ECOLE NORMALE SUPÉRIEURE DE
L'ENSEIGNEMENT TECHNIQUE DE MOHAMMEDIA

UNIVERSITÉ HASSAN II DE CASABLANCA

ENSET

المدرسة العليا لأساتذة التعليم التقني
المحمدية
جامعة الحسن الثاني بالدار البيضاء

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Activité Pratique N°6

Filière :
« Génie du Logiciel et des Systèmes Informatiques Distribués »
GLSID

Mise en œuvre d'un Micro-service

Réalisé par :

Najat ES-SAYYAD

Année Universitaire : 2022-2023

Introduction

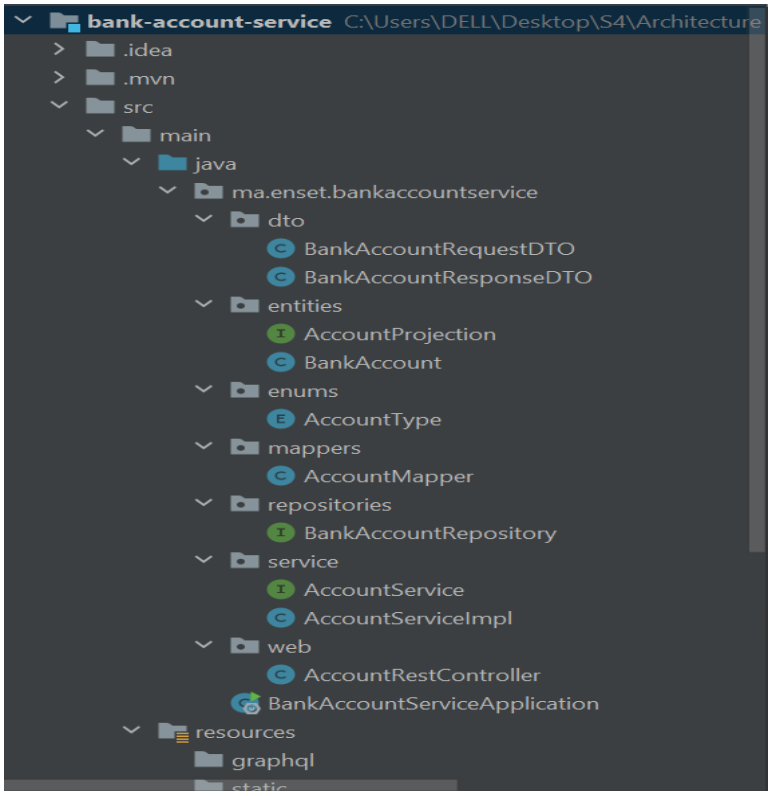
Dans ce travail pratique, nous allons développer un microservice de gestion de comptes bancaires en utilisant Spring Boot. Notre microservice utilise les dépendances Web, Spring Data JPA, H2 et Lombok. Nous avons également intégré Swagger pour générer la documentation de notre API RESTful.

Nous allons suivre un ensemble d'étapes pour créer ce microservice, notamment la création d'une entité JPA Compte, la création d'une interface CompteRepository basée sur Spring Data, la couche DAO de test, la création de méthodes CRUD pour notre entité Compte, et la création de la couche de service métier pour notre application.

