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

RAPPORT DE PROJET DIGITAL BANKING

Filière :« Ingénierie Informatique : Big Data et Cloud Computing » II-BDCC

E-banking

Réalisé par :

Khadija BENJILALI

Encadré par :

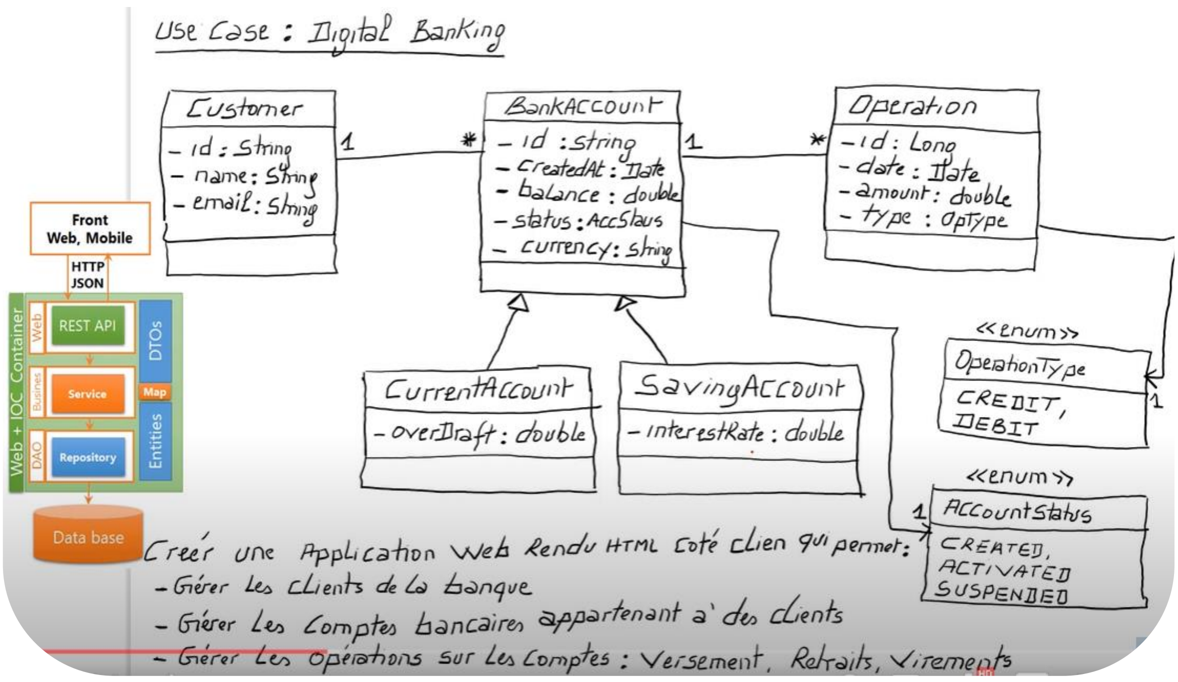
Mohamed YOUSSFI

Année Universitaire : 2021-2022

Table des matières

I. Description d'application

« E-banking » est 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.

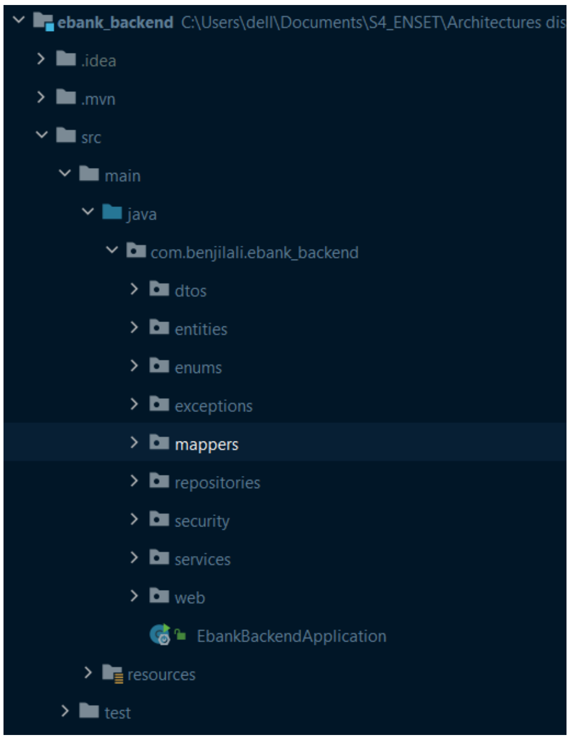


Creer une Application Web Rendu HTML coté client qui permet :
- Gérer les clients de la banque
- Gérer les Comptes bancaires appartenant à des clients
- Gérer les opérations sur les Comptes : Versement, Retraits, Virements

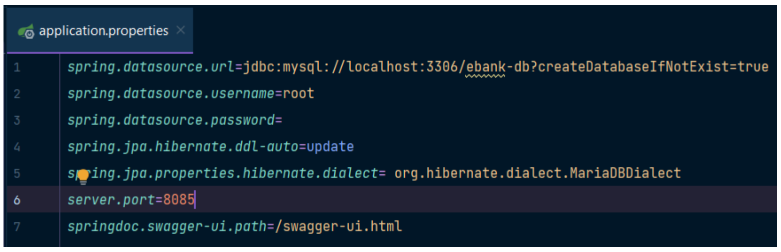
II. Backend d'application

➢ Structure du projet

L'application se compose des couches suivantes :

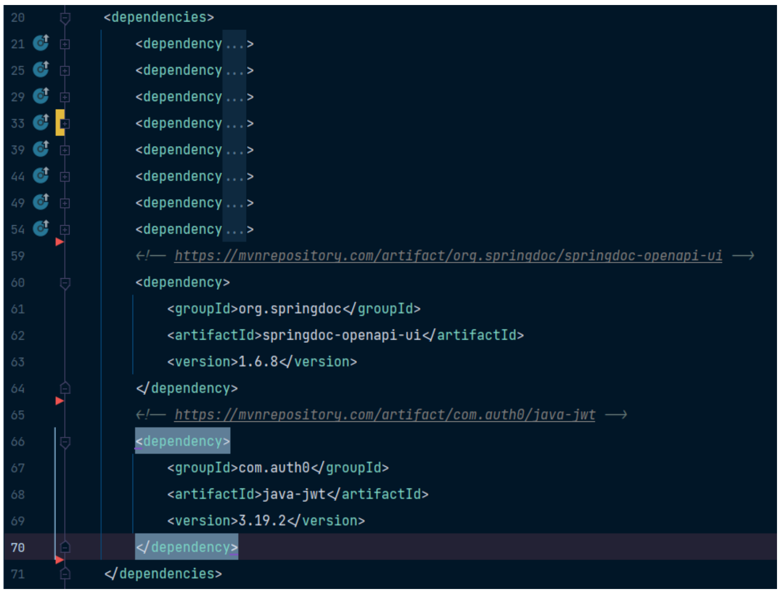


```
ebank_backend C:\Users\dell\Documents\S4_ENSET\Architectures dis
  > .idea
  > .mvn
  v src
    v main
      v java
        v com.benjilali.ebank_backend
          > dtos
          > entities
          > enums
          > exceptions
          > mappers
          > repositories
          > security
          > services
          > web
            EbankBackendApplication
      > resources
    > test
```



application.properties

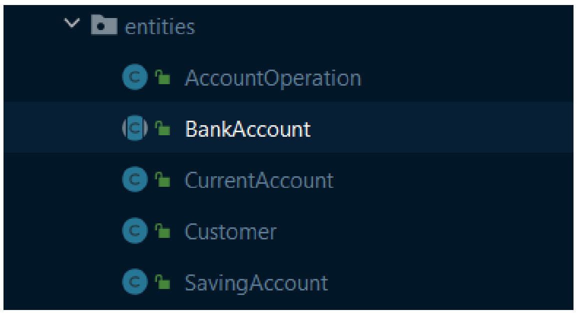
```
spring.datasource.url=jdbc:mysql://localhost:3306/ebank-db?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MariaDBDialect
server.port=8085
springdoc.swagger-ui.path=/swagger-ui.html
```



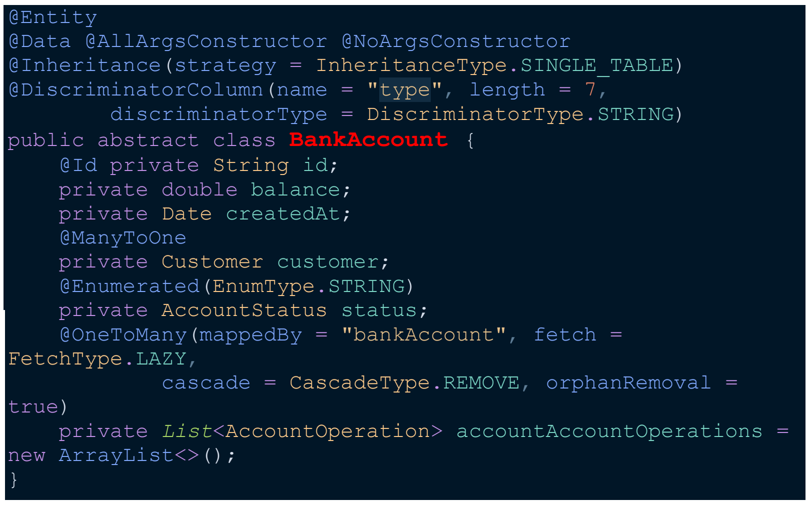
```
<dependencies>
    <dependency...>
    <dependency...>
    <dependency...>
    <dependency...>
    <dependency...>
    <dependency...>
    <dependency...>
    <dependency...>
    <!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-ui -->
    <dependency>
        <groupId>org.springdoc</groupId>
        <artifactId>springdoc-openapi-ui</artifactId>
        <version>1.6.8</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.auth0/java-jwt -->
    <dependency>
        <groupId>com.auth0</groupId>
        <artifactId>java-jwt</artifactId>
        <version>3.19.2</version>
    </dependency>
</dependencies>
```

1. Couche DAO (Entités JPA et Repositories) :

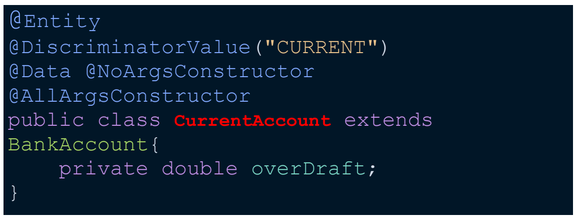
➢ Entities JPA



```
entities
    AccountOperation
    BankAccount
    CurrentAccount
    Customer
    SavingAccount
```

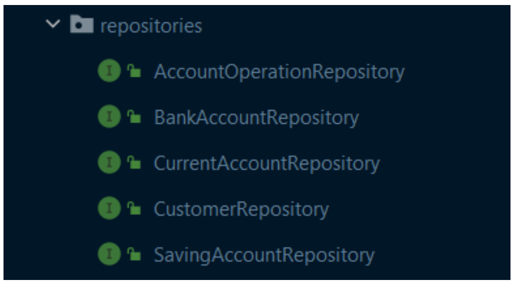


```java
@Entity
@Data @AllArgsConstructor @NoArgsConstructor
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "type", length = 7,
        discriminatorType = DiscriminatorType.STRING)
public abstract class BankAccount {
    @Id private String id;
    private double balance;
    private Date createdAt;
    @ManyToOne
    private Customer customer;
    @Enumerated(EnumType.STRING)
    private AccountStatus status;
    @OneToMany(mappedBy = "bankAccount", fetch =
FetchType.LAZY,
            cascade = CascadeType.REMOVE, orphanRemoval =
true)
    private List<AccountOperation> accountAccountOperations =
new ArrayList<>();
}
```

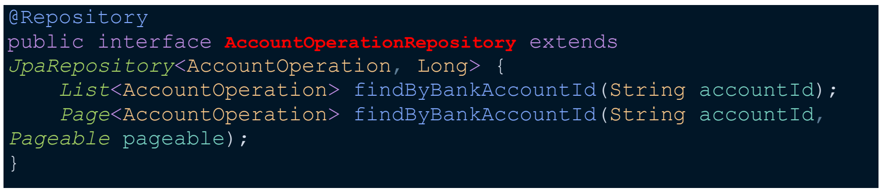


```java
@Entity
@DiscriminatorValue("CURRENT")
@Data @NoArgsConstructor
@AllArgsConstructor
public class CurrentAccount extends
BankAccount{
    private double overDraft;
}
```

