



جامعة الحسن الثاني بالدار البيضاء  
+αααααααααα ααααα αααα ααα α αααααααααα  
UNIVERSITÉ HASSAN II DE CASABLANCA



# Master SID-1

## Gestion des Comptes Bancaires Spring Boot « JEE »

Réalisé Par :

Abdelhakim HADI-ALAOUI

Encadré Par :

Mr. Mohammed YOUSSEFI

Année Universitaire : 2016/2017

## Table des matières

Introduction.....	2
Enoncé .....	2
Architecture technique.....	3
Diagramme de classes des entités .....	4
Structure de projet .....	4
Code Source .....	5
1)    Couche dao.....	5
2)    Couche metier.....	10
3)    Couche web .....	12
Démonstration de l'application .....	20

## Introduction

On souhaite créer une application qui permet de gérer des comptes bancaires.

- Chaque compte est défini un code, un solde et une date de création
- Un compte courant est un compte qui possède en plus un découvert
- Un compte épargne est un compte qui possède en plus un taux d'intérêt.
- Chaque compte appartient à un client.
- Chaque client est défini par son code et son nom
- Chaque compte peut subir plusieurs opérations.
- Il existe deux types d'opérations : Versement et Retrait
- Une opération est définie par un numéro, une date et un montant.

## Enoncé

### Exigences fonctionnelles

L'application doit permettre de :

- Gérer /les clients :
  - Ajouter un client
  - Consulter tous les clients
  - Consulter les clients dont le nom contient un mot clé.
- Gérer les comptes :
  - Ajouter un compte
  - Consulter un compte
- Gérer les opérations :
  - Effectuer un versement d'un montant dans un compte
  - Effectuer un retrait d'un montant dans un compte
  - Consulter les opérations d'un compte page par page
  - Les opérations nécessitent une opération

### Exigences Techniques

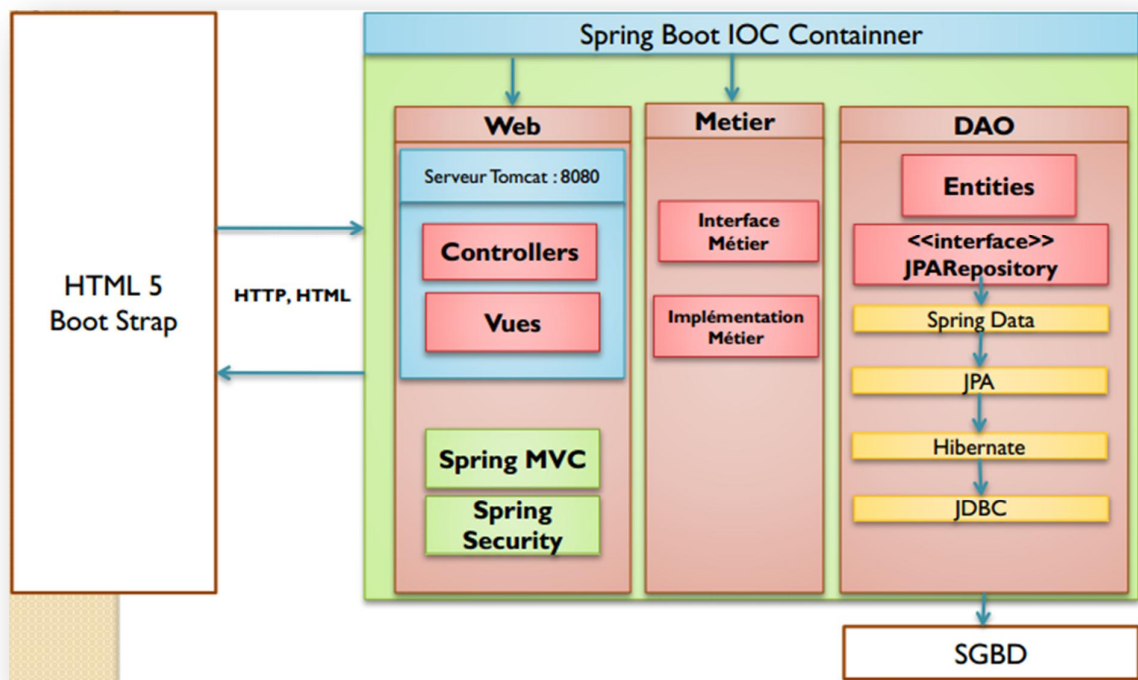
- Les données sont stockées dans une base de données MySQL
- L'application se compose de trois couches :
  - La couche DAO qui est basée sur Spring Data, JPA, Hibernate et JDBC.