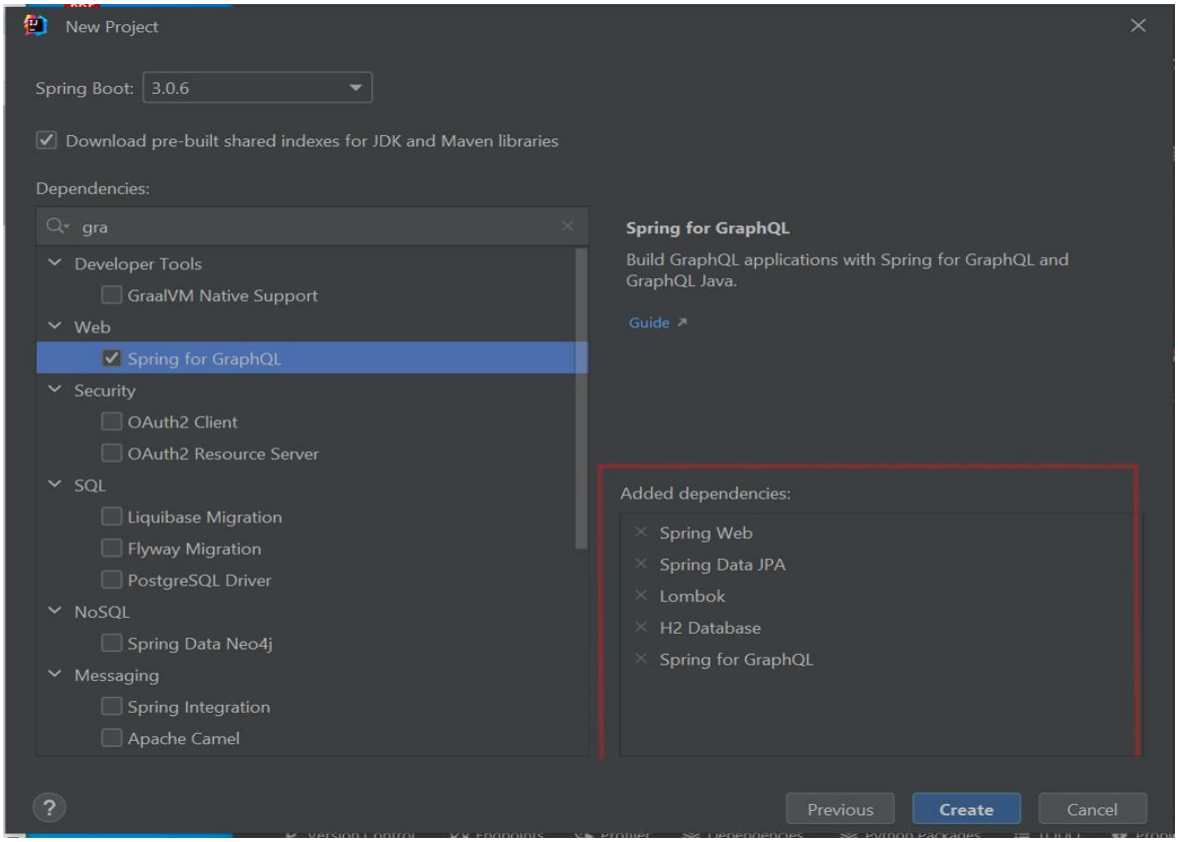
Nous allons également utiliser Spring Data Rest pour exposer une API RESTful en utilisant des projections, ce qui facilite l'accès aux données. Nous allons créé des DTOs et des Mappers pour faciliter la manipulation des données.

Enfin, nous allons tester notre microservice en utilisant un client REST comme Postman pour nous assurer que tout fonctionnait correctement. La documentation Swagger générée nous va permettre de visualiser toutes les méthodes de notre API RESTful.

Structure du projet :



1. Créer un projet Spring Boot avec les dépendances Web, Spring Data JPA, H2, Lombok



2. Créer l'entité JPA BankAccount

L'entité JPA BankAccount ayant les attributs :

- id de type String
- createdAt de type Date
- balance de type Double
- currency de type String
- type de type enum AccountType

```java
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
@Builder
public class BankAccount {
    no usages
    @Id
    private String id ;
    no usages
    private Date createdAt;
    no usages
    private Double balance;
    no usages
    private String currency;
    no usages
    @Enumerated(EnumType.STRING)
    private AccountType type ;
}
```

AccountType

```java
package ma.enset.bankaccountservice.enums;

usages
public enum AccountType {
    1 usage
    CURRENT_ACCOUNT, SAVING_ACCOUNT
}
```

3. Créer l'interface BankAccountRepository basée sur Spring Data

```java
usages
public interface BankAccountRepository extends JpaRepository<BankAccount, String> {
}
```

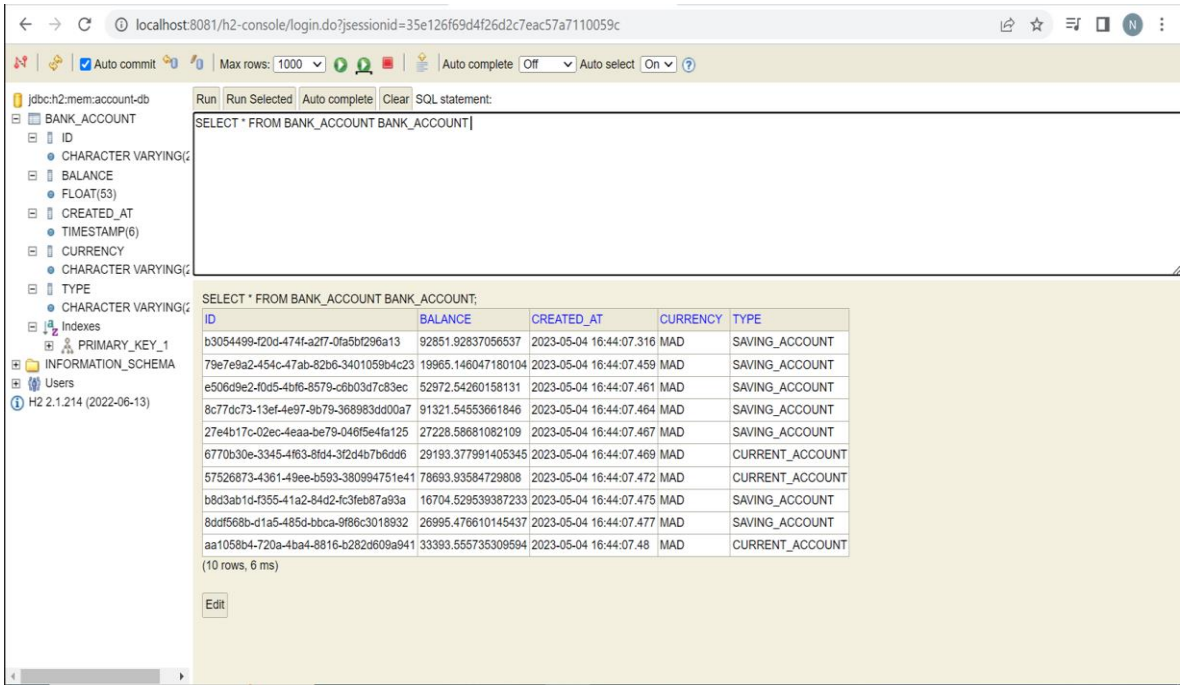
Configurer l'unité de persistance dans le ficher application.properties

```
spring.datasource.url=jdbc:h2:mem:account-db
spring.h2.console.enabled=true
server.port=8081
```