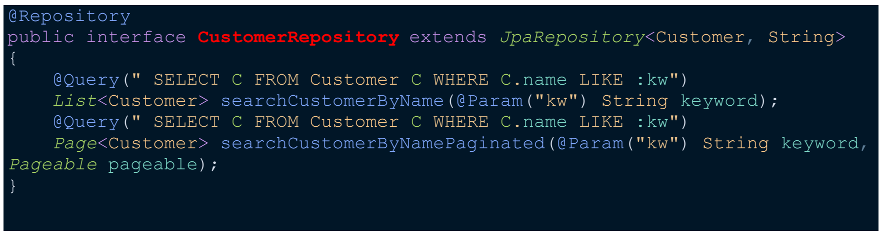
➢ Repositories



```
repositories
    AccountOperationRepository
    BankAccountRepository
    CurrentAccountRepository
    CustomerRepository
    SavingAccountRepository
```

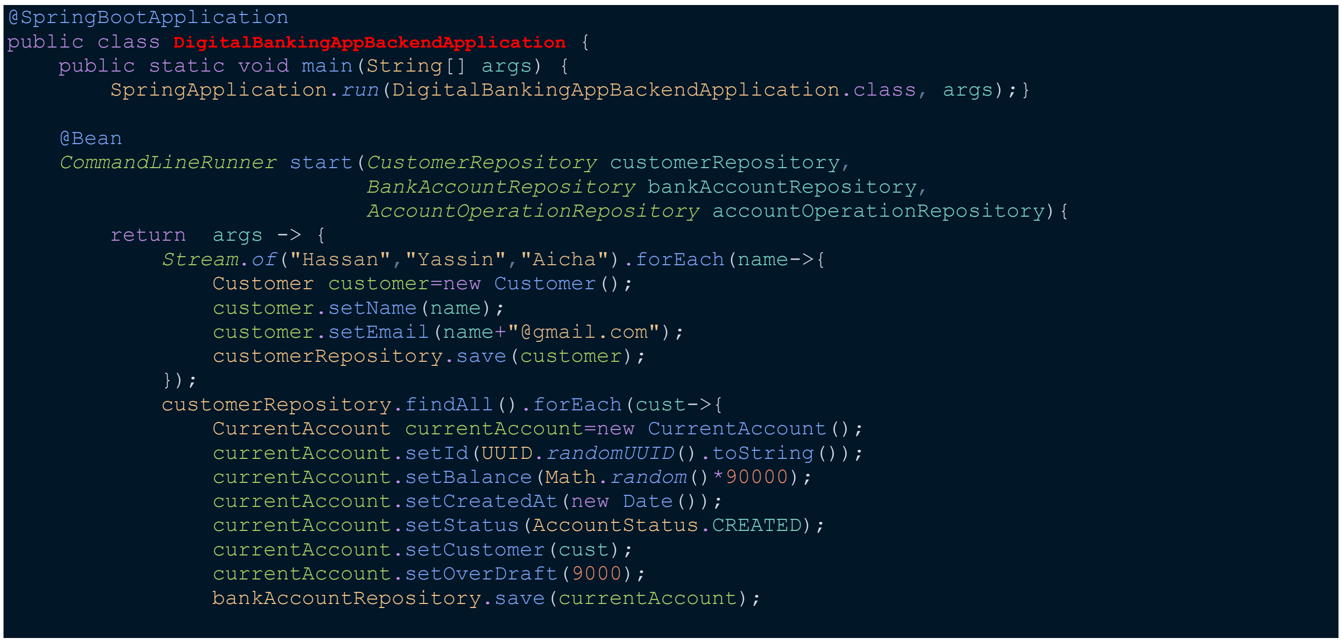


```java
@Repository
public interface AccountOperationRepository extends
JpaRepository<AccountOperation, Long> {
    List<AccountOperation> findByBankAccountId(String accountId);
    Page<AccountOperation> findByBankAccountId(String accountId,
Pageable pageable);
}
```



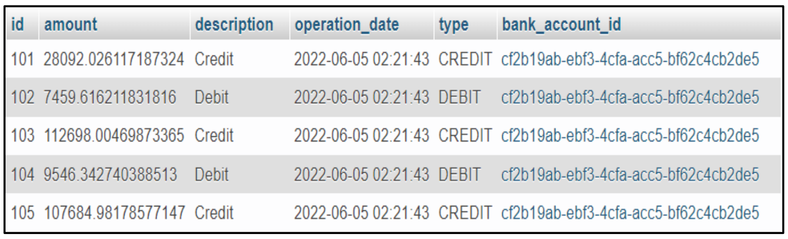
```java
@Repository
public interface CustomerRepository extends JpaRepository<Customer, String>
{
    @Query(" SELECT C FROM Customer C WHERE C.name LIKE :kw")
    List<Customer> searchCustomerByName(@Param("kw") String keyword);
    @Query(" SELECT C FROM Customer C WHERE C.name LIKE :kw")
    Page<Customer> searchCustomerByNamePaginated(@Param("kw") String keyword,
Pageable pageable);
}
```

2. Test de la couche DAO



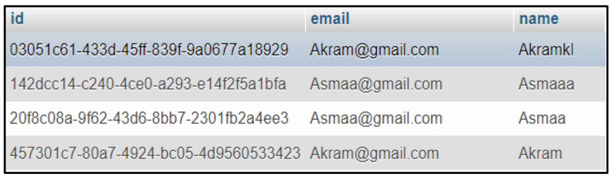
```java
@SpringBootApplication
public class DigitalBankingAppBackendApplication {
    public static void main(String[] args) {
        SpringApplication.run(DigitalBankingAppBackendApplication.class, args);}

    @Bean
    CommandLineRunner start(CustomerRepository customerRepository,
                            BankAccountRepository bankAccountRepository,
                            AccountOperationRepository accountOperationRepository){
        return  args -> {
            Stream.of("Hassan","Yassin","Aicha").forEach(name->{
                Customer customer=new Customer();
                customer.setName(name);
                customer.setEmail(name+"@gmail.com");
                customerRepository.save(customer);
            });
            customerRepository.findAll().forEach(cust->{
                CurrentAccount currentAccount=new CurrentAccount();
                currentAccount.setId(UUID.randomUUID().toString());
                currentAccount.setBalance(Math.random()*90000);
                currentAccount.setCreatedAt(new Date());
                currentAccount.setStatus(AccountStatus.CREATED);
                currentAccount.setCustomer(cust);
                currentAccount.setOverDraft(9000);
                bankAccountRepository.save(currentAccount);
            SavingAccount savingAccount=new SavingAccount();
            savingAccount.setId(UUID.randomUUID().toString());
            savingAccount.setBalance(Math.random()*90000);
            savingAccount.setCreatedAt(new Date());
            savingAccount.setStatus(AccountStatus.CREATED);
            savingAccount.setCustomer(cust);
            savingAccount.setInterestRate(5.5);
            bankAccountRepository.save(savingAccount);
        });

        bankAccountRepository.findAll().forEach(acc->{
            for(int i=0; i<5; i++){
                AccountOperation accountOperation = new AccountOperation();
                accountOperation.setOperationDate(new Date());
                accountOperation.setAmount(Math.random()*12000);
                accountOperation.setType(Math.random()>0.5?  OperationType.DEBIT:OperationType.CREDIT);
                accountOperation.setBankAccount(acc);
                accountOperationRepository.save(accountOperation);
            }
        });

    };
  }
}
```

➢ Table AccountOperation :



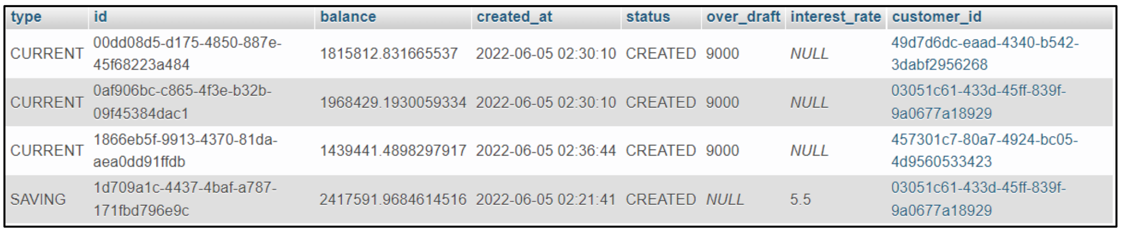
id	amount	description	operation_date	type	bank_account_id
101	28092.026117187324	Credit	2022-06-05 02:21:43	CREDIT	cf2b19ab-ebf3-4cfa-acc5-bf62c4cb2de5
102	7459.616211831816	Debit	2022-06-05 02:21:43	DEBIT	cf2b19ab-ebf3-4cfa-acc5-bf62c4cb2de5
103	112698.00469873365	Credit	2022-06-05 02:21:43	CREDIT	cf2b19ab-ebf3-4cfa-acc5-bf62c4cb2de5
104	9546.342740388513	Debit	2022-06-05 02:21:43	DEBIT	cf2b19ab-ebf3-4cfa-acc5-bf62c4cb2de5
105	107684.98178577147	Credit	2022-06-05 02:21:43	CREDIT	cf2b19ab-ebf3-4cfa-acc5-bf62c4cb2de5

➢ Table Customer :



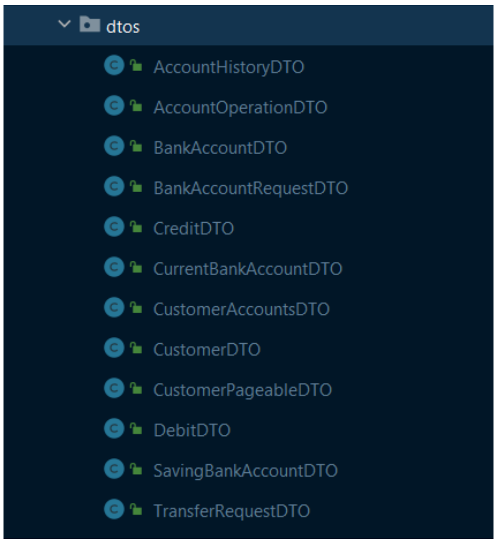
id	email	name
03051c61-433d-45ff-839f-9a0677a18929	Akram@gmail.com	Akramkl
142dcc14-c240-4ce0-a293-e14f2f5a1bfa	Asmaa@gmail.com	Asmaaa
20f8c08a-9f62-43d6-8bb7-2301fb2a4ee3	Asmaa@gmail.com	Asmaa
457301c7-80a7-4924-bc05-4d9560533423	Akram@gmail.com	Akram

➢ Table BankAccount :



type	id	balance	created_at	status	over_draft	interest_rate	customer_id
CURRENT	00dd08d5-d175-4850-887e-45f68223a484	1815812.831665537	2022-06-05 02:30:10	CREATED	9000	*NULL*	49d7d6dc-eaad-4340-b542-3dabf2956268
CURRENT	0af906bc-c865-4f3e-b32b-09f45384dac1	1968429.1930059334	2022-06-05 02:30:10	CREATED	9000	*NULL*	03051c61-433d-45ff-839f-9a0677a18929
CURRENT	1866eb5f-9913-4370-81da-aea0dd91ffdb	1439441.4898297917	2022-06-05 02:36:44	CREATED	9000	*NULL*	457301c7-80a7-4924-bc05-4d9560533423
SAVING	1d709a1c-4437-4baf-a787-171fbd796e9c	2417591.9684614516	2022-06-05 02:21:41	CREATED	*NULL*	5.5	03051c61-433d-45ff-839f-9a0677a18929

3. La couche DTO



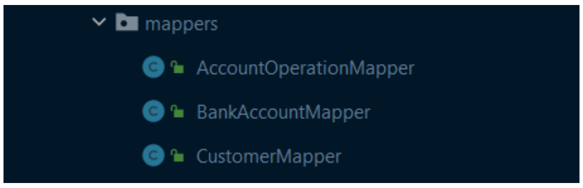
```
dtos
    AccountHistoryDTO
    AccountOperationDTO
    BankAccountDTO
    BankAccountRequestDTO
    CreditDTO
    CurrentBankAccountDTO
    CustomerAccountsDTO
    CustomerDTO
    CustomerPageableDTO
    DebitDTO
    SavingBankAccountDTO
    TransferRequestDTO
```

```
@AllArgsConstructor
@NoArgsConstructor
@Data
public class AccountOperationDTO {
    private long id;
    private Date operationDate;
    private double amount;
    private OperationType type;
    private String description;
}
```

```
@Data
public class CustomerDTO {
    private String id;
    private String name;
    private String email;
}
```