- La couche Métier
- La couche Web basée sur MVC coté Serveur en utilisant Thymeleaf.
- La sécurité est basée sur Spring Security d'authentification

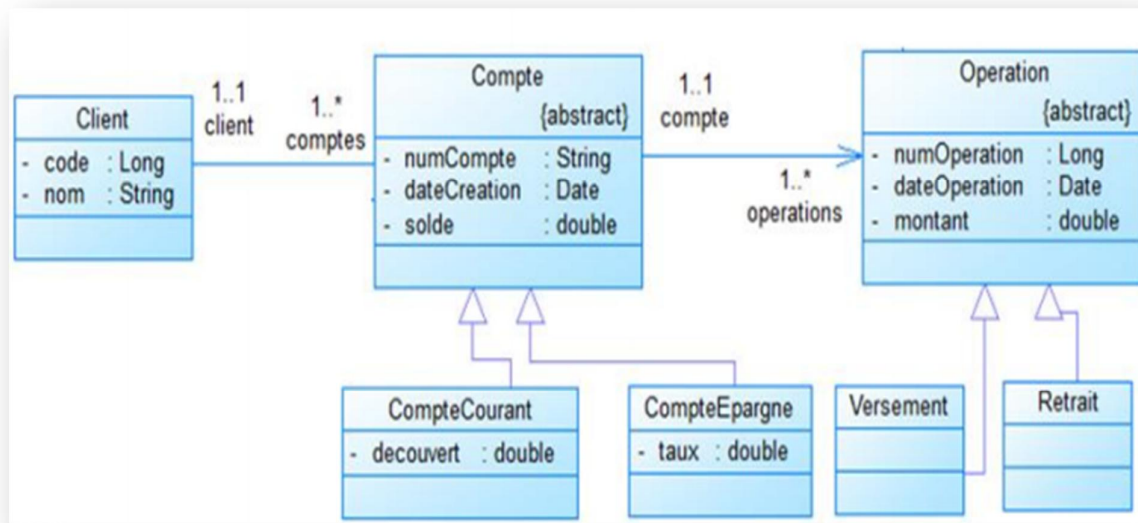
#### Travail demandé :

- Etablir une architecture technique du projet
- Etablir un diagramme de classes qui montre les entités, la couche DAO et la couche métier.
- Créer un projet SpringBoot qui contient les éléments suivants :
  - Les entités
  - La couche DAO (Interfaces Spring data)
  - La couche métier (Interfaces et implémentations)
  - La couche web :
    - Les contrôleurs Spring MVC
    - Les Vue basée sur Thymeleaf
- Sécuriser l'application en utilisant un système d'authentification basé sur Spring Security

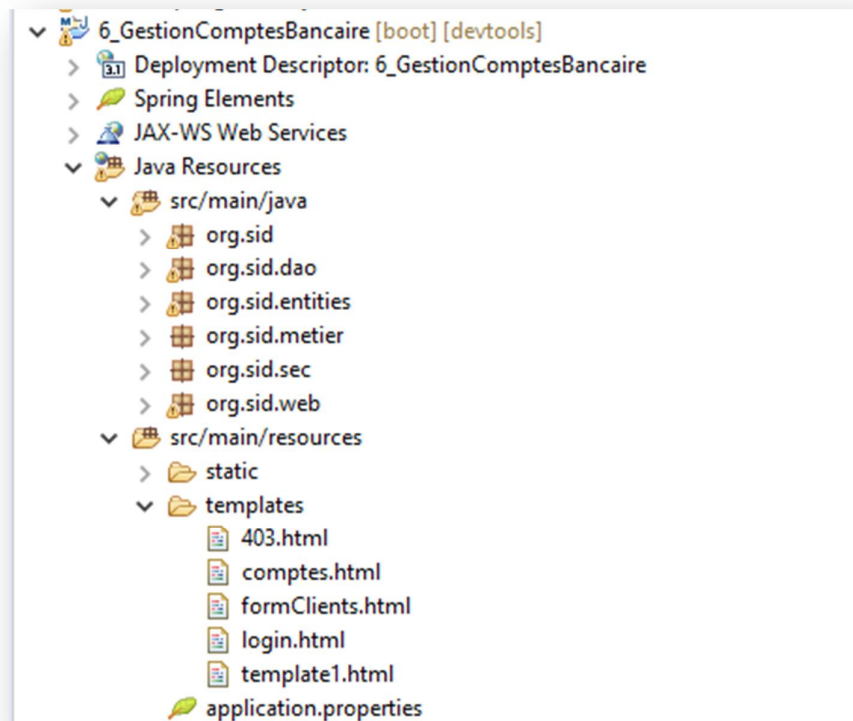
#### Architecture technique



## Diagramme de classes des entités



## Structure de projet



## Code Source

### 1) Couche dao

#### *Package com.sid.dao*

##### - ClientRepository.java

```
package org.sid.dao;

import org.sid.entities.Client;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ClientRepository extends JpaRepository<Client, Long> {

}
```

