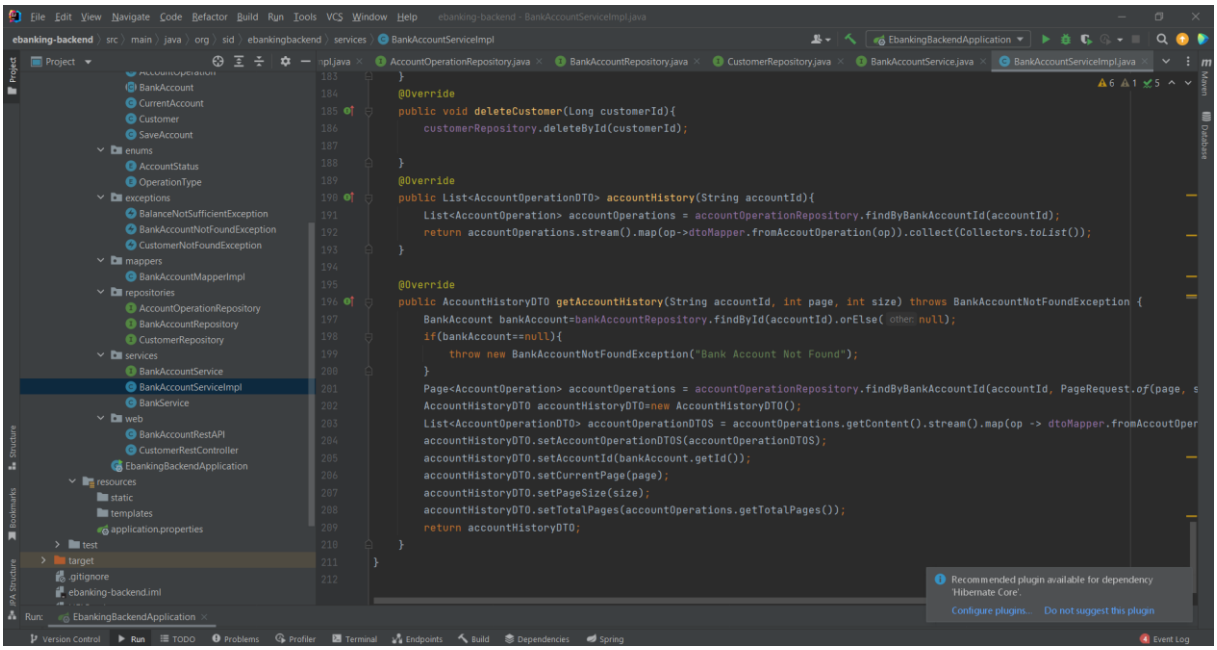


```
        accountOperation.setOperationDate(new Date());
        accountOperation.setBankAccount(bankAccount);
        accountOperationRepository.save(accountOperation);
        bankAccount.setBalance(bankAccount.getBalance()+amount);
        bankAccountRepository.save(bankAccount);


    }

    @Override
    public void debit(String accountId, double amount, String description) throws BankAccountNotFoundException, BalanceNotSufficier
        BankAccount bankAccount=bankAccountRepository.findById(accountId).
                orElseThrow(()->new BankAccountNotFoundException("Bank Account Not Found"));
        if(bankAccount.getBalance()<amount){
            throw new BalanceNotSufficientException("Solde insuffisant");
        }
        AccountOperation accountOperation=new AccountOperation();
        accountOperation.setType(OperationType.DEBIT);
        accountOperation.setAmount(amount);
        accountOperation.setDescription(description);
        accountOperation.setOperationDate(new Date());
        accountOperation.setBankAccount(bankAccount);
        accountOperationRepository.save(accountOperation);
        bankAccount.setBalance(bankAccount.getBalance()-amount);
        bankAccountRepository.save(bankAccount);


    }

    @Override
    public void transfer(String accountIdSource, String accountIdDestination, double amo
        debit(accountIdSource,amount, description: "Transfert to"+accountIdDestination);
```

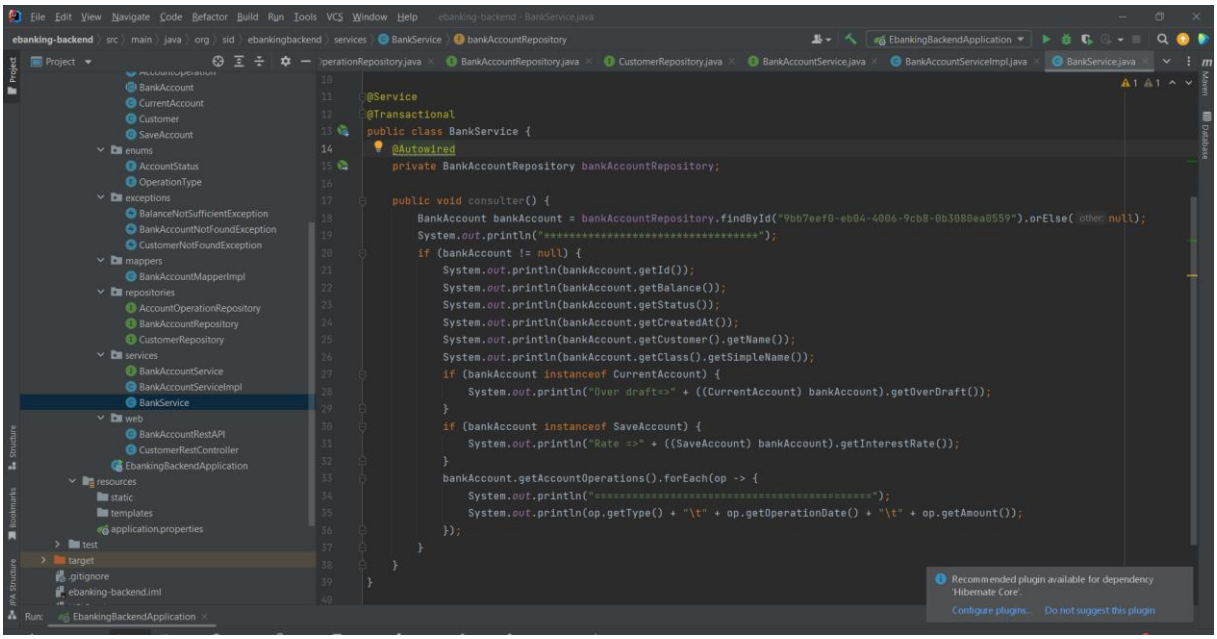


```
        debit(accountIdSource,amount, description: "Transfert to "+accountIdDestination);
        credit(accountIdDestination,amount, description: "Transfert from "+accountIdSource);


    }

    @Override
    public List<BankAccountDTO> bankAccountList(){
       List<BankAccount> bankAccounts=  bankAccountRepository.findAll();
       List<BankAccountDTO> bankAccountDTOS=
       bankAccounts.stream().map(bankAccount -> {
           if(bankAccount instanceof SaveAccount){
               SaveAccount saveAccount=(SaveAccount) bankAccount;
               return dtoMapper.fromSaveBankAccount(saveAccount);
           }
           else{
               CurrentAccount currentAccount=(CurrentAccount) bankAccount;
               return dtoMapper.fromCurrentBankAccount(currentAccount);
           }
       }).collect(Collectors.toList());
       return bankAccountDTOS;
    }

    @Override
    public CustomerDTO getCustomer(@PathVariable(name = "id") Long customerId) throws CustomerNotFoundException {
        Customer customer = customerRepository.findById(customerId)
                .orElseThrow(() -> new CustomerNotFoundException("Customer Not Foud"));
        return dtoMapper.fromCustomer(customer);
    }

    @Override
    public CustomerDTO updateCustomer(CustomerDTO customerDTO) {
```