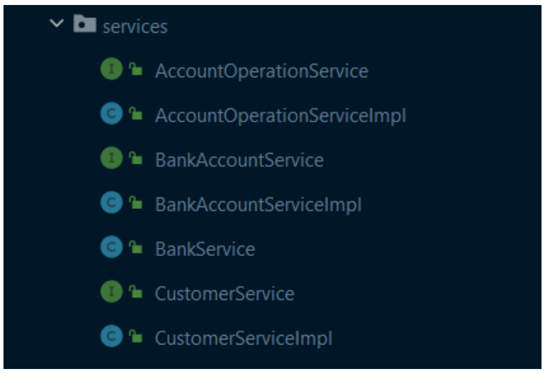
```
@Data @AllArgsConstructor @NoArgsConstructor
public class CreditDTO {
    private String accountId;
    private double amount;
    private String description;
}
```

4. Mappers (DTO <=>Entities)


mappers
AccountOperationMapper
BankAccountMapper
CustomerMapper

```
@Service @AllArgsConstructor
public class BankAccountMapper {
    private CustomerMapper customerMapper;

    public SavingAccount fromSavingAccountDTO(SavingBankAccountDTO savingBankAccountDTO){
        SavingAccount savingAccount = new SavingAccount();
        BeanUtils.copyProperties( savingBankAccountDTO, savingAccount);
        savingAccount.setCustomer( customerMapper.fromCustomerDto( savingBankAccountDTO.getCustomer() ) );
        return savingAccount;
    }

    public SavingBankAccountDTO fromSavingAccount(SavingAccount savingBankAccount){
        SavingBankAccountDTO savingAccountDto = new SavingBankAccountDTO();
        BeanUtils.copyProperties( savingBankAccount, savingAccountDto);
        savingAccountDto.setCustomer( customerMapper.fromCustomer( savingBankAccount.getCustomer() ) );
        savingAccountDto.setType( savingBankAccount.getClass().getSimpleName() );
        return savingAccountDto;
    }

    public CurrentAccount fromCurrentAccountDTO(CurrentBankAccountDTO currentBankAccountDTO){
        CurrentAccount currentAccount = new CurrentAccount();
        BeanUtils.copyProperties( currentBankAccountDTO, currentAccount);
        currentAccount.setCustomer( customerMapper.fromCustomerDto( currentBankAccountDTO.getCustomer() ) );
        return currentAccount;
    }

    public CurrentBankAccountDTO fromCurrentAccount(CurrentAccount currentAccount){
        CurrentBankAccountDTO currentAccountDTO = new CurrentBankAccountDTO();
        BeanUtils.copyProperties( currentAccount, currentAccountDTO);
        currentAccountDTO.setCustomer( customerMapper.fromCustomer( currentAccount.getCustomer() ) );
        currentAccountDTO.setType( currentAccount.getClass().getSimpleName() );
        return currentAccountDTO;
    }

    public SavingAccount savingAccountFromBankAccountRequestDto(BankAccountRequestDTO bankAccountRequestDTO){
        SavingAccount savingAccount = new SavingAccount();
        BeanUtils.copyProperties( bankAccountRequestDTO, savingAccount);
        savingAccount.setCustomer( customerMapper.fromCustomerDto( bankAccountRequestDTO.getCustomer() ) );
        return savingAccount;
    }

    public CurrentAccount currentAccountFromBankAccountRequestDto(BankAccountRequestDTO bankAccountRequestDTO){
        CurrentAccount currentAccount = new CurrentAccount();
        BeanUtils.copyProperties( bankAccountRequestDTO, currentAccount);
        currentAccount.setCustomer( customerMapper.fromCustomerDto( bankAccountRequestDTO.getCustomer() ) );
        return currentAccount;
    }
}
```

5. Couche Service définissant les opérations suivantes :


services
AccountOperationService
AccountOperationServiceImpl
BankAccountService
BankAccountServiceImpl
BankService
CustomerService
CustomerServiceImpl

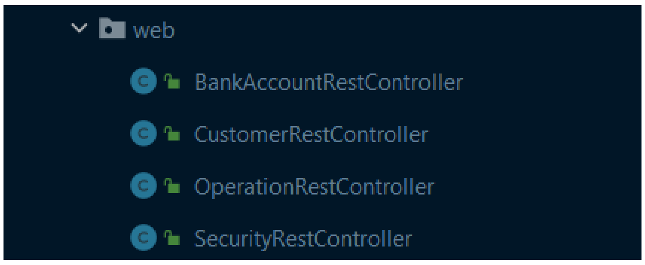
- Ajouter des comptes
- Ajouter des clients
- Effectuer un débit (Retrait)
- Effectuer un crédit (Versement)
- Effectuer un virement
- Consulter un compte

```java
public interface BankAccountService {
    CurrentBankAccountDTO saveCurrentBankAccount(double initialBalance, double overDraft, String customerId) throws
CustomerNotFoundException;
    BankAccountDTO updateCurrentBankAccount(CurrentAccount currentAccount) throws BankAccountNotFoundExcetion;
    SavingBankAccountDTO saveSavingBankAccount(double initialBalance, double interestRate, String customerId) throws
CustomerNotFoundException;
    BankAccountDTO updateSavingBankAccount(SavingAccount savingAccount) throws BankAccountNotFoundExcetion;
    BankAccountDTO getBankAccount(String bankAccountId) throws BankAccountNotFoundExcetion;
    void debit( String accountId, double amount, String description) throws BankAccountNotFoundExcetion,
BalanceNotSufficientException;
    void credit( String accountId, double amount, String description) throws BankAccountNotFoundExcetion;
    void transfer( String accountSourceId, String accountDestinationId, double amount, String description) throws
BankAccountNotFoundExcetion, BalanceNotSufficientException;
    List<BankAccountDTO> listBankAccountDTO();
    void deleteAccount(String accountId);
}
```

➢ Implémentation de l'interface BankAccountService :

```java
@Service @Transactional @AllArgsConstructor
@Slf4j // equals adding this line :     private Logger log = LoggerFactory.getLogger(this.getClass());
public class BankAccountServiceImpl implements BankAccountService {
    private CustomerRepository customerRepository;
    private BankAccountRepository accountRepository;
    private AccountOperationRepository operationRepository;
    private BankAccountMapper bankAccountMapper;
    private CustomerMapper customerMapper;

    @Override
    public CurrentBankAccountDTO saveCurrentBankAccount(double initialBalance, double overDraft, String
customerId)
            throws CustomerNotFoundException {
        log.info("Checking if customer exists... ");
        Customer customer = customerRepository.findById(customerId).orElse(null);
        if (customer == null)
            throw new CustomerNotFoundException(" Customer not found !! ");
        log.info("✓ Customer found !");

        CurrentAccount currentBankAccount;
        log.info("creating Current-bank-account ... ");
        currentBankAccount = new CurrentAccount();
        currentBankAccount.setId(UUID.randomUUID().toString());
```

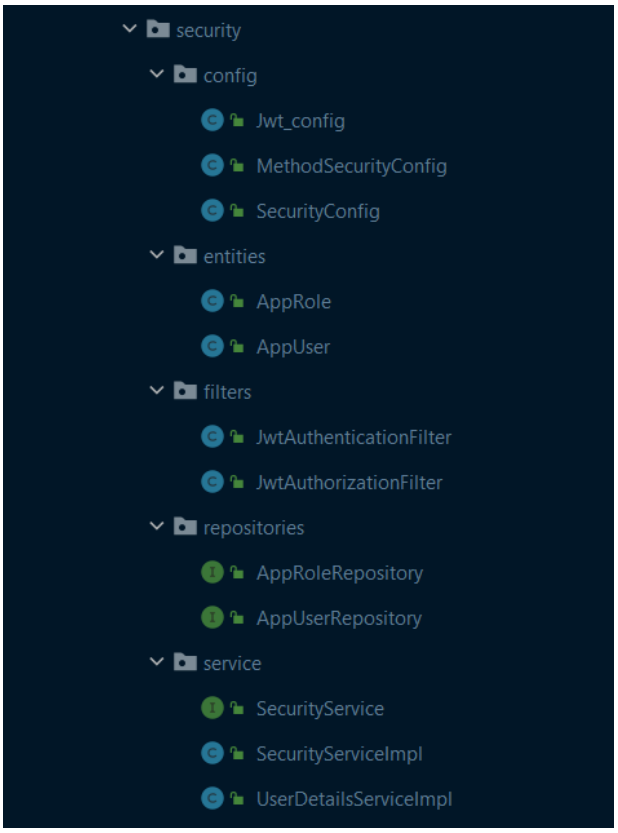
6. La couche Web (Rest Controllers)



```
web
    BankAccountRestController
    CustomerRestController
    OperationRestController
    SecurityRestController
```

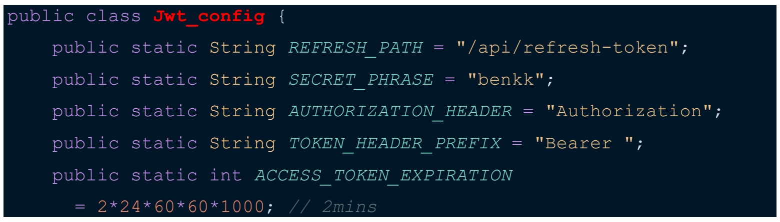
```java
@RestController @AllArgsConstructor
@Slf4j @RequestMapping("/api")
@CrossOrigin(origins = "*")
public class CustomerRestController {
    private CustomerService customerService;

    @PreAuthorize("hasAuthority('USER')")
    @GetMapping("/customers")
    public List<CustomerDTO> getCustomers(){
        return customerService.listCustomers(); }

    @PreAuthorize("hasAuthority('USER')")
    @GetMapping("/customers/paginated")
    public CustomerPageableDTO getPaginatedCustomers(
            @RequestParam(name = "page", defaultValue = "0") int page,
            @RequestParam(name = "size", defaultValue = "10") int size
    ){
        return customerService.paginateCustomers( page, size);
    }

    @PreAuthorize("hasAuthority('USER')")
    @GetMapping("/customers/paginated/search")
    public CustomerPageableDTO searchCustomersPaginated( @RequestParam(name = "keyword", defaultValue = "")
String keyword,
                                    @RequestParam(name = "page", defaultValue = "0") int page,
                                     @RequestParam(name = "size", defaultValue = "10") int size){
        return customerService.searchCustomerPaginated( page, size, keyword);
    }

    @PreAuthorize("hasAuthority('USER')")
    @GetMapping("/customers/search")
    public List<CustomerDTO> getCustomers( @RequestParam(name = "keyword", defaultValue = "") String keyword
){
        return customerService.searchCustomer( keyword);
    }

    @PreAuthorize("hasAuthority('USER')")
    @GetMapping("/customers/{id}")
    public CustomerDTO getCustomer( @PathVariable(name="id") String customerId) throws
CustomerNotFoundException {
         return customerService.getCustomer( customerId);
    }

    @PreAuthorize("hasAuthority('ADMIN')")
    @PostMapping("/customers")
    public CustomerDTO saveCustomer( @RequestBody CustomerDTO request){
        return customerService.saveCustomer( request);
    }

    @PreAuthorize("hasAuthority('ADMIN')")
    @PutMapping("/customers/{id}")
    public CustomerDTO updateCustomer( @PathVariable(name="id") String customerId, @RequestBody CustomerDTO
request) throws CustomerNotFoundException {
```

7. Couche sécurité (Spring Security avec JWT)



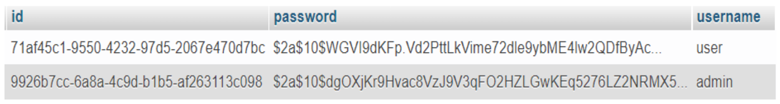
```
security
    config
        Jwt_config
        MethodSecurityConfig
        SecurityConfig
    entities
        AppRole
        AppUser
    filters
        JwtAuthenticationFilter
        JwtAuthorizationFilter
    repositories
        AppRoleRepository
        AppUserRepository
    service
        SecurityService
        SecurityServiceImpl
        UserDetailsServiceImpl
```