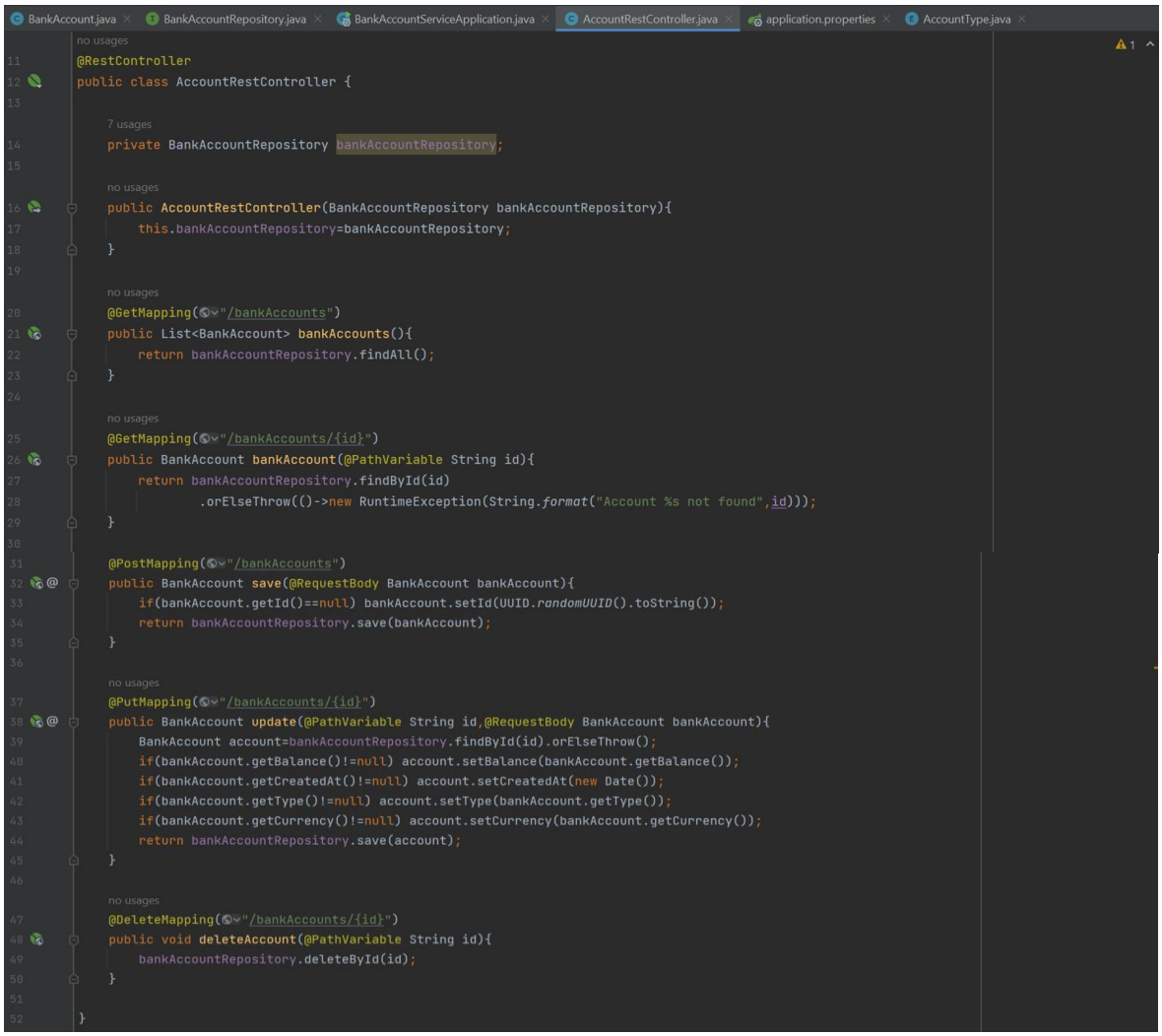
4. Tester la couche DAO

```
@SpringBootApplication
public class BankAccountServiceApplication {

    no usages
    public static void main(String[] args) {
        SpringApplication.run(BankAccountServiceApplication.class, args);
    }

    no usages
    @Bean
    CommandLineRunner start(BankAccountRepository bankAccountRepository){
        return args -> {
            for (int i = 0; i < 10; i++) {
                BankAccount bankAccount=BankAccount.builder()
                        .id(UUID.randomUUID().toString())
                        .type(Math.random()>0.5? AccountType.CURRENT_ACCOUNT:AccountType.SAVING_ACCOUNT)
                        .balance(10000+Math.random()*90000)
                        .createdAt(new Date())
                        .currency("MAD")
                        .build();
                bankAccountRepository.save(bankAccount);
            }
        };
    }
}
```