##### - CompteRepository.java

```
package org.sid.dao;

import org.sid.entities.Compte;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

public interface CompteRepository extends JpaRepository<Compte, String> {
    @Query("select c from Compte c where c.client.code=:x")
    public Page<Compte> listComptes(@Param("x")Long codeCte,Pageable page);
}
```

##### - OperationRepository.java

```
package org.sid.dao;

import org.sid.entities.Operation;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

public interface OperationRepository extends JpaRepository<Operation, Long> {
    @Query("select o from Operation o where o.compte.code like :x order by o.dateOp desc")
    public Page<Operation> listOperation(@Param("x")String codeCte,Pageable page);
}
```

#### *Package com.sid.entities*

##### - Client.java

```
package org.sid.entities;

import java.io.Serializable;
import java.util.Collection;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToMany;
```

```

@Entity
public class Client implements Serializable {
    @Id @GeneratedValue
    private Long code ;
    private String nom;
    @OneToMany(mappedBy="client", fetch=FetchType.LAZY)
    private Collection<Compte> comptes;

    public Client(String nom) {
        super();
        this.nom = nom;
    }
    public Client(String nom, Collection<Compte> comptes) {
        super();
        this.nom = nom;
        this.comptes = comptes;
    }
    public Client() {
        super();
        // TODO Auto-generated constructor stub
    }

    // getters/setters
}

```

- Compte.java

```

package org.sid.entities;

import java.io.Serializable;
import java.util.Date;
import java.util.List;
import javax.persistence.DiscriminatorColumn;
import javax.persistence.DiscriminatorType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;

@Entity
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="TYPE_CPTE", discriminatorType=DiscriminatorType.STRING, length=2)
public abstract class Compte implements Serializable {
    @Id
    private String code ;
    private double solde;
    private Date dateCreation;
    @ManyToOne
    @JoinColumn(name="CODE_CLIENT")
    private Client client ;
    @OneToMany(mappedBy="compte", fetch=FetchType.LAZY)
    private List<Operation> operations;
}

```

```

    public Compte() {}

    public Compte(String code, double solde, Date dateCreation, Client client) {
        this.code = code;
        this.solde = solde;
        this.dateCreation = dateCreation;
        this.client = client;
    }
    public Compte(double solde, Date dateCreation, Client client) {
        this.solde = solde;
        this.dateCreation = dateCreation;
        this.client = client;
    }
    // getters/setters
}

```

#### - CompteCourant.java

```

package org.sid.entities;
import java.util.Date;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@DiscriminatorValue("CC")
public class CompteCourant extends Compte {
    private double decouvert;

    public CompteCourant() {
        super();
    }

    public CompteCourant(String code, double solde, Date dateCreation, Client
        client, double decouvert) {
        super(code, solde, dateCreation, client);
        this.decouvert = decouvert;
    }
    public CompteCourant(double solde, Date dateCreation, Client client,
        double decouvert) {
        super(solde, dateCreation, client);
        this.decouvert = decouvert;
    }

    public double getDecouvert() {
        return decouvert;
    }

    public void setDecouvert(double decouvert) {
        this.decouvert = decouvert;
    }
}

```

#### - CompteEpargne.java

```

package org.sid.entities;
import java.util.Date;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@DiscriminatorValue("CE")
public class CompteEpargne extends Compte {

```



```

    private double taux;

    public CompteEpargne() {
        super();
    }

    public CompteEpargne(String code, double solde, Date dateCreation, Client
    client, double taux) {
        super(code, solde, dateCreation, client);
        this.taux = taux;
    }

    public CompteEpargne(double solde, Date dateCreation, Client client,
    double taux) {
        super(solde, dateCreation, client);
        this.taux = taux;
    }

    public double getTaux() {
        return taux;
    }

    public void setTaux(double taux) {
        this.taux = taux;
    }
}

```

- **Operation.java**

```

package org.sid.entities;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.DiscriminatorColumn;
import javax.persistence.DiscriminatorType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

@Entity
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="TYPE_OP",discriminatorType=DiscriminatorType.STRING,length=1)
public abstract class Operation implements Serializable {

    @Id @GeneratedValue
    private Long numero ;
    private Date dateOp;
    private double montant;
    @ManyToOne
    @JoinColumn(name="CODE_CPT")
    private Compte compte;
    public Operation() {}
    public Operation(Long numero, Date dateOp, double montant, Compte compte) {
        super();
        this.numero = numero;
        this.dateOp = dateOp;
    }
}