➢ JWT Config



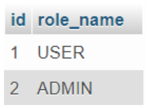
```java
public class Jwt_config {
    public static String REFRESH_PATH = "/api/refresh-token";
    public static String SECRET_PHRASE = "benkk";
    public static String AUTHORIZATION_HEADER = "Authorization";
    public static String TOKEN_HEADER_PREFIX = "Bearer ";
    public static int ACCESS_TOKEN_EXPIRATION
      = 2*24*60*60*1000; // 2mins
```

➢ Table AppUser :



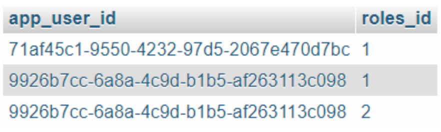
id	password	username
71af45c1-9550-4232-97d5-2067e470d7bc	$2a$10$WGVl9dKFp.Vd2PttLkVime72dle9ybME4lw2QDfByAc...	user
9926b7cc-6a8a-4c9d-b1b5-af263113c098	$2a$10$dgOXjKr9Hvac8VzJ9V3qFO2HZLGwKEq5276LZ2NRMX5...	admin

➢ Table AppRole :



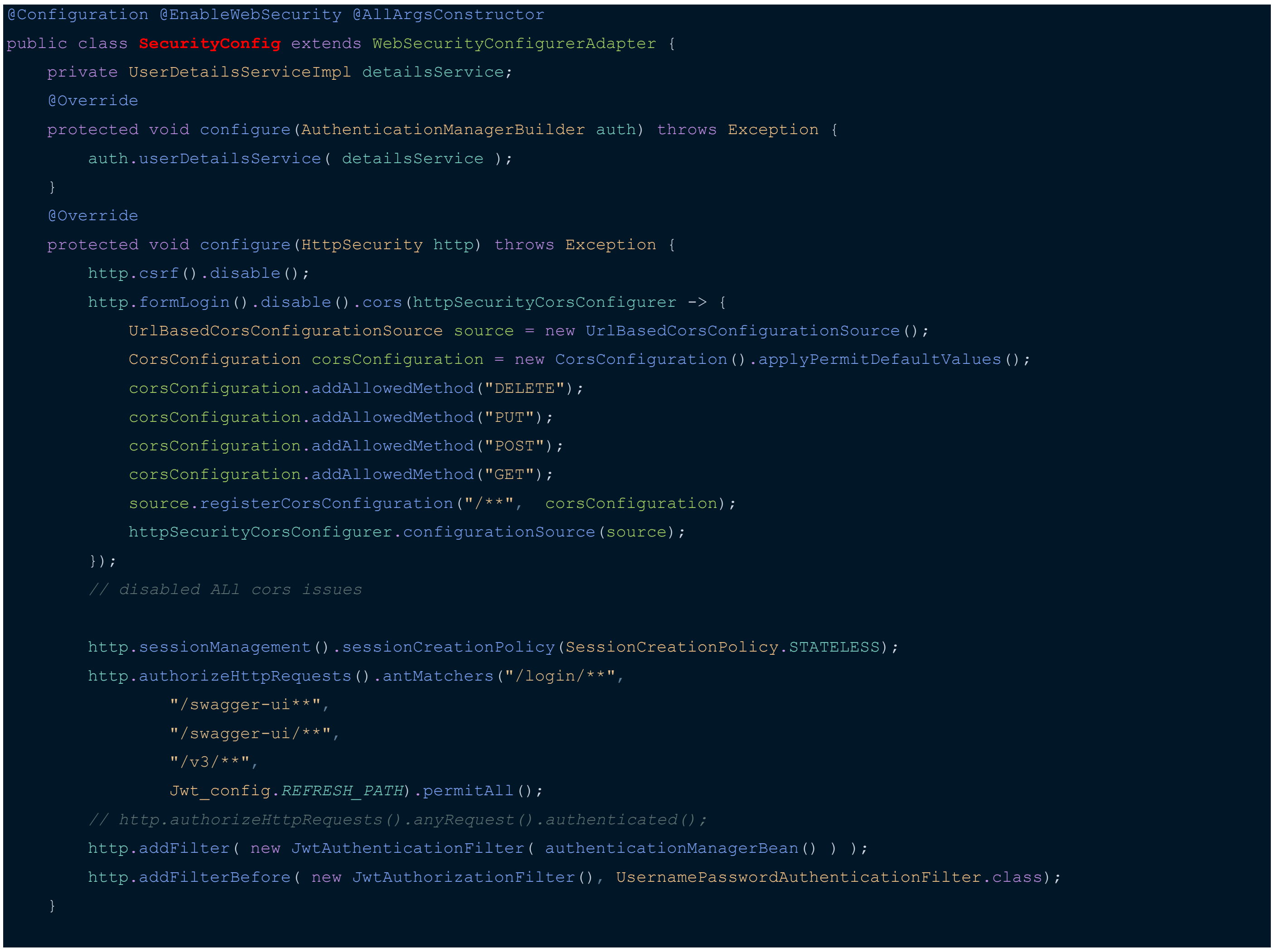
id	role_name
1	USER
2	ADMIN

➢ Table AppUserRole :



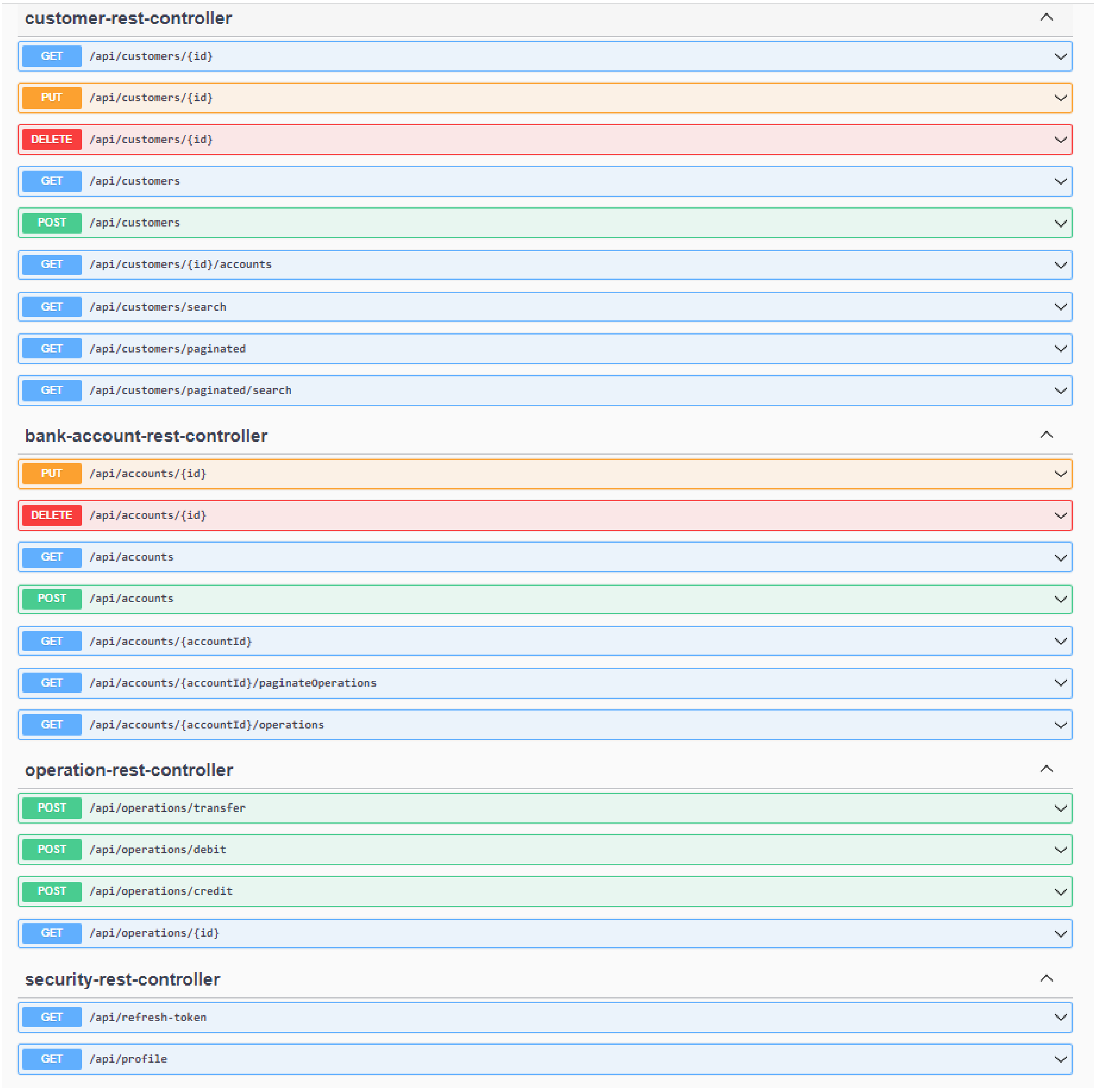
app_user_id	roles_id
71af45c1-9550-4232-97d5-2067e470d7bc	1
9926b7cc-6a8a-4c9d-b1b5-af263113c098	1
9926b7cc-6a8a-4c9d-b1b5-af263113c098	2

➢ Security Config



```java
@Configuration @EnableWebSecurity @AllArgsConstructor
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    private UserDetailsServiceImpl detailsService;
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService( detailsService );
    }
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable();
        http.formLogin().disable().cors(httpSecurityCorsConfigurer -> {
            UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
            CorsConfiguration corsConfiguration = new CorsConfiguration().applyPermitDefaultValues();
            corsConfiguration.addAllowedMethod("DELETE");
            corsConfiguration.addAllowedMethod("PUT");
            corsConfiguration.addAllowedMethod("POST");
            corsConfiguration.addAllowedMethod("GET");
            source.registerCorsConfiguration("/**",  corsConfiguration);
            httpSecurityCorsConfigurer.configurationSource(source);
        });
        // disabled ALL cors issues

        http.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);
        http.authorizeHttpRequests().antMatchers("/login/**",
                "/swagger-ui**",
                "/swagger-ui/**",
                "/v3/**",
                Jwt_config.REFRESH_PATH).permitAll();
        // http.authorizeHttpRequests().anyRequest().authenticated();
        http.addFilter( new JwtAuthenticationFilter( authenticationManagerBean() ) );
        http.addFilterBefore( new JwtAuthorizationFilter(), UsernamePasswordAuthenticationFilter.class);
    }
```

8. Routes



customer-rest-controller

```
GET     /api/customers/{id}
PUT     /api/customers/{id}
DELETE  /api/customers/{id}
GET     /api/customers
POST    /api/customers
GET     /api/customers/{id}/accounts
GET     /api/customers/search
GET     /api/customers/paginated
GET     /api/customers/paginated/search
```

bank-account-rest-controller

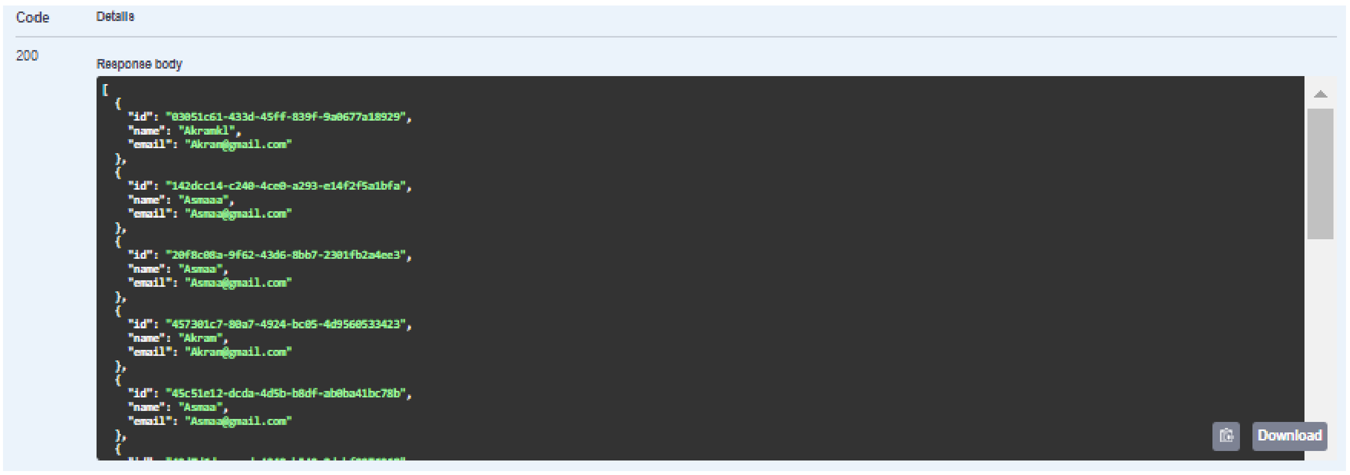
```
PUT     /api/accounts/{id}
DELETE  /api/accounts/{id}
GET     /api/accounts
POST    /api/accounts
GET     /api/accounts/{accountId}
GET     /api/accounts/{accountId}/paginateOperations
GET     /api/accounts/{accountId}/operations
```

operation-rest-controller

```
POST    /api/operations/transfer
POST    /api/operations/debit
POST    /api/operations/credit
GET     /api/operations/{id}
```

security-rest-controller

```
GET     /api/refresh-token
GET     /api/profile
```