La base de données sous H2 :

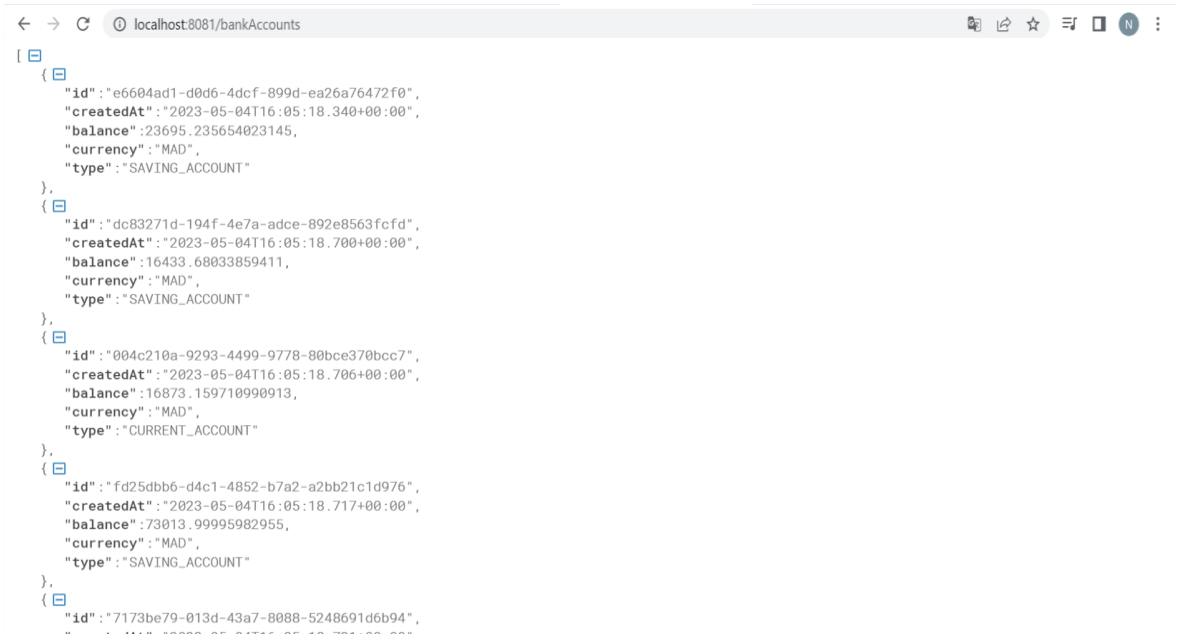


5. Créer le Web service Restfull qui permet de gérer des comptes



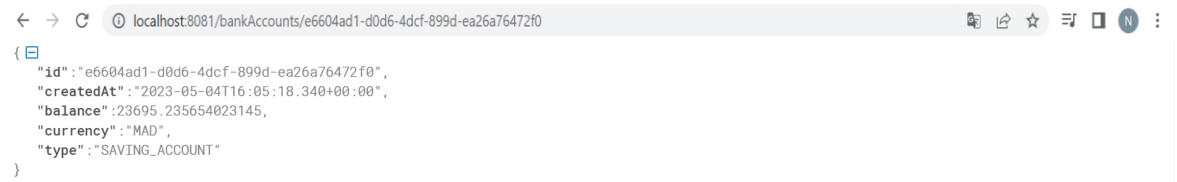
```java
@RestController
public class AccountRestController {

    private BankAccountRepository bankAccountRepository;

    public AccountRestController(BankAccountRepository bankAccountRepository){
        this.bankAccountRepository=bankAccountRepository;
    }

    @GetMapping("/bankAccounts")
    public List<BankAccount> bankAccounts(){
        return bankAccountRepository.findAll();
    }

    @GetMapping("/bankAccounts/{id}")
    public BankAccount bankAccount(@PathVariable String id){
        return bankAccountRepository.findById(id)
                .orElseThrow(()->new RuntimeException(String.format("Account %s not found",id)));
    }

    @PostMapping("/bankAccounts")
    public BankAccount save(@RequestBody BankAccount bankAccount){
        if(bankAccount.getId()==null) bankAccount.setId(UUID.randomUUID().toString());
        return bankAccountRepository.save(bankAccount);
    }

    @PutMapping("/bankAccounts/{id}")
    public BankAccount update(@PathVariable String id,@RequestBody BankAccount bankAccount){
        BankAccount account=bankAccountRepository.findById(id).orElseThrow();
        if(bankAccount.getBalance()!=null) account.setBalance(bankAccount.getBalance());
        if(bankAccount.getCreatedAt()!=null) account.setCreatedAt(new Date());
        if(bankAccount.getType()!=null) account.setType(bankAccount.getType());
        if(bankAccount.getCurrency()!=null) account.setCurrency(bankAccount.getCurrency());
        return bankAccountRepository.save(account);
    }

    @DeleteMapping("/bankAccounts/{id}")
    public void deleteAccount(@PathVariable String id){
        bankAccountRepository.deleteById(id);
    }

}
```