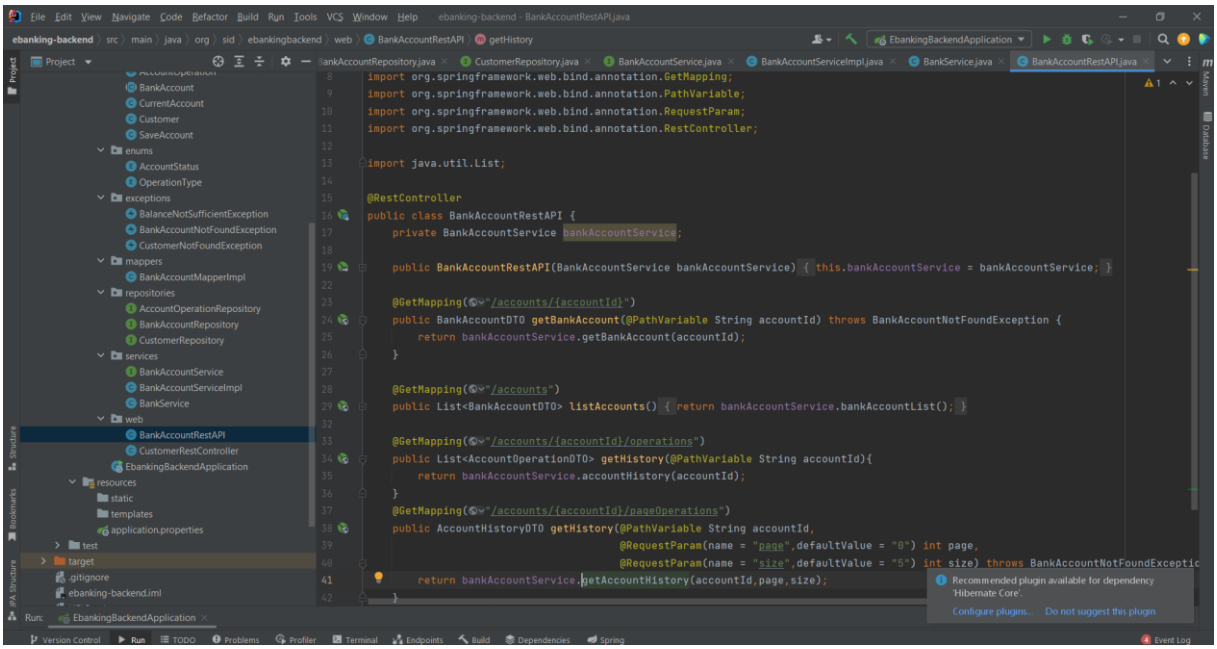
BankService.java



```java
@Service
@Transactional
public class BankService {
    @Autowired
    private BankAccountRepository bankAccountRepository;

    public void consulter() {
        BankAccount bankAccount = bankAccountRepository.findById("9bb7eef0-eb04-4006-9cb8-0b3088ea0559").orElse( other: null);
        System.out.println("**********************************");
        if (bankAccount != null) {
            System.out.println(bankAccount.getId());
            System.out.println(bankAccount.getBalance());
            System.out.println(bankAccount.getStatus());
            System.out.println(bankAccount.getCreatedAt());
            System.out.println(bankAccount.getCustomer().getName());
            System.out.println(bankAccount.getClass().getSimpleName());
            if (bankAccount instanceof CurrentAccount) {
                System.out.println("Over draft=>" + ((CurrentAccount) bankAccount).getOverDraft());
            }
            if (bankAccount instanceof SaveAccount) {
                System.out.println("Rate =>" + ((SaveAccount) bankAccount).getInterestRate());
            }
            bankAccount.getAccountOperations().forEach(op -> {
                System.out.println("==========================================");
                System.out.println(op.getType() + "\t" + op.getOperationDate() + "\t" + op.getAmount());
            });
        }
    }
}
```