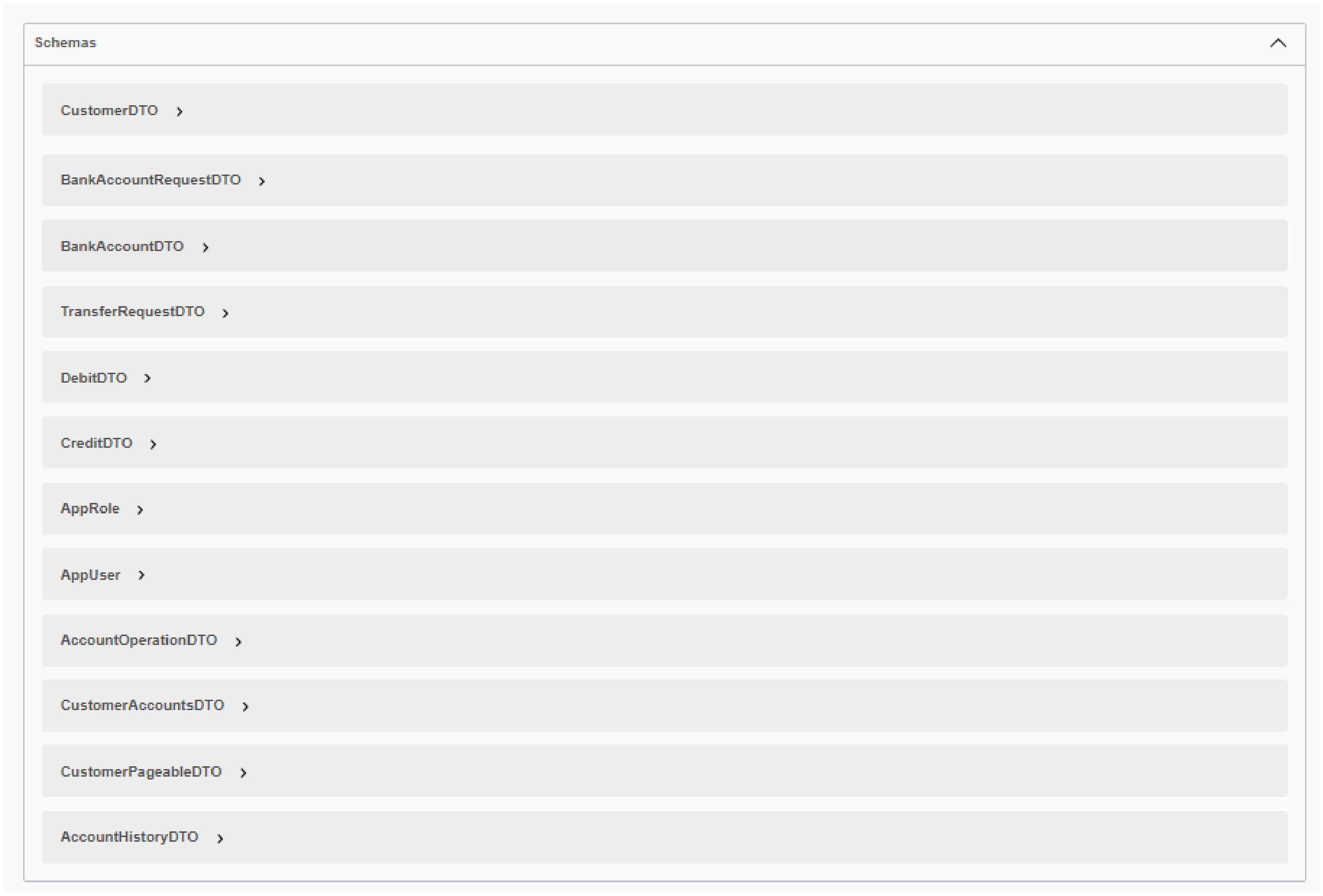


Code | Details

200 | Response body

```
[
  {
    "id": "03051c61-433d-45ff-839f-9a0677a18929",
    "name": "Akrankl",
    "email": "Akram@gmail.com"
  },
  {
    "id": "142dcc14-c240-4ce0-a293-e14f2f5a1bfa",
    "name": "Asmaaa",
    "email": "Asmaa@gmail.com"
  },
  {
    "id": "20f8c08a-9f62-43d6-8bb7-2301fb2a4ee3",
    "name": "Asmaa",
    "email": "Asmaa@gmail.com"
  },
  {
    "id": "457301c7-80a7-4924-bc05-4d9560533423",
    "name": "Akram",
    "email": "Akram@gmail.com"
  },
  {
    "id": "45c51e12-dcda-4d5b-b8df-ab0ba41bc78b",
    "name": "Asmaa",
    "email": "Asmaa@gmail.com"
  },
  {
```

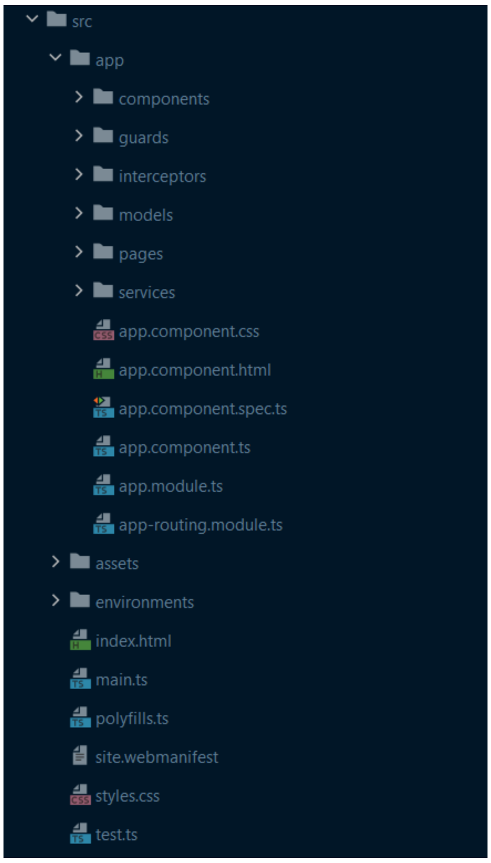
9. Schemas



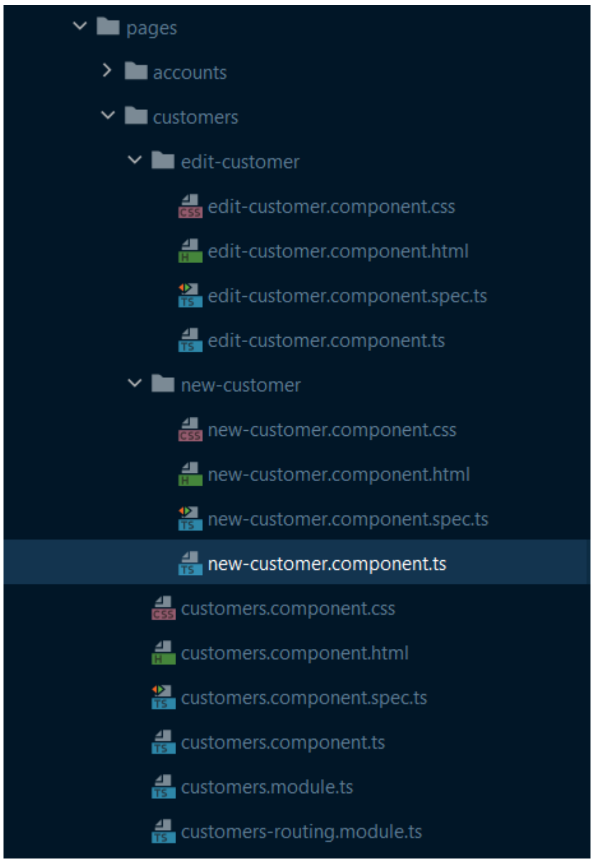
Schemas

- CustomerDTO
- BankAccountRequestDTO
- BankAccountDTO
- TransferRequestDTO
- DebitDTO
- CreditDTO
- AppRole
- AppUser
- AccountOperationDTO
- CustomerAccountsDTO
- CustomerPageableDTO
- AccountHistoryDTO

III. Frontend d'application (Angular)



```
src
  app
    components
    guards
    interceptors
    models
    pages
    services
    app.component.css
    app.component.html
    app.component.spec.ts
    app.component.ts
    app.module.ts
    app-routing.module.ts
  assets
  environments
  index.html
  main.ts
  polyfills.ts
  site.webmanifest
  styles.css
  test.ts
```



```
pages
  accounts
  customers
    edit-customer
      edit-customer.component.css
      edit-customer.component.html
      edit-customer.component.spec.ts
      edit-customer.component.ts
    new-customer
      new-customer.component.css
      new-customer.component.html
      new-customer.component.spec.ts
      new-customer.component.ts
    customers.component.css
    customers.component.html
    customers.component.spec.ts
    customers.component.ts
    customers.module.ts
    customers-routing.module.ts
```

1. Add New Customer

➢ **new-customer.component.html**

```html
<div class="card mb-3 cardLogin">
  <div class="card-header">
    <h2 class="h2 py-2 fw-light text-center">Add new customer</h2>
  </div>
  <div class="card-body">
    <form [formGroup]="form" class="col-md-8 col-lg-9 mx-auto" (ngSubmit)="handleCustomerCreate()">
      <div class="mb-3">
        <label for="name" class="form-label">Name :</label>
        <input type="text" formControlName="name" [class.is-valid]="
            form.get('name')!.valid &&
            (form.get('name')!.dirty || form.get('name')!.touched)
          " [class.is-invalid]="
            form.get('name')!.invalid &&
            (form.get('name')!.dirty || form.get('name')!.touched)
          " class="form-control" id="name" />
        <ng-container *ngIf="form.get('name')!.touched">
          <small class="text-warning d-block" *ngIf="form.get('name')!.hasError('required')">
            Cutomer name is required
          </small>
          <small class="text-warning d-block" *ngIf="form.get('name')!.hasError('minlength')">
            Cutomer name should be at least 6 characters long
          </small>
        </ng-container>
      </div>

      <div style="margin-bottom: 20px;">
        <label for="email" class="form-label">Email address :</label>
        <input type="email" formControlName="email" class="form-control" id="email" [class.is-valid]="
            form.get('email')!.valid &&
            (form.get('email')!.dirty || form.get('email')!.touched)
          " [class.is-invalid]="
            form.get('email')!.invalid &&
            (form.get('email')!.dirty || form.get('email')!.touched)
          " class="form-control" id="email" />
        <ng-container *ngIf="form.get('email')!.touched">
          <small class="text-warning d-block" *ngIf="form.get('email')!.hasError('required')">
            Cutomer email is required
          </small>
          <small class="text-warning d-block" *ngIf="form.get('email')!.hasError('email')" >
            Invalid email address
          </small>
        </ng-container>
      </div>
      <div class="card-footer" style="margin-top: 50px; background: transparent; border: none">
        <button type="submit" class="btn btn-outline-light mb-3 mt-12 d-block mx-auto" [disabled]="!form.valid||submitting">
          Create
        </button>
      </div>
    </form>

  </div>
</div>
```