Résultat :

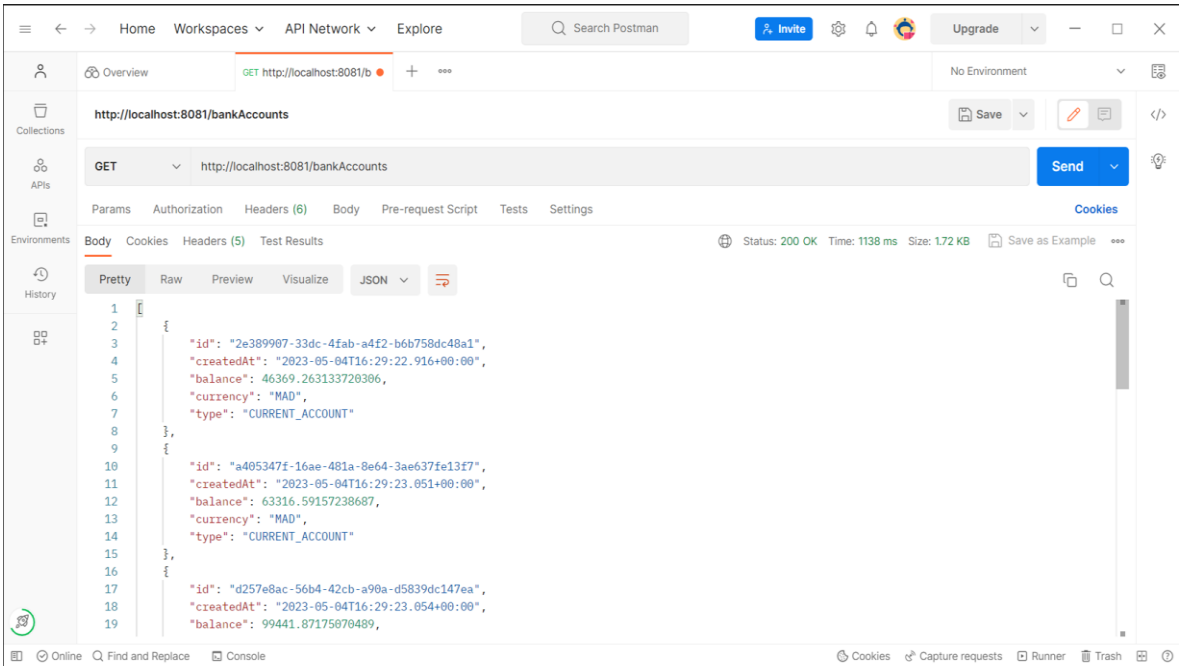


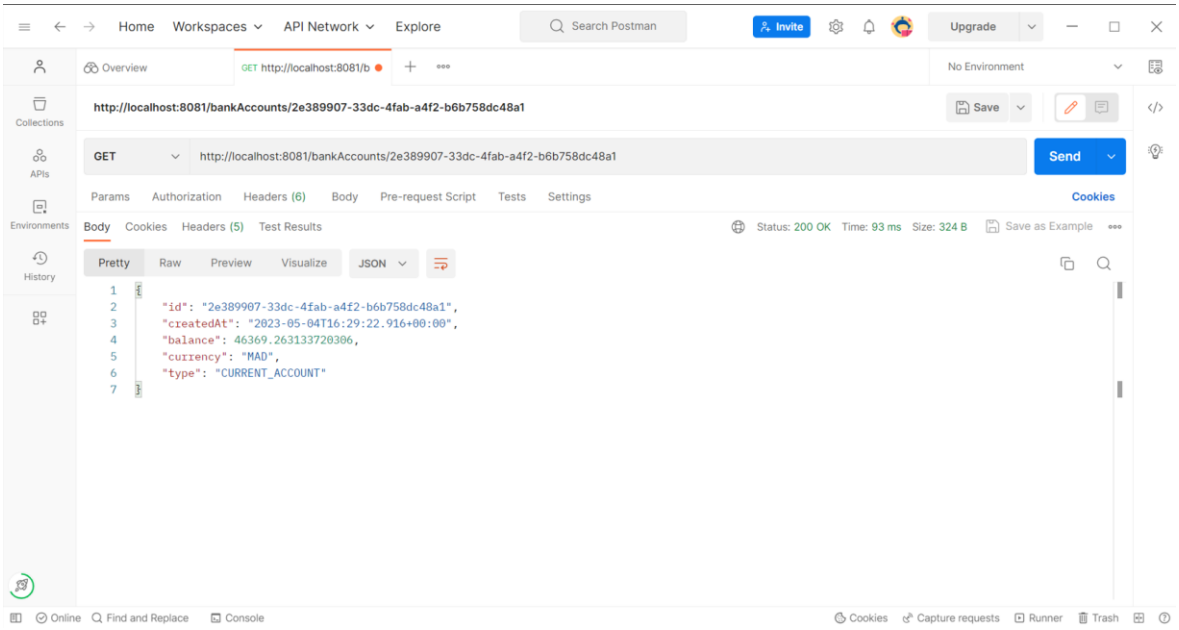
localhost:8081/bankAccounts

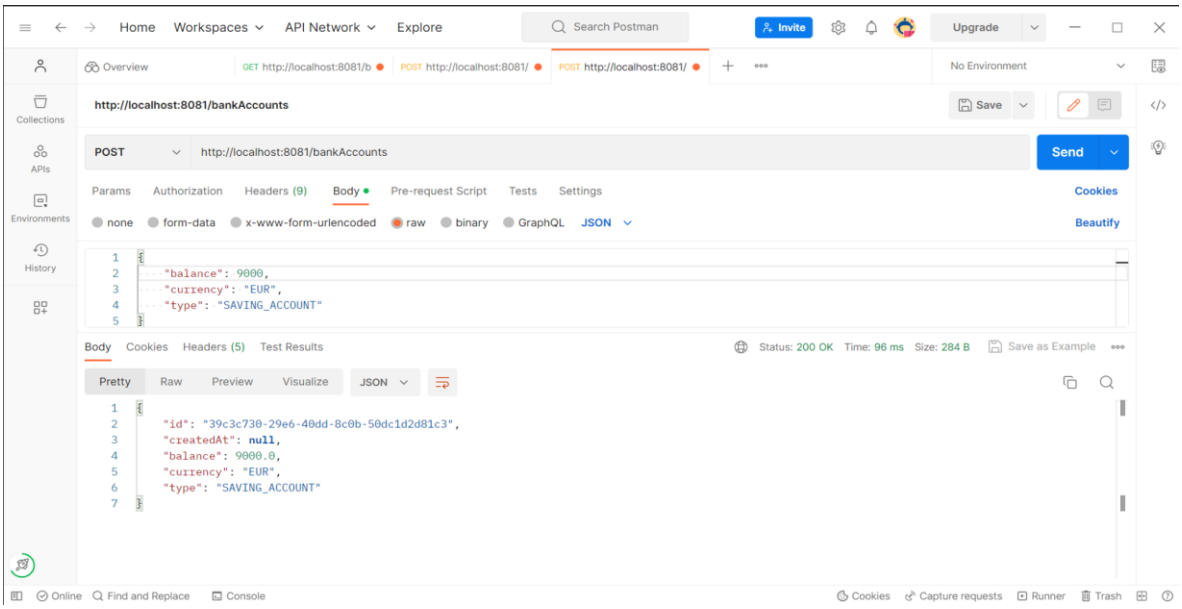

```
[
  {
    "id":"e6604ad1-d0d6-4dcf-899d-ea26a76472f0",
    "createdAt":"2023-05-04T16:05:18.340+00:00",
    "balance":23695.235654023145,
    "currency":"MAD",
    "type":"SAVING_ACCOUNT"
  },
  {
    "id":"dc83271d-194f-4e7a-adce-892e8563fcfd",
    "createdAt":"2023-05-04T16:05:18.700+00:00",
    "balance":16433.68033859411,
    "currency":"MAD",
    "type":"SAVING_ACCOUNT"
  },
  {
    "id":"004c210a-9293-4499-9778-80bce370bcc7",
    "createdAt":"2023-05-04T16:05:18.706+00:00",
    "balance":16873.159710990913,
    "currency":"MAD",
    "type":"CURRENT_ACCOUNT"
  },
  {
    "id":"fd25dbb6-d4c1-4852-b7a2-a2bb21c1d976",
    "createdAt":"2023-05-04T16:05:18.717+00:00",
    "balance":73013.99995982955,
    "currency":"MAD",
    "type":"SAVING_ACCOUNT"
  },
  {
    "id":"7173be79-013d-43a7-8088-5248691d6b94",
```

