






E E-banking-App


 Project overview


 Repository


 Issues 0


 Merge requests 0


 Requirements


 CI/CD


 Security & Compliance


 Operations

 Packages & Registries


 Analytics

 **Wiki**

 Snippets

 Members

Abdellah AHBANE > E-banking-App > Wiki > Account Service

Last edited by  **Manal Outaleb** 11 months ago

Page history

New page

Account Service



All what you need to run this service is to**

- `CREATE database accounts` .
- Add database password if you have one in the `application.properties`.
- Run the Registry service eureka and then this service " account-service " .
- RabbitMQ ! pour le service transactions

If you take a look at the main application :

```
@SpringBootApplication
@EnableFeignClients
public class accountService implements CommandLineRunner {

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

    @Autowired
    CurrentAccountRepository currentAccountRepository;
    @Autowired
    SavingsAccountRepository savingsAccountRepository;
    @Override
    public void run(String... args) throws Exception {
        currentAccountRepository.save ( new CurrentAccount ( null, 76528932992L,new Date ( ), 188888d,2000000d,1.
        currentAccountRepository.save ( new CurrentAccount ( null, 7652899972L,new Date ( ), 188888d,2000000d,21.
        currentAccountRepository.findAll ().forEach ( System.out::println );

        savingsAccountRepository.save ( new SavingsAccount ( null, 76528932992L,new Date ( ), 188888d,2000000d,1.
        savingsAccountRepository.save ( new SavingsAccount ( null, 7652899972L,new Date ( ), 188888d,2000000d,21.
        savingsAccountRepository.findAll ().forEach ( System.out::println );

    }
}
```

This will simply create the table in DB and insert those data into it. But if u stop running it and re-run it again, it will show exceptions. This is because there're constraints (unique) in some columns so if you insert the same data it will not accept it simply.

So you need, either

- to drop the database by `DROP database accounts` Or
- to edit the main application by this :

```
@SpringBootApplication
@EnableFeignClients
public class AccountServiceApplication {

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

I recommend the first one, by dropping just the database especially if you have IntelliJ, it's easy and quick.

- PS. You can apply this to all my codes.

APIs of this Service :

NB: In this service: `host = http://localhost:8082/api`

The service has two accounts: `current-account` , `Cards` && `savingsAccount` please check the package `Entities` to know more about each one. We've apis for each type of account.

currentAccount :

GET :

- `host/currentAccounts` : To get the list of all currentAccounts. (`OFFICERS` and `ADMINS` are allowed to call this)
- `host/currentAccounts/{id}` : To get a currentAccount with the given id. (`ADMINS` && `OFFICERS` who have same `AgencyId` as the owner of this account && `CLIENT` owner of this account , are all allowed to call this)
- `host/currentAccounts/accountNum/{accountNum}` : To get a currentAccount by his accountNumber (`ADMINS` && `CLIENT` are allowed to call this)

 Clone repository  Edit sidebar

AA Documentation for using Gateway !

Account Service

Agency Service

Authentication Service

Customer Service

GlobalParameters Service

Officer Service

Request Service

Transaction Service

- `host/currentAccounts/{accountNum}/{accountNum}` : To get a currentAccount by his accountNumber.(**ADMINS && OFFICERS who have same AgencyId as the owner of this account && CLIENT owner of this account** , are all allowed to call this)
- `host/customer/currentAccounts/{clientId}` : To get a currentAccount by his cliendId.(**ADMINS && OFFICERS who have same AgencyId as the owner of this account && CLIENT owner of this account** , are all allowed to call this)

POST :

- `host/currentAccounts` : to create a currentAccount. (**All OFFICERS and ADMINS are allowed to call this**)

the given data should be in this format:

```
{
  "accountNum": 1111111111111111,
  "accountOpeningDate": "2020-06-14",
  "accountBalance": 900000.0,
  "accountLimit": 1000000.0,
  "clientId": 3
}
```

the client to whom you want to create an account must not have another previous account !!!!

DELETE :

- `host/currentAccounts/{id}` : To delete a currentAccount with the given id, we decide to do not delete his transactions.(**ADMINS && OFFICERS who have same AgencyId as the owner of this account ,are allowed to call this**)
- `host/customer/currentAccounts/{clientId}` : To delete a currentAccount with the given clientId, we use it in the customer service; when we delete a customer we should delete his currentAccount. (**ADMINS && OFFICERS who have same AgencyId as the owner of this account ,are allowed to call this**)

Cards of CurrentAccounts :

GET :

- `host/cards/{clientId}` : To get the list of Cards for the given ClientId. (**CLIENT with the given ID is the only one allowed to call this**)

PUT :

- `host/cards/OpnoseCard/{id}` : To oppose a card. (**ADMIN && OFFICER are allowed to call this**)

POST :

- `host/cards/MasterCard` : To create a new MasterCard Card associated to currentAccount. (**ADMINS && OFFICERS with the given ID is the only one allowed to call this**)

```
{
  "cardNumber": 111990000,
  "secretCode": 23222,
  "clientId": 1
}
```

- `host/cards/visa` : To create a new VISA Card associated to currentAccount. (**ADMINS && OFFICERS with the given ID is the only one allowed to call this**)

```
{
  "cardNumber": 111990000,
  "secretCode": 23222,
  "clientId": 1
}
```

- Depending on the endPoint called we set the `Type` && `expirationDate` (2 years from the creation date)

savingsAccount :

GET :

- `host/savingsAccounts` : To get the list of all savingsAccounts. (**OFFICERS and ADMINS are allowed to call this**)
- `host/savingsAccounts/{id}` : To get a savingsAccount with the given id. (**ADMINS && OFFICERS who have same AgencyId as the owner of this account && CLIENT owner of this account** , are all allowed to call this)
- `host/savingsAccounts/accountNum/{accountNum}` : To get a savingsAccount by his accountNumber. (**ADMINS && OFFICERS who have same AgencyId as the owner of this account && CLIENT owner of this account** , are all allowed to call this)
- `host/customer/savingsAccounts/{clientId}` : To get a savingsAccount by his cliendId.(**ADMINS && OFFICERS who have same AgencyId as the owner of this account && CLIENT owner of this account** , are all allowed to call this)

POST :

- `host/savingsAccounts` : to create a savingsAccount. (**All OFFICERS and ADMINS are allowed to call this**) the given data should be in this format:

For example :

```
{
  "accountNum": 765200893298992,
  "accountOpeningDate": "2020-05-16",
  "accountBalance": 188888,
  "accountLimit": 2000000,
  "clientId": 1,
  "rate": 0.65
}
```

DELETE :

- `host/savingsAccounts/{id}` : To delete a savingsAccount with the given id.(**ADMINs && OFFICERs who have same AgencyId as the owner of this account ,are allowed to call this**)
- `host/customer/savingsAccounts/{clientId}` : To delete a savingsAccount with the given clientId, we use it in the customer service; when we delete a customer we should delete his savingsAccount. (**ADMINs && OFFICERs who have same AgencyId as the owner of this account ,are allowed to call this**)