➢ **new-customer.component.css**

```css
@import url(//fonts.googleapis.com/css?family=Lato:300:400);

.cardLogin{
  margin: 100px 300px 30px 300px;
  background: linear-gradient(60deg, #1b75bb 40%, #26a9e0 100%);
  font-family: 'Lato', sans-serif;
  color: white;
}

label{
  font-size: 18px;
  margin-top: 20px;
}

input{
  padding: 15px
}

.btn{
  width: 300px;
  height: 60px;
  font-size: 24px;
  font-weight: bolder;
  border-width: 2px;
}
```

➢ new-customer.component.spec.ts

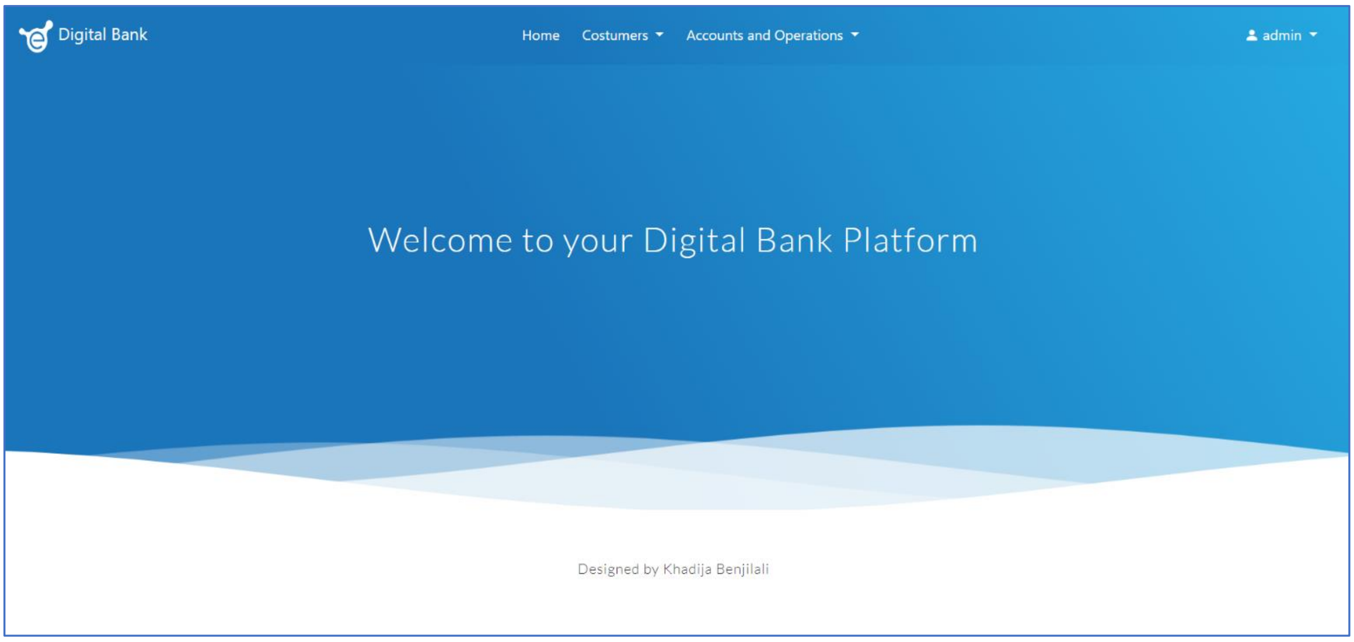
```typescript
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { NewCustomerComponent } from './new-customer.component';

describe('NewCustomerComponent', () => {
  let component: NewCustomerComponent;
  let fixture: ComponentFixture<NewCustomerComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ NewCustomerComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(NewCustomerComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

➢ new-customer.component.ts

```typescript
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { ToastrService } from 'ngx-toastr';
import { Customer } from 'src/app/models/customer.model';
import { CustomerService } from 'src/app/services/customer.service';


@Component({
  selector: 'app-new-customer',
  templateUrl: './new-customer.component.html',
  styleUrls: ['./new-customer.component.css']
})
export class NewCustomerComponent implements OnInit {

  form !: FormGroup;
  success_toast :any;
  submitting=false;

  constructor( private fb: FormBuilder,
    private customerService: CustomerService,
    private toastr: ToastrService) {

    this.form = this.fb.group({
      name: this.fb.control("", [
        Validators.required, Validators.minLength(6)
      ]),
      email: this.fb.control("", [
        Validators.required, Validators.email
      ])})
  }

  ngOnInit(): void {

  }

  handleCustomerCreate(){
    this.submitting=true;
    let customer:Customer = {
      id:"",
      name: this.form.value.name,
      email: this.form.value.email
    }
    this.customerService.saveCustomer( customer).subscribe( {
      next: data=>{
        this.toastr.success( '', 'Customer added successfully!', { closeButton: true, positionClass: "toast-top-center" });
        this.form.reset();
        this.submitting=false;
      },
      error: err=>{
        this.toastr.error( '', 'Customer not saved, an error happened !', { closeButton: true, positionClass: "toast-top-
center", });
        this.submitting=false;
      }
    });
  }