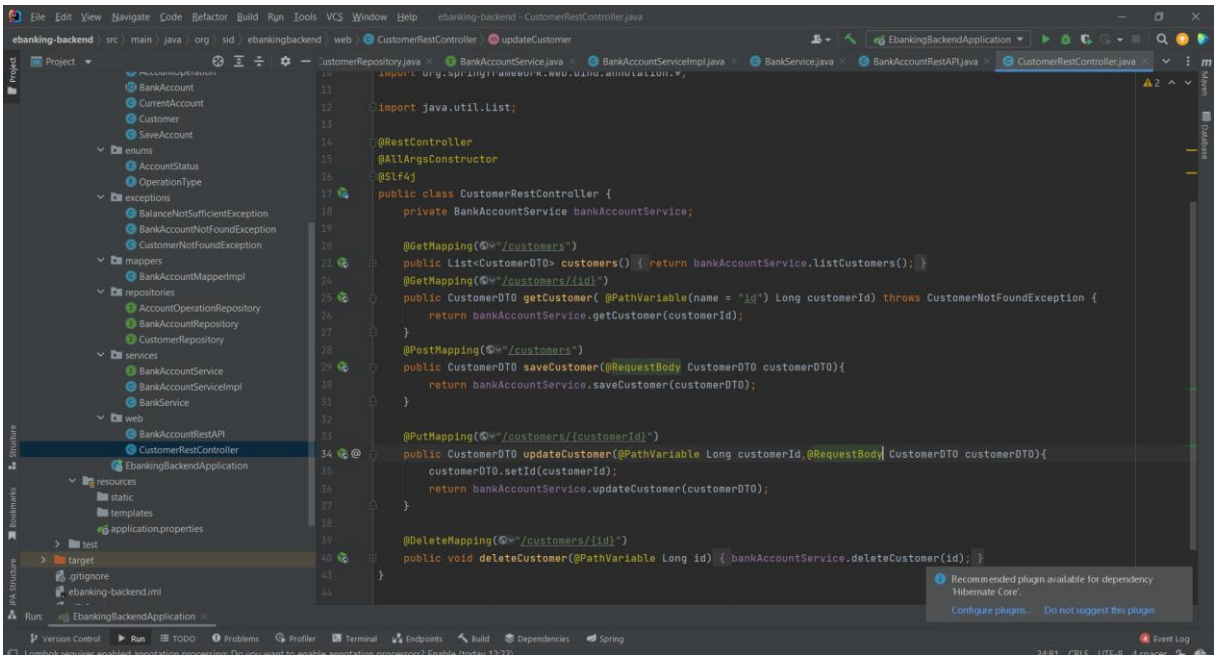
Package web

BankAccountRestAPI.java:



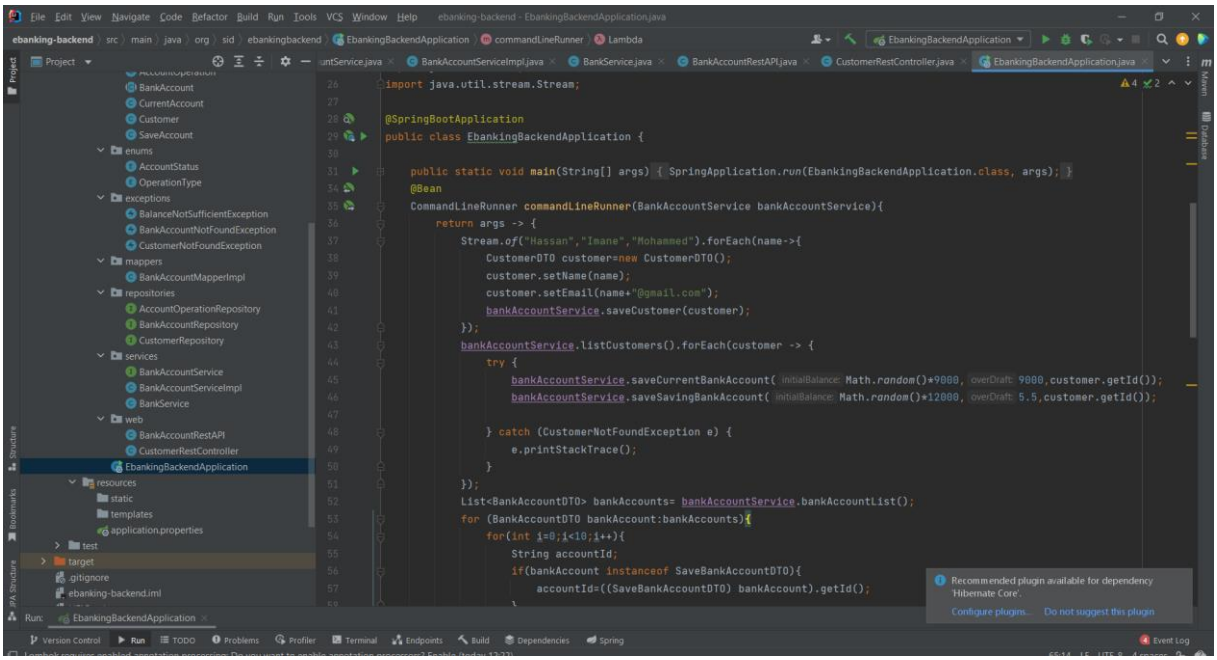
```java
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
public class BankAccountRestAPI {
    private BankAccountService bankAccountService;

    public BankAccountRestAPI(BankAccountService bankAccountService) { this.bankAccountService = bankAccountService; }

    @GetMapping("/accounts/{accountId}")
    public BankAccountDTO getBankAccount(@PathVariable String accountId) throws BankAccountNotFoundException {
        return bankAccountService.getBankAccount(accountId);
    }

    @GetMapping("/accounts")
    public List<BankAccountDTO> listAccounts() { return bankAccountService.bankAccountList(); }

    @GetMapping("/accounts/{accountId}/operations")
    public List<AccountOperationDTO> getHistory(@PathVariable String accountId){
        return bankAccountService.accountHistory(accountId);
    }
    @GetMapping("/accounts/{accountId}/pageOperations")
    public AccountHistoryDTO getHistory(@PathVariable String accountId,
                                        @RequestParam(name = "page",defaultValue = "0") int page,
                                        @RequestParam(name = "size",defaultValue = "5") int size) throws BankAccountNotFoundExceptic
        return bankAccountService.getAccountHistory(accountId,page,size);
    }
```