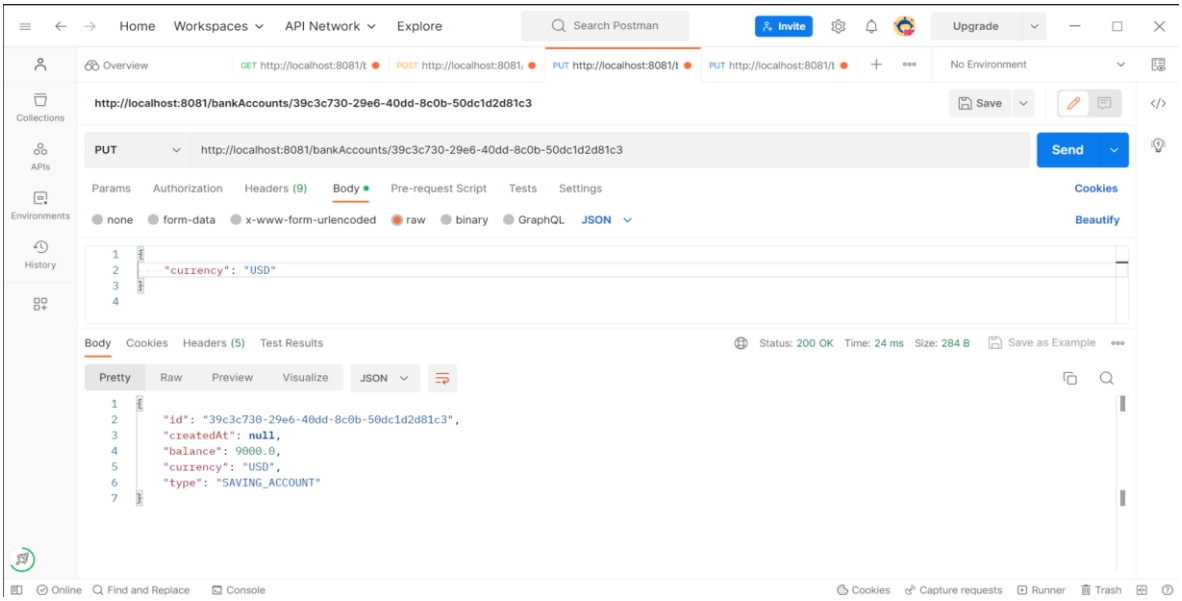


6. Tester le web micro-service en utilisant un client REST comme Postman









7. Générer et tester le documentation Swagger de des API Rest du Web service

```xml
    <dependency>
        <groupId>org.springdoc</groupId>
        <artifactId>springdoc-openapi-ui</artifactId>
        <version>1.7.0</version>
    </dependency>
```

8. Exposer une API Restful en utilisant Spring Data Rest en exploitant des projections

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-rest</artifactId>
</dependency>
```

localhost:8081/bankAccounts?page=0&size=2


```
{
  "_embedded" : {
    "bankAccounts" : [ {
      "createdAt" : "2023-05-04T19:10:34.732+00:00",
      "balance" : 62072.660649666165,
      "currency" : "MAD",
      "type" : "CURRENT_ACCOUNT",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/bankAccounts/04384f98-24b3-44f8-b641-284a0dd9697d"
        },
        "bankAccount" : {
          "href" : "http://localhost:8081/bankAccounts/04384f98-24b3-44f8-b641-284a0dd9697d"
        }
      }
    }, {
      "createdAt" : "2023-05-04T19:10:34.901+00:00",
      "balance" : 85025.60710791558,
      "currency" : "MAD",
      "type" : "CURRENT_ACCOUNT",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/bankAccounts/d108bf17-b697-431d-9c87-c89affcf1b44"
        },
        "bankAccount" : {
          "href" : "http://localhost:8081/bankAccounts/d108bf17-b697-431d-9c87-c89affcf1b44"
        }
      }
    } ]
  },
  "_links" : {
    "first" : {
      "href" : "http://localhost:8081/bankAccounts?page=0&size=2"
    },
    "self" : {
      "href" : "http://localhost:8081/bankAccounts?page=0&size=2"
    },
    "next" : {
      "href" : "http://localhost:8081/bankAccounts?page=1&size=2"
    },
    "last" : {
      "href" : "http://localhost:8081/bankAccounts?page=4&size=2"
    },
    "profile" : {
      "href" : "http://localhost:8081/profile/bankAccounts"
    }
  },
  "page" : {
    "size" : 2,
    "totalElements" : 10,
    "totalPages" : 5,
    "number" : 0
  }
}
```


sitory.java | AccountService.java | AccountServiceImpl.java | BankAccountResponseDTO.java | AccountProjection.java

```java
package ma.enset.bankaccountservice.entities;

import ma.enset.bankaccountservice.enums.AccountType;
import org.springframework.data.rest.core.config.Projection;

no usages
@Projection(types= BankAccount.class, name="p1")
public interface AccountProjection {
    no usages
    public String getId();
    no usages
    public AccountType getType();
}
```

localhost:8081/bankAccounts?projection=p1