}
```

2. Les interfaces de l'application

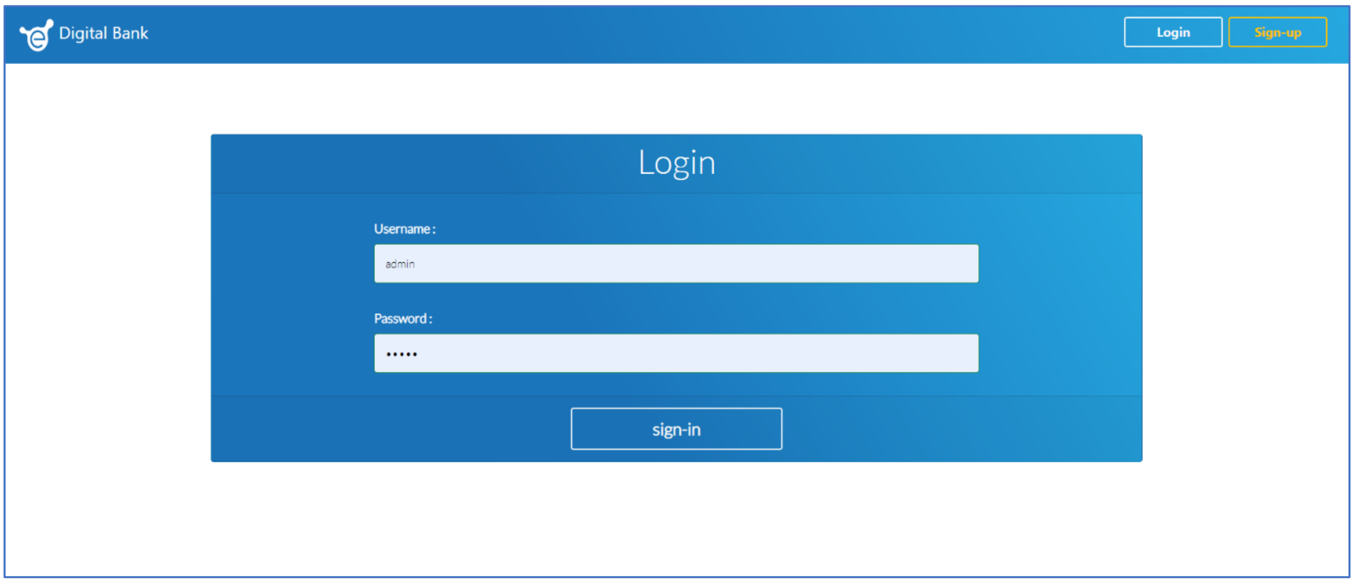
➢ Home



Welcome to your Digital Bank Platform

Designed by Khadija Benjilali

➢ Login

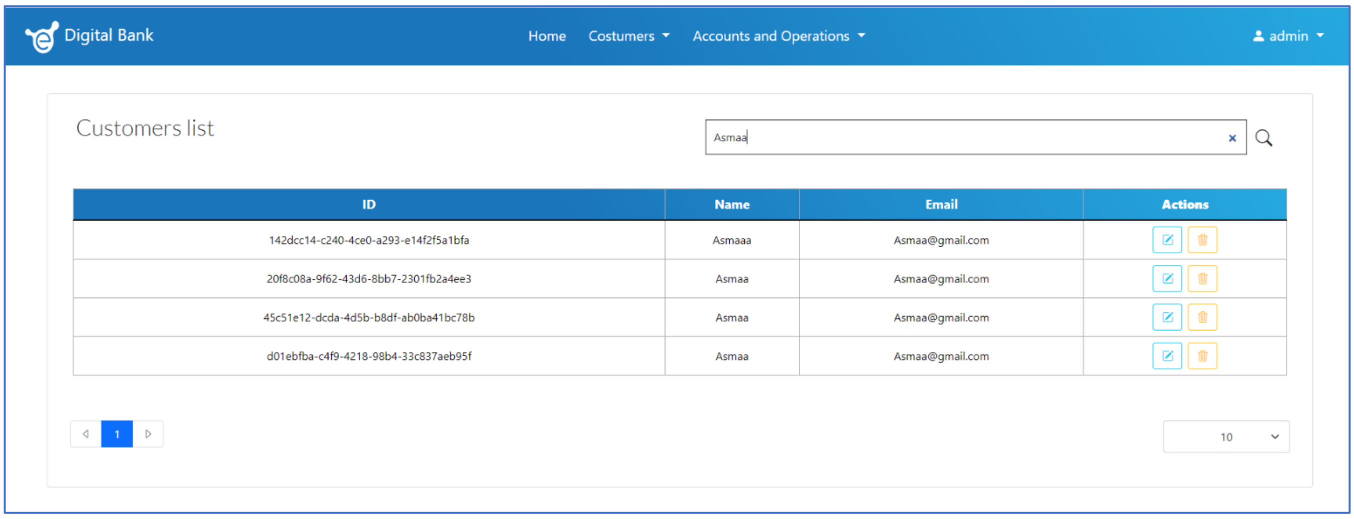


Login

Username : admin

Password : •••••

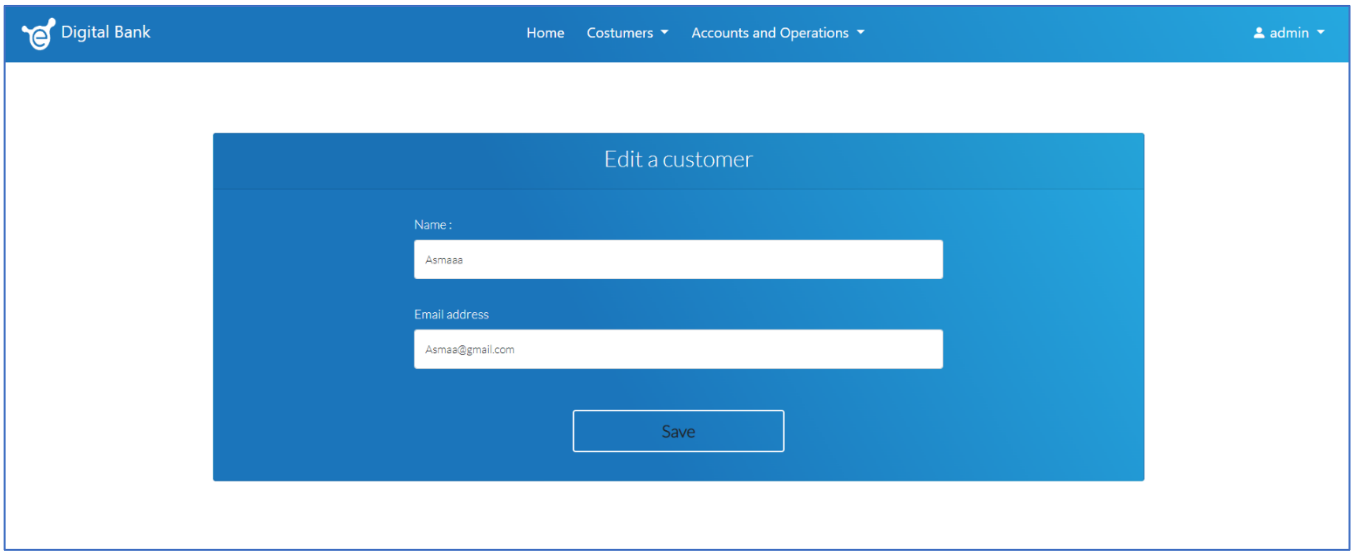
➢ Customer Interfaces



Customers list

Asmaa

ID	Name	Email	Actions
142dcc14-c240-4ce0-a293-e14f2f5a1bfa	Asmaaa	Asmaa@gmail.com	
20f8c08a-9f62-43d6-8bb7-2301fb2a4ee3	Asmaa	Asmaa@gmail.com	
45c51e12-dcda-4d5b-b8df-ab0ba41bc78b	Asmaa	Asmaa@gmail.com	
d01ebfba-c4f9-4218-98b4-33c837aeb95f	Asmaa	Asmaa@gmail.com	

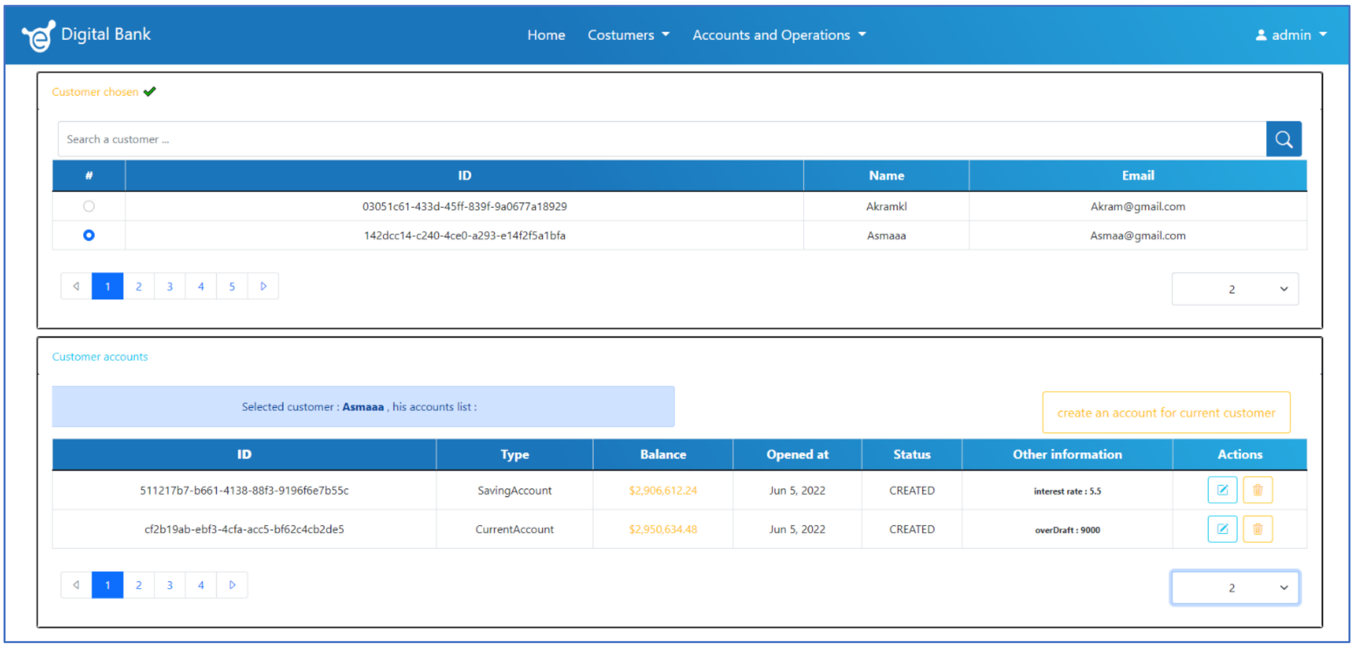


Edit a customer

Name : Asmaaa

Email address: Asmaa@gmail.com

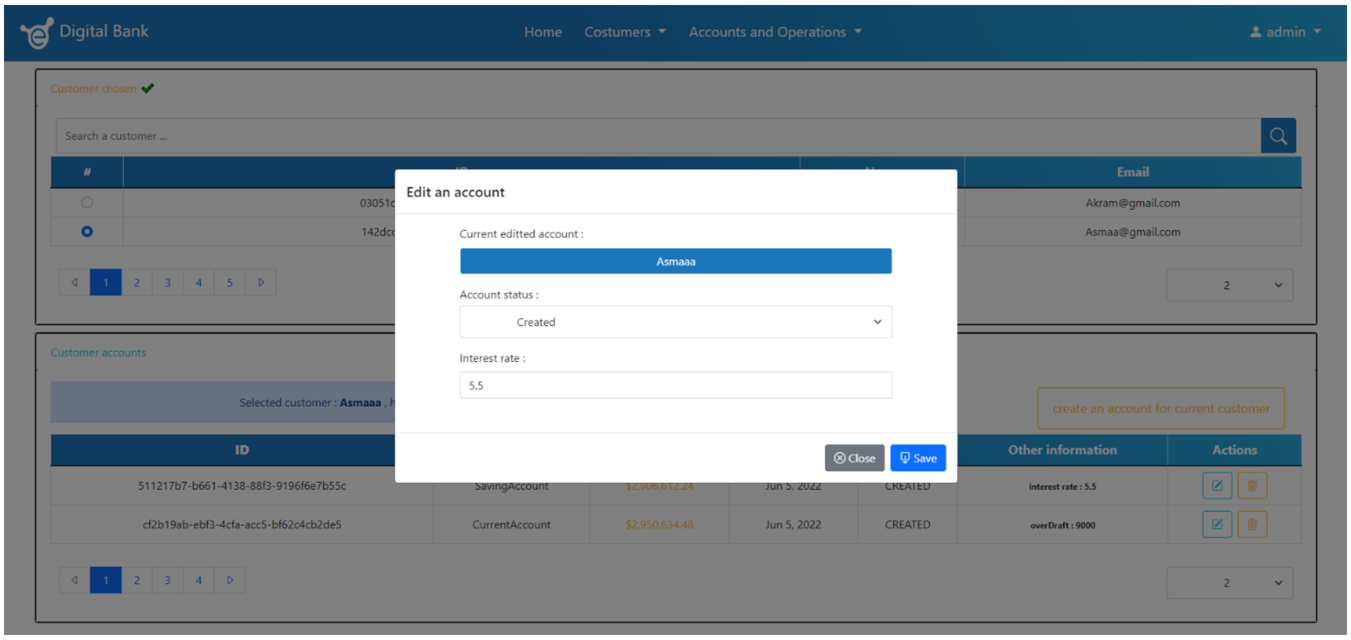
➢ Bank Account Interfaces



#	ID	Name	Email
○	03051c61-433d-45ff-839f-9a0677a18929	Akramkl	Akram@gmail.com
●	142dcc14-c240-4ce0-a293-e14f2f5a1bfa	Asmaaa	Asmaa@gmail.com

Selected customer : **Asmaaa** , his accounts list :

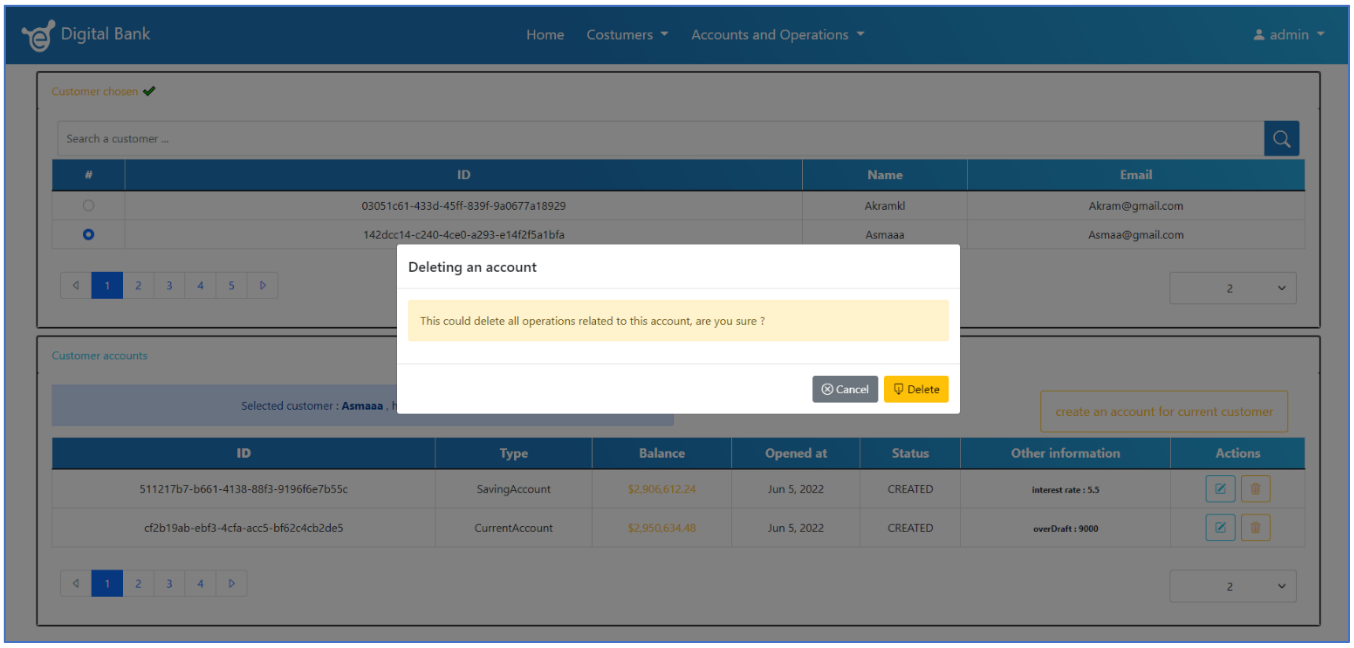
ID	Type	Balance	Opened at	Status	Other information	Actions
511217b7-b661-4138-88f3-9196f6e7b55c	SavingAccount	$2,906,612.24	Jun 5, 2022	CREATED	**interest rate : 5.5**	
cf2b19ab-ebf3-4cfa-acc5-bf62c4cb2de5	CurrentAccount	$2,950,634.48	Jun 5, 2022	CREATED	**overDraft : 9000**	


Edit an account

Current editted account : Asmaaa

Account status : Created

Interest rate : 5.5



Customer chosen ✓

Search a customer ...

#	ID	Name	Email
○	03051c61-433d-45ff-839f-9a0677a18929	Akramkl	Akram@gmail.com
●	142dcc14-c240-4ce0-a293-e14f2f5a1bfa	Asmaaa	Asmaa@gmail.com

1 2 3 4 5 — 2

Deleting an account

This could delete all operations related to this account, are you sure ?

Cancel Delete

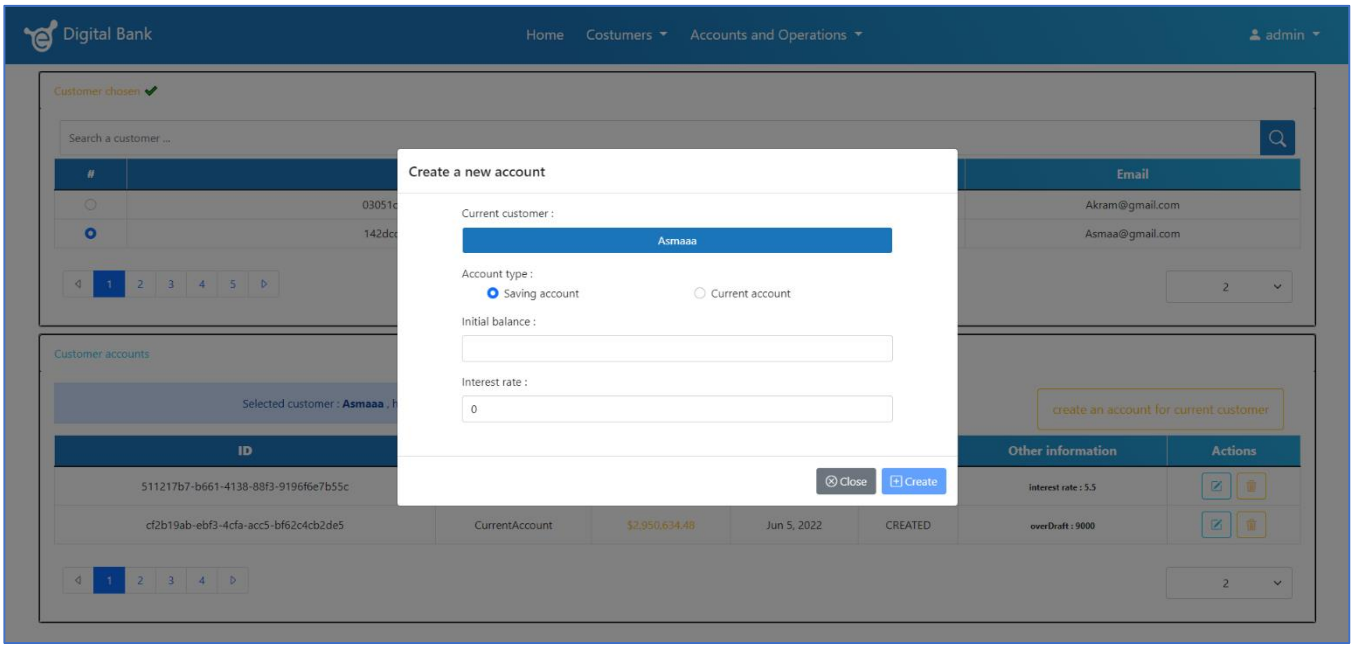
Customer accounts

Selected customer : **Asmaaa** ,

create an account for current customer

ID	Type	Balance	Opened at	Status	Other information	Actions
511217b7-b661-4138-88f3-9196f6e7b55c	SavingAccount	$2,906,612.24	Jun 5, 2022	CREATED	interest rate : 5.5	