CustomerRestController.java

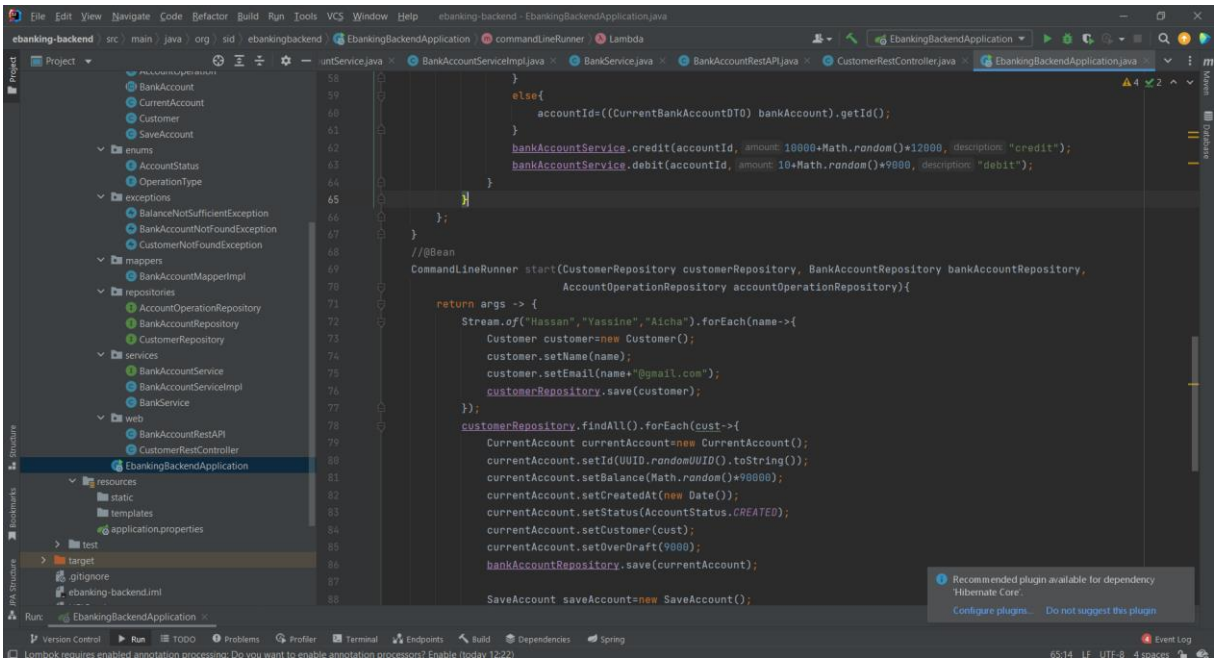


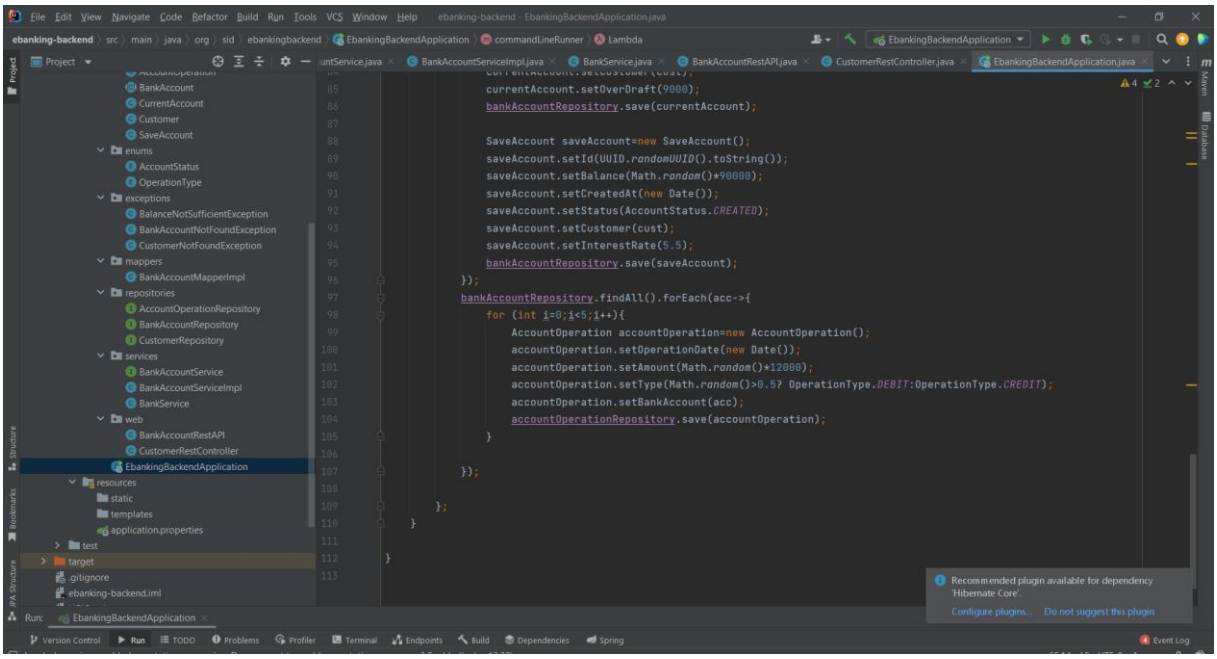
```
import java.util.List;

@RestController
@AllArgsConstructor
@Slf4j
public class CustomerRestController {
    private BankAccountService bankAccountService;

    @GetMapping("/customers")
    public List<CustomerDTO> customers() { return bankAccountService.listCustomers(); }
    @GetMapping("/customers/{id}")
    public CustomerDTO getCustomer( @PathVariable(name = "id") Long customerId) throws CustomerNotFoundException {
        return bankAccountService.getCustomer(customerId);
    }
    @PostMapping("/customers")
    public CustomerDTO saveCustomer(@RequestBody CustomerDTO customerDTO){
        return bankAccountService.saveCustomer(customerDTO);
    }

    @PutMapping("/customers/{customerId}")
    public CustomerDTO updateCustomer(@PathVariable Long customerId,@RequestBody CustomerDTO customerDTO){
        customerDTO.setId(customerId);
        return bankAccountService.updateCustomer(customerDTO);
    }

    @DeleteMapping("/customers/{id}")
    public void deleteCustomer(@PathVariable Long id) { bankAccountService.deleteCustomer(id); }
}
```

EBankingBackendApplication.java

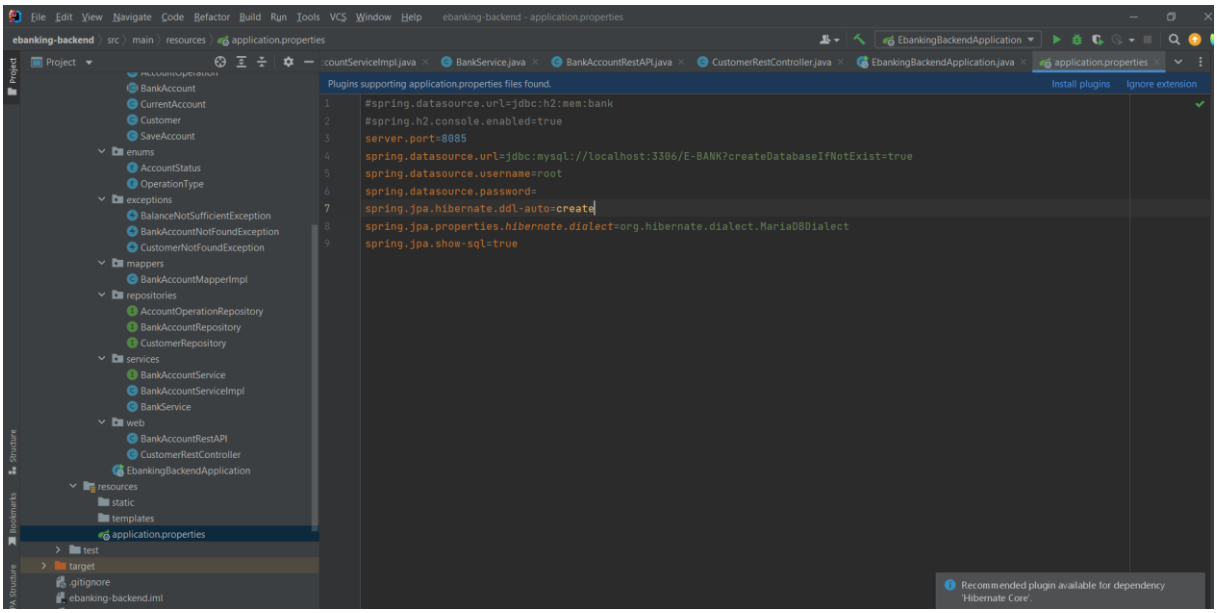


```
import java.util.stream.Stream;

@SpringBootApplication
public class EbankingBackendApplication {

    public static void main(String[] args) { SpringApplication.run(EbankingBackendApplication.class, args); }
    @Bean
    CommandLineRunner commandLineRunner(BankAccountService bankAccountService){
        return args -> {
            Stream.of("Hassan","Imane","Mohammed").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,customer.getId());
                    bankAccountService.saveSavingBankAccount( initialBalance: Math.random()*12000, overDraft: 5.5,customer.getId());

                } catch (CustomerNotFoundException e) {
                    e.printStackTrace();
                }
            });
            List<BankAccountDTO> bankAccounts= bankAccountService.bankAccountList();
            for (BankAccountDTO bankAccount:bankAccounts){
                for(int i=0;i<10;i++){
                    String accountId;
                    if(bankAccount instanceof SaveBankAccountDTO){
                        accountId=((SaveBankAccountDTO) bankAccount).getId();
```