```
{
  "_embedded" : {
    "bankAccounts" : [ {
      "id" : "4418d63e-2871-44e7-a690-0dab846c324c",
      "type" : "SAVING_ACCOUNT",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/bankAccounts/4418d63e-2871-44e7-a690-0dab846c324c"
        },
        "bankAccount" : {
          "href" : "http://localhost:8081/bankAccounts/4418d63e-2871-44e7-a690-0dab846c324c{?projection}",
          "templated" : true
        }
      }
    }, {
      "id" : "c0246191-4aef-486d-83c2-d8d34e47decc",
      "type" : "SAVING_ACCOUNT",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/bankAccounts/c0246191-4aef-486d-83c2-d8d34e47decc"
        },
        "bankAccount" : {
          "href" : "http://localhost:8081/bankAccounts/c0246191-4aef-486d-83c2-d8d34e47decc{?projection}",
          "templated" : true
        }
      }
    }, {
      "id" : "d83b743d-10be-449c-adf9-6111244f4f1a",
      "type" : "CURRENT_ACCOUNT",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/bankAccounts/d83b743d-10be-449c-adf9-6111244f4f1a"
        },
        "bankAccount" : {
          "href" : "http://localhost:8081/bankAccounts/d83b743d-10be-449c-adf9-6111244f4f1a{?projection}",
          "templated" : true
        }
      }
    }, {
```

```java
package ma.enset.bankaccountservice.repositories;

import ma.enset.bankaccountservice.entities.BankAccount;
import ma.enset.bankaccountservice.enums.AccountType;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;
import org.springframework.data.rest.core.annotation.RestResource;

import java.util.List;


5 usages
@RepositoryRestResource
public interface BankAccountRepository extends JpaRepository<BankAccount, String> {
    no usages
    @RestResource(path="/byType")
    List<BankAccount> findByType(@Param("t") AccountType type);
}
```

localhost:8081/bankAccounts/search/byType?t=CURRENT_ACCOUNT


```
{
  "_embedded" : {
    "bankAccounts" : [ {
      "createdAt" : "2023-05-04T19:46:22.587+00:00",
      "balance" : 11514.18978821456,
      "currency" : "MAD",
      "type" : "CURRENT_ACCOUNT",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/bankAccounts/ebc1ea29-d36d-462d-b830-eb47743e4344"
        },
        "bankAccount" : {
          "href" : "http://localhost:8081/bankAccounts/ebc1ea29-d36d-462d-b830-eb47743e4344{?projection}",
          "templated" : true
        }
      }
    }, {
      "createdAt" : "2023-05-04T19:46:22.745+00:00",
      "balance" : 33282.52427581932,
      "currency" : "MAD",
      "type" : "CURRENT_ACCOUNT",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/bankAccounts/8ba4397c-360d-4811-9843-289ca4e9336c"
        },
        "bankAccount" : {
          "href" : "http://localhost:8081/bankAccounts/8ba4397c-360d-4811-9843-289ca4e9336c{?projection}",
          "templated" : true
        }
      }
    }, {
      "createdAt" : "2023-05-04T19:46:22.752+00:00",
      "balance" : 53850.42389399113,
      "currency" : "MAD",
      "type" : "CURRENT_ACCOUNT",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/bankAccounts/0535fe31-ca0d-4d24-a4ef-428635734a25"
        },
```


9. Créer les DTOs et Mappers

```java
@Component
public class AccountMapper {
    1 usage
    public BankAccountResponseDTO fromBankAccount(BankAccount bankAccount){

        BankAccountResponseDTO bankAccountResponseDTO=new BankAccountResponseDTO();

        BeanUtils.copyProperties(bankAccount,bankAccountResponseDTO);
        return bankAccountResponseDTO;
    }
}
```

```java
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class BankAccountRequestDTO {

    no usages
    private Double balance;
    no usages
    private String currency;
    no usages
    private AccountType type ;
}
```

```java
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class BankAccountResponseDTO {
    no usages
    private String id ;
    no usages
    private Date createdAt;
    no usages
    private Double balance;
    no usages
    private String currency;
    no usages
    private AccountType type ;
}
```

10. Créer la couche Service (métier) et du micro service

```java
public interface AccountService {
    1 usage  1 implementation
    BankAccountResponseDTO addAccount(BankAccountRequestDTO bankAccountDTO);
}
```

```java
@Service
@Transactional
public class AccountServiceImpl implements AccountService {
    2 usages
    private BankAccountRepository bankAccountRepository;
    2 usages
    private AccountMapper accountMapper;
    no usages
    public AccountServiceImpl(BankAccountRepository bankAccountRepository, AccountMapper accountMapper) {
        this.bankAccountRepository = bankAccountRepository;
        this.accountMapper = accountMapper;
    }
    1 usage
    @Override
    public BankAccountResponseDTO addAccount(BankAccountRequestDTO bankAccountDTO) {
        BankAccount bankAccount=BankAccount.builder()
                .id(UUID.randomUUID().toString())
                .createdAt(new Date())
                .balance(bankAccountDTO.getBalance())
                .type(bankAccountDTO.getType())
                .currency(bankAccountDTO.getCurrency())
                .build();
        BankAccount saveBankAccount = bankAccountRepository.save(bankAccount);
        BankAccountResponseDTO bankAccountResponseDTO=accountMapper.fromBankAccount(saveBankAccount);
        return bankAccountResponseDTO;
    }
}
```