cf2b19ab-ebf3-4cfa-acc5-bf62c4cb2de5	CurrentAccount	$2,950,634.48	Jun 5, 2022	CREATED	overDraft : 9000	

1 2 3 4 — 2



Customer chosen ✓

Search a customer ...

#	ID	Email
○	03051c	Akram@gmail.com
●	142dc	Asmaa@gmail.com

1 2 3 4 5 — 2

Create a new account

Current customer : Asmaaa

Account type : [✓ Saving account] Current account

Initial balance :

Interest rate : 0

Close Create

Customer accounts

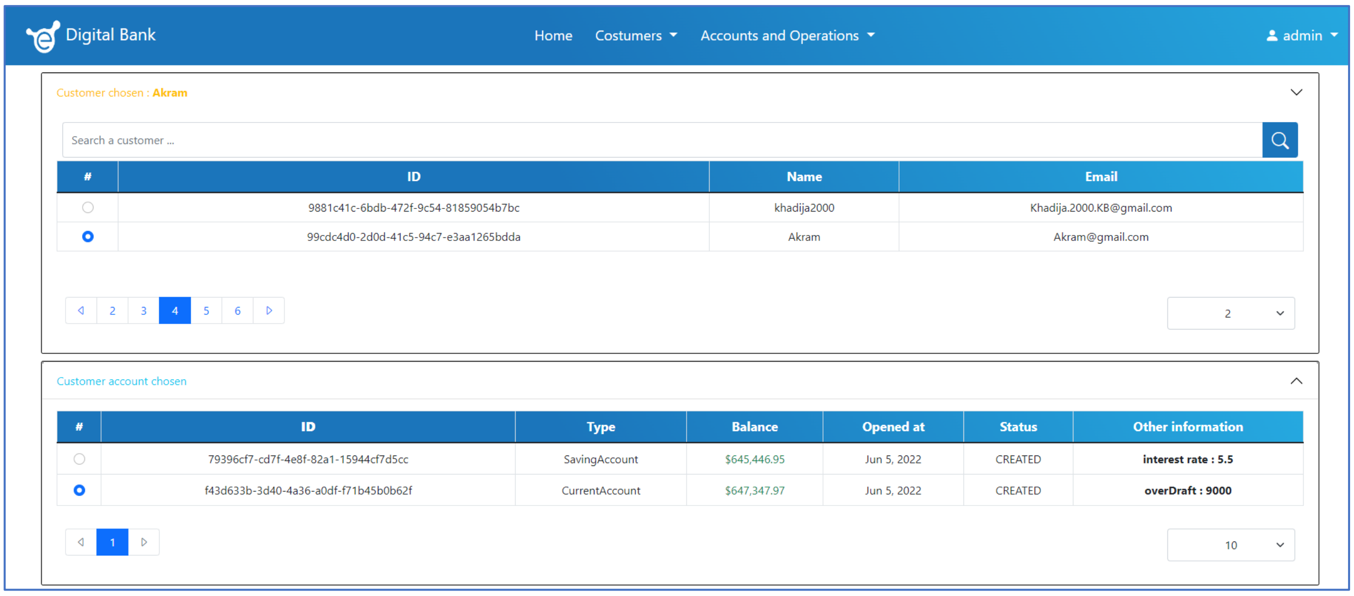
Selected customer : **Asmaaa** ,

create an account for current customer

ID	Type	Balance	Opened at	Status	Other information	Actions
511217b7-b661-4138-88f3-9196f6e7b55c					interest rate : 5.5	
cf2b19ab-ebf3-4cfa-acc5-bf62c4cb2de5	CurrentAccount	$2,950,634.48	Jun 5, 2022	CREATED	overDraft : 9000	

1 2 3 4 — 2

➢ Account Operations Interfaces



Customer chosen : **Akram**

Search a customer ...

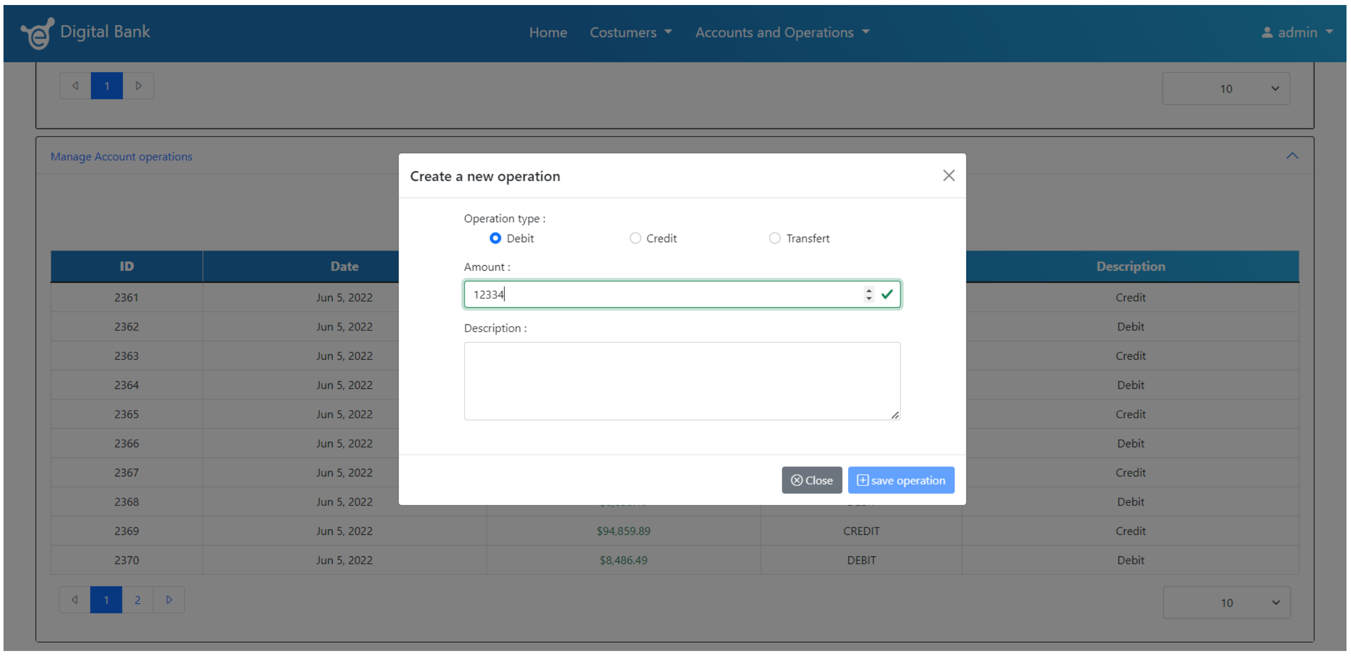
#	ID	Name	Email
○	9881c41c-6bdb-472f-9c54-81859054b7bc	khadija2000	Khadija.2000.KB@gmail.com
●	99cdc4d0-2d0d-41c5-94c7-e3aa1265bdda	Akram	Akram@gmail.com

2 3 4 5 6 — 2

Customer account chosen

#	ID	Type	Balance	Opened at	Status	Other information
○	79396cf7-cd7f-4e8f-82a1-15944cf7d5cc	SavingAccount	$645,446.95	Jun 5, 2022	CREATED	**interest rate : 5.5**
●	f43d633b-3d40-4a36-a0df-f71b45b0b62f	CurrentAccount	$647,347.97	Jun 5, 2022	CREATED	**overDraft : 9000**

1 — 10



Digital Bank

Home Costumers Accounts and Operations admin

Manage Account operations

Create a new operation

Operation type :
[✓ Debit] Credit Transfert

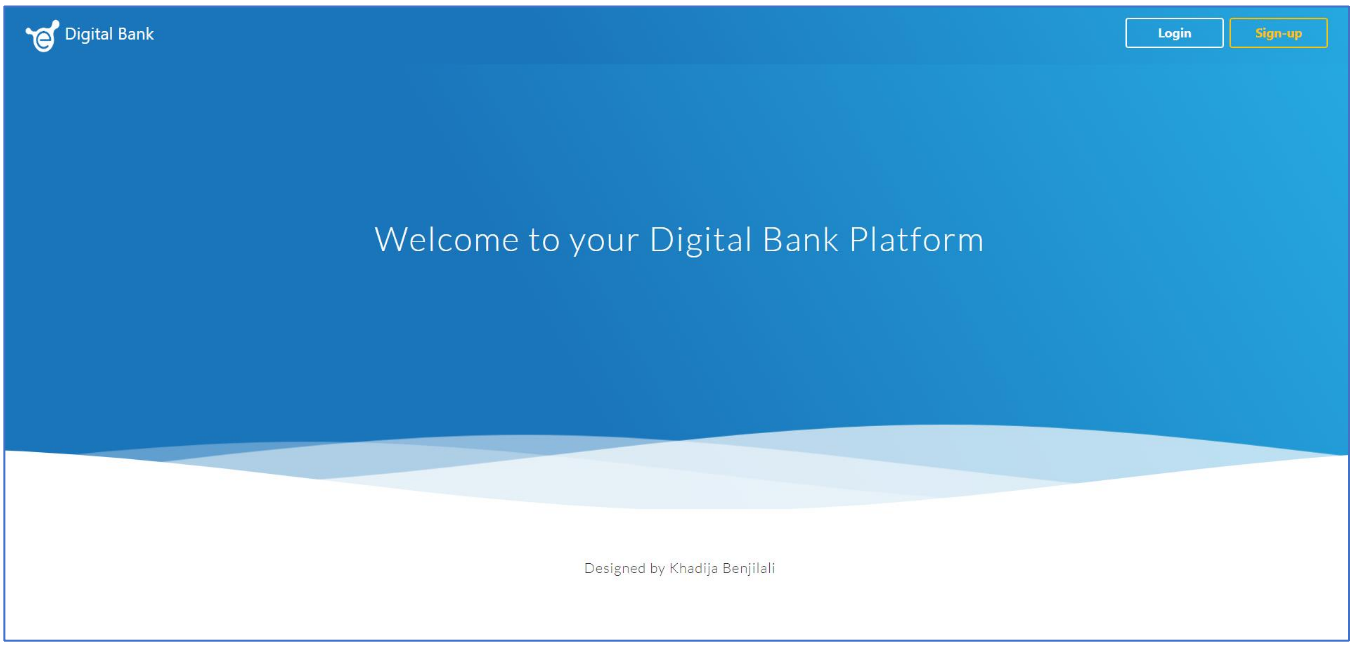
Amount :
12334

Description :

Close save operation

ID	Date			Description
2361	Jun 5, 2022			Credit
2362	Jun 5, 2022			Debit
2363	Jun 5, 2022			Credit
2364	Jun 5, 2022			Debit
2365	Jun 5, 2022			Credit
2366	Jun 5, 2022			Debit
2367	Jun 5, 2022			Credit
2368	Jun 5, 2022			Debit
2369	Jun 5, 2022	$94,859.89	CREDIT	Credit
2370	Jun 5, 2022	$8,486.49	DEBIT	Debit

➢ Log out



Welcome to your Digital Bank Platform

Designed by Khadija Benjilali