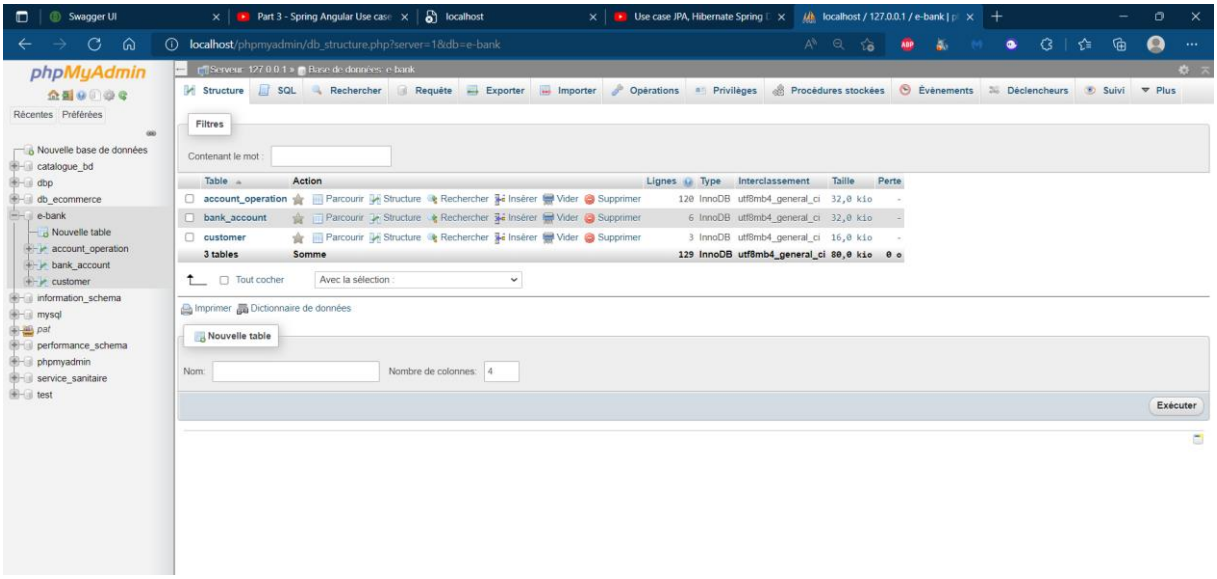


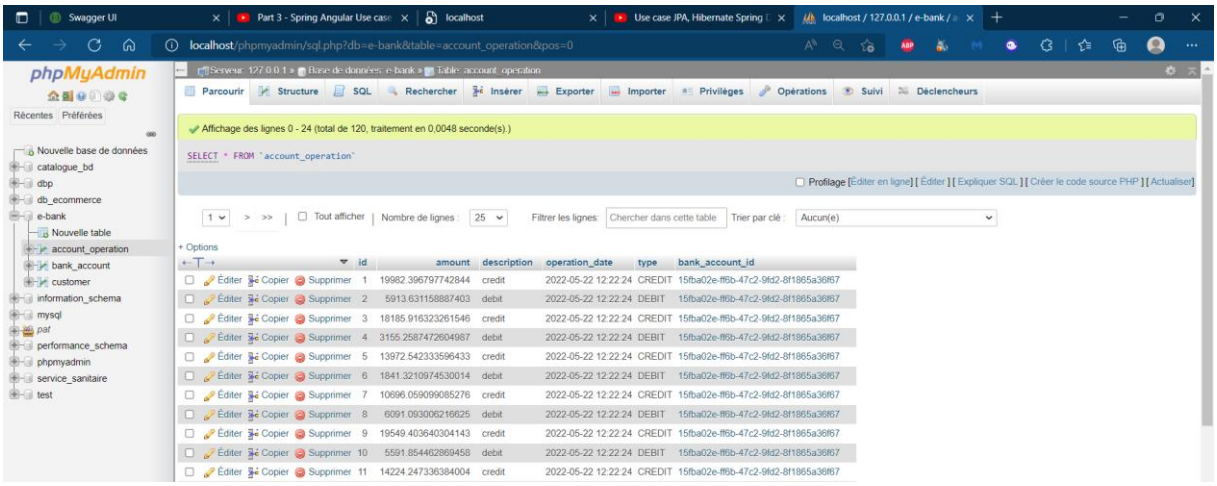


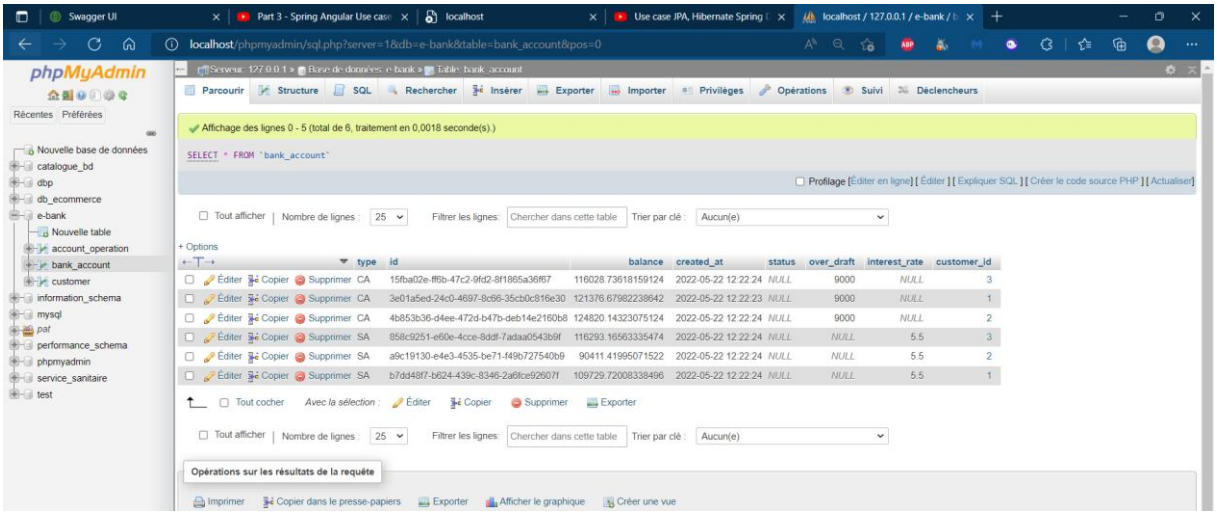
Application.properties

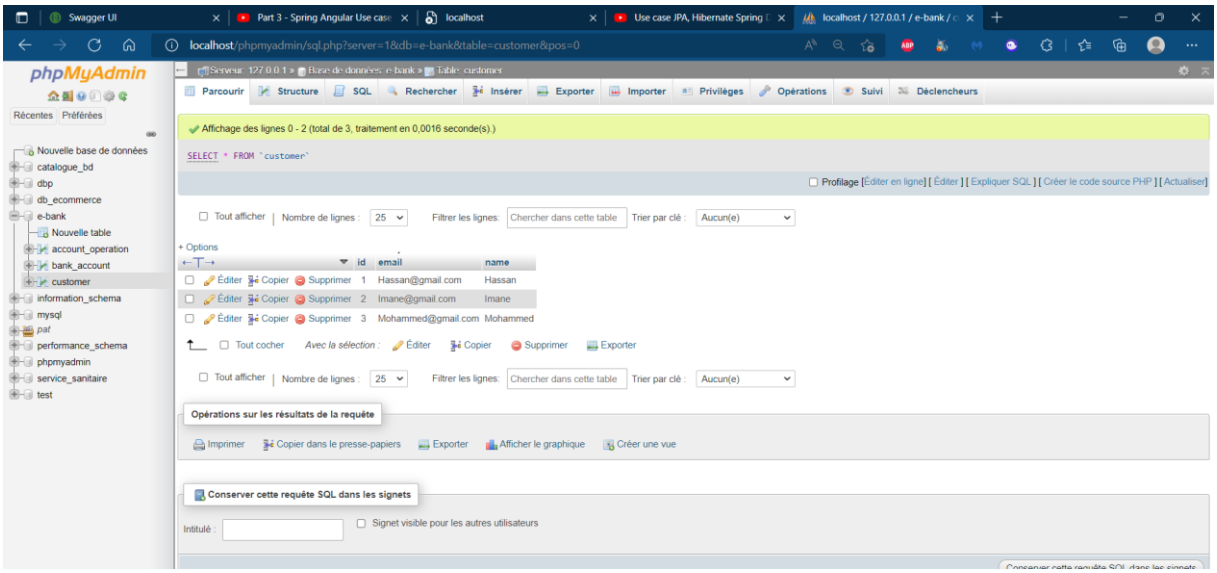


Base de données

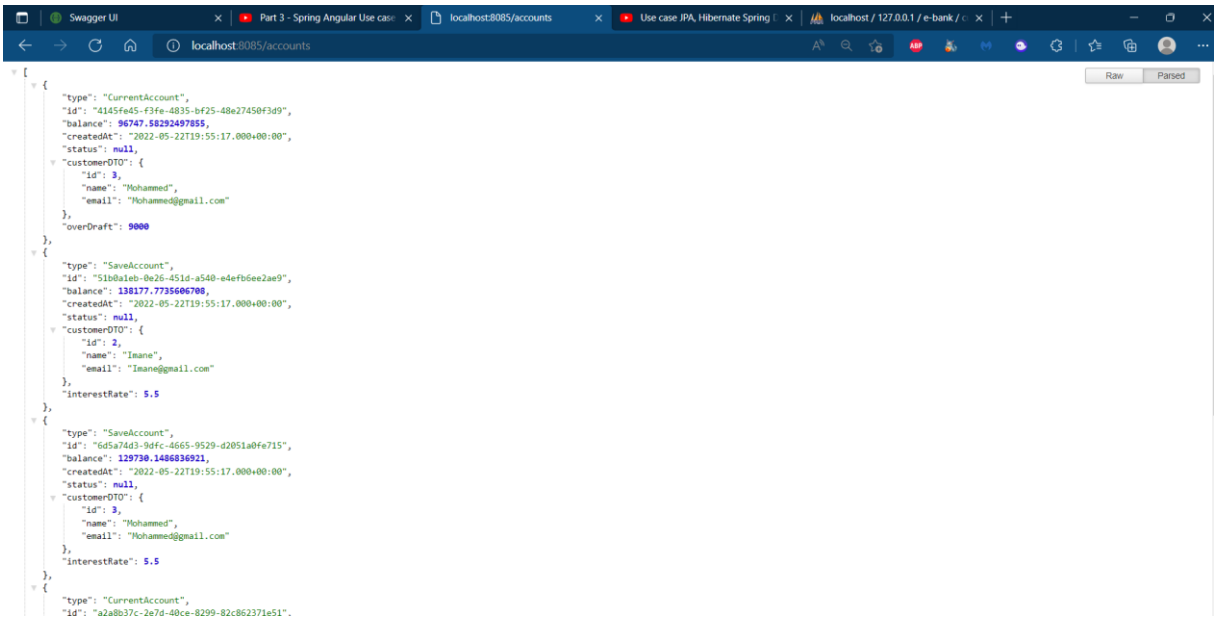



Swagger UI
Part 3 - Spring Angular Use case
localhost
Use case JPA, Hibernate Spring
localhost / 127.0.0.1 / e-bank /
localhost/phpmyadmin/sql.php?db=e-bank&table=account_operation&pos=0
phpMyAdmin
Serveur: 127.0.0.1 » Base de données: e-bank » Table: account_operation
Parcourir Structure SQL Rechercher Insérer Exporter Importer Privilèges Opérations Suivi Déclencheurs
Affichage des lignes 0 - 24 (total de 120, traitement en 0,0048 seconde(s).)
SELECT * FROM `account_operation`
Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]
Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette table | Trier par clé : Aucun(e)
+ Options
id amount description operation_date type bank_account_id
1 19982.396797742844 credit 2022-05-22 12:22:24 CREDIT 15fba02e-ff6b-47c2-9fd2-8f1865a36f67
2 5913.631158887403 debit 2022-05-22 12:22:24 DEBIT 15fba02e-ff6b-47c2-9fd2-8f1865a36f67
3 18185.916323261546 credit 2022-05-22 12:22:24 CREDIT 15fba02e-ff6b-47c2-9fd2-8f1865a36f67
4 3155.2587472604987 debit 2022-05-22 12:22:24 DEBIT 15fba02e-ff6b-47c2-9fd2-8f1865a36f67
5 13972.542333596433 credit 2022-05-22 12:22:24 CREDIT 15fba02e-ff6b-47c2-9fd2-8f1865a36f67
6 1841.3210974530014 debit 2022-05-22 12:22:24 DEBIT 15fba02e-ff6b-47c2-9fd2-8f1865a36f67
7 10696.059099085276 credit 2022-05-22 12:22:24 CREDIT 15fba02e-ff6b-47c2-9fd2-8f1865a36f67
8 6091.093006216625 debit 2022-05-22 12:22:24 DEBIT 15fba02e-ff6b-47c2-9fd2-8f1865a36f67
9 19549.403640304143 credit 2022-05-22 12:22:24 CREDIT 15fba02e-ff6b-47c2-9fd2-8f1865a36f67
10 5591.854462869458 debit 2022-05-22 12:22:24 DEBIT 15fba02e-ff6b-47c2-9fd2-8f1865a36f67
11 14224.247336384004 credit 2022-05-22 12:22:24 CREDIT 15fba02e-ff6b-47c2-9fd2-8f1865a36f67

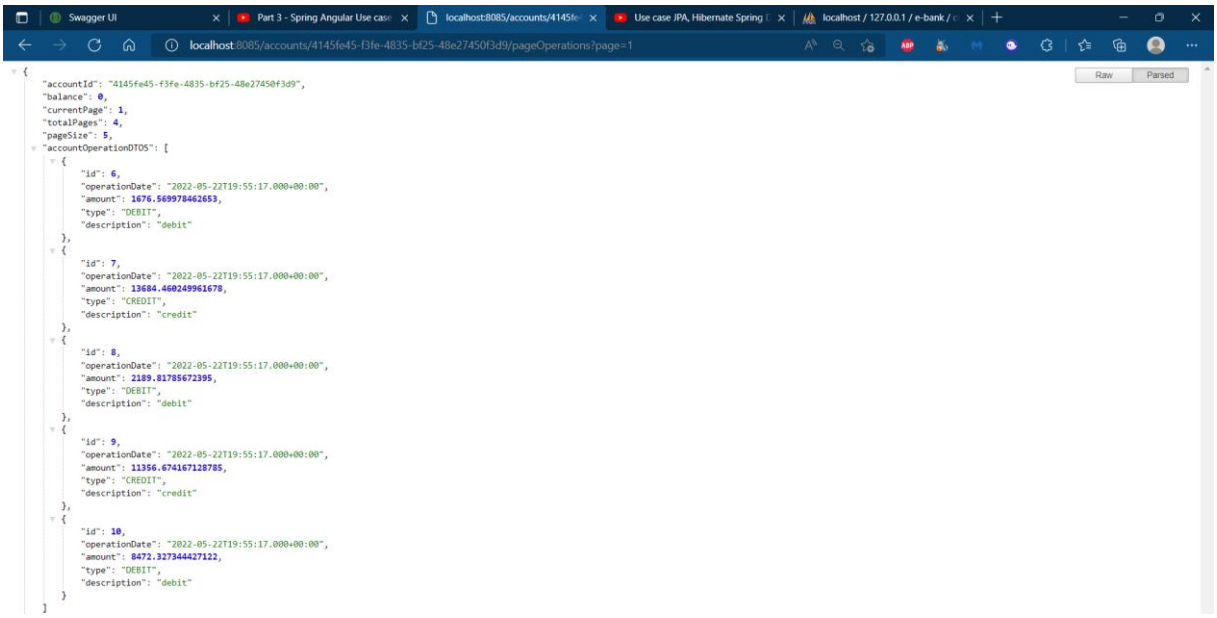

Swagger UI
Part 3 - Spring Angular Use case
localhost
Use case JPA, Hibernate Spring
localhost / 127.0.0.1 / e-bank /
localhost/phpmyadmin/sql.php?server=1&db=e-bank&table=bank_account&pos=0
phpMyAdmin
Serveur: 127.0.0.1 » Base de données: e-bank » Table: bank_account
Parcourir Structure SQL Rechercher Insérer Exporter Importer Privilèges Opérations Suivi Déclencheurs
Affichage des lignes 0 - 5 (total de 6, traitement en 0,0018 seconde(s).)
SELECT * FROM `bank_account`
Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]
Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette table | Trier par clé : Aucun(e)
+ Options
type id balance created_at status over_draft interest_rate customer_id
CA 15fba02e-ff6b-47c2-9fd2-8f1865a36f67 116028.73618159124 2022-05-22 12:22:24 NULL 9000 NULL 3
CA 3e01a5ed-24c0-4697-8c66-35cb0c816e30 121376.67982238642 2022-05-22 12:22:23 NULL 9000 NULL 1
CA 4b853b36-d4ee-472d-b47b-deb14e2160b8 124820.14323075124 2022-05-22 12:22:24 NULL 9000 NULL 2
SA 858c9251-e60e-4cce-8ddf-7adaa0543b9f 116293.16563335474 2022-05-22 12:22:24 NULL NULL 5.5 3
SA a9c19130-e4e3-4535-be71-f49b727540b9 90411.41995071522 2022-05-22 12:22:24 NULL NULL 5.5 2
SA b7dd48f7-b624-439c-8346-2a6fce92607f 109729.72008338496 2022-05-22 12:22:24 NULL NULL 5.5 1
Tout cocher Avec la sélection : Éditer Copier Supprimer Exporter
Opérations sur les résultats de la requête
Imprimer Copier dans le presse-papiers Exporter Afficher le graphique Créer une vue


Swagger UI
Part 3 - Spring Angular Use case
localhost
Use case JPA, Hibernate Spring
localhost / 127.0.0.1 / e-bank /
localhost/phpmyadmin/sql.php?server=1&db=e-bank&table=customer&pos=0
phpMyAdmin
Serveur: 127.0.0.1 » Base de données: e-bank » Table: customer
Parcourir Structure SQL Rechercher Insérer Exporter Importer Privilèges Opérations Suivi Déclencheurs
Affichage des lignes 0 - 2 (total de 3, traitement en 0,0016 seconde(s).)
SELECT * FROM `customer`
Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]
Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette table | Trier par clé : Aucun(e)
+ Options
id email name
1 Hassan@gmail.com Hassan
2 Imane@gmail.com Imane
3 Mohammed@gmail.com Mohammed
Tout cocher Avec la sélection : Éditer Copier Supprimer Exporter
Opérations sur les résultats de la requête
Imprimer Copier dans le presse-papiers Exporter Afficher le graphique Créer une vue
Conserver cette requête SQL dans les signets
Intitulé :
Signet visible pour les autres utilisateurs

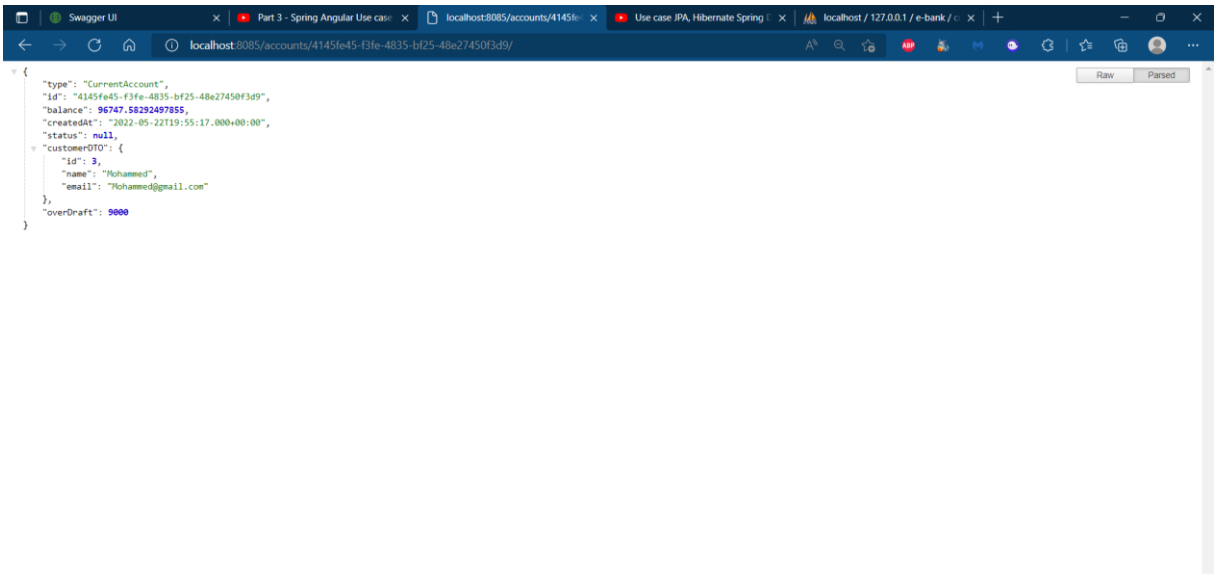
Web




```
[
  {
    "type": "CurrentAccount",
    "id": "4145fe45-f3fe-4835-bf25-48e27450f3d9",
    "balance": 96747.58292497855,
    "createdAt": "2022-05-22T19:55:17.000+00:00",
    "status": null,
    "customerDTO": {
      "id": 3,
      "name": "Mohammed",
      "email": "Mohammed@gmail.com"
    },
    "overDraft": 9000
  },
  {
    "type": "SaveAccount",
    "id": "51b0a1eb-0e26-451d-a540-e4efb6ee2ae9",
    "balance": 138177.7735606708,
    "createdAt": "2022-05-22T19:55:17.000+00:00",
    "status": null,
    "customerDTO": {
      "id": 2,
      "name": "Imane",
      "email": "Imane@gmail.com"
    },
    "interestRate": 5.5
  },
  {
    "type": "SaveAccount",
    "id": "6d5a74d3-9dfc-4665-9529-d2051a0fe715",
    "balance": 129730.1486836921,
    "createdAt": "2022-05-22T19:55:17.000+00:00",
    "status": null,
    "customerDTO": {
      "id": 3,
      "name": "Mohammed",
      "email": "Mohammed@gmail.com"
    },
    "interestRate": 5.5
  },
  {
    "type": "CurrentAccount",
    "id": "a2a8b37c-2e7d-40ce-8299-82c862371e51",
```


```
{
  "accountId": "4145fe45-f3fe-4835-bf25-48e27450f3d9",
  "balance": 0,
  "currentPage": 1,
  "totalPages": 4,
  "pageSize": 5,
  "accountOperationDTOS": [
    {
      "id": 6,
      "operationDate": "2022-05-22T19:55:17.000+00:00",
      "amount": 1676.569978462653,
      "type": "DEBIT",
      "description": "debit"
    },
    {
      "id": 7,
      "operationDate": "2022-05-22T19:55:17.000+00:00",
      "amount": 13684.460249961678,
      "type": "CREDIT",
      "description": "credit"
    },
    {
      "id": 8,
      "operationDate": "2022-05-22T19:55:17.000+00:00",
      "amount": 2189.81785672395,
      "type": "DEBIT",
      "description": "debit"
    },
    {
      "id": 9,
      "operationDate": "2022-05-22T19:55:17.000+00:00",
      "amount": 11356.674167128785,
      "type": "CREDIT",
      "description": "credit"
    },
    {
      "id": 10,
      "operationDate": "2022-05-22T19:55:17.000+00:00",
      "amount": 8472.327344427122,
      "type": "DEBIT",
      "description": "debit"
    }
  ]
```


Swagger UI
Part 3 - Spring Angular Use case
localhost:8085/accounts/4145fe
Use case JPA, Hibernate Spring
localhost / 127.0.0.1 / e-bank /
localhost:8085/accounts/4145fe45-f3fe-4835-bf25-48e27450f3d9/
Raw
Parsed
{
"type": "CurrentAccount",
"id": "4145fe45-f3fe-4835-bf25-48e27450f3d9",
"balance": 96747.58292497855,
"createdAt": "2022-05-22T19:55:17.000+00:00",
"status": null,
"customerDTO": {
"id": 3,
"name": "Mohammed",
"email": "Mohammed@gmail.com"
},
"overDraft": 9000
}