```

```

        this.montant = montant;
        this.compte = compte;
    }
    public Operation(Date dateOp, double montant, Compte compte) {
        super();
        this.dateOp = dateOp;
        this.montant = montant;
        this.compte = compte;
    }

    //getters and setters
}

```

#### - Retrait.java

```

package org.sid.entities;
import java.util.Date;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@DiscriminatorValue("R")
public class Retrait extends Operation {

    public Retrait() {
        super();
    }

    public Retrait(Date dateOp, double montant, Compte compte) {
        super(dateOp, montant, compte);
    }

    public Retrait(Long numero, Date dateOp, double montant, Compte compte) {
        super(numero, dateOp, montant, compte);
    }
}

```

#### - Versement.java

```

package org.sid.entities;
import java.util.Date;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@DiscriminatorValue("V")
public class Versement extends Operation {

    public Versement() {
        super();
    }

    public Versement(Date dateOp, double montant, Compte compte) {
        super(dateOp, montant, compte);
    }

    public Versement(Long numero, Date dateOp, double montant, Compte compte) {
        super(numero, dateOp, montant, compte);
    }
}

```

## 2) Couche metier

*Package com.sid.metier*

### - IBanqueMetier

```
package org.sid.metier;

import org.sid.entities.Compte;
import org.sid.entities.Operation;
import org.springframework.data.domain.Page;

public interface IBanqueMetier {

    public Compte consulterCompte(String codeCte);
    public void verser(String codeCte,double montant);
    public void retirer(String codeCte,double montant);
    public void virement(String codeCte1,String codeCte2,double montant);
    public Page<Operation> listOperation(String codeCte,int page,int size);
}
```

### - IClientMetier

```
package org.sid.metier;

import org.sid.entities.Client;
import org.sid.entities.Compte;
import org.springframework.data.domain.Page;

public interface IClientMetier {

    public Client consulterClient(Long codeClt);
    public void supprimerClient(Long codeClt);
    public Client ajouter(Client c);
    //public Client modifier(Client c);
    public Page<Compte> listComptes(Long codeClt,int page,int size);
}
```

### - BanqueMetierImpl

```
package org.sid.metier;

import java.util.Date;

import org.sid.dao.CompteRepository;
import org.sid.dao.OperationRepository;
import org.sid.entities.Compte;
import org.sid.entities.CompteCourant;
import org.sid.entities.Operation;
import org.sid.entities.Retrait;
import org.sid.entities.Versement;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
@Transactional
public class BanqueMetierImpl implements IBanqueMetier {
```

```

@Autowired
CompteRepository compteRepository;
@Autowired
OperationRepository operationRepository;
@Override
public Compte consulterCompte(String codeCte) {
    Compte compte= compteRepository.findOne(codeCte);
    if(compte==null) throw new RuntimeException("Compte introuvable");

    return compte;
}

@Override
public void verser(String codeCte, double montant) {
    Compte compte=consulterCompte(codeCte);
    Versement opv=new Versement(new Date(), montant, compte);
    operationRepository.save(opv);
    compte.setSolde(compte.getSolde()+montant);
    compteRepository.save(compte);
}

@Override
public void retirer(String codeCte, double montant) {
    Compte compte=consulterCompte(codeCte);
    Retrait opr=new Retrait(new Date(), montant, compte);
    double faciliteCaisse=0;
    if(compte instanceof CompteCourant)
        faciliteCaisse=((CompteCourant) compte).getDecouvert();
    if(compte.getSolde()+faciliteCaisse<montant)
        throw new RuntimeException("Solde Insuffisant !");
    operationRepository.save(opr);
    compte.setSolde(compte.getSolde()-montant);
    compteRepository.save(compte);
}

@Override
public void virement(String codeCteR, String codeCteV, double montant) {
    //Compte compte1=consulterCompte(codeCte1);
    //Compte compte2=consulterCompte(codeCte2);
    if(codeCteR.equals(codeCteV))
        throw new RuntimeException("impossible de virer sur le meme
compte !");
    retirer(codeCteR,montant);
    verser(codeCteV,montant);
}

@Override
public Page<Operation> listOperation(String codeCte, int page, int size) {

    return operationRepository.listOperation(codeCte, new
PageRequest(page, size));
}

}

- ClientMetierImpl
package org.sid.metier;

import javax.transaction.Transactional;

```

```

import org.sid.dao.ClientRepository;
import org.sid.dao.CompteRepository;
import org.sid.entities.Client;
import org.sid.entities.Compte;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.stereotype.Service;

@Service
@Transactional
public class ClientMetierImpl implements IClientMetier {

    @Autowired
    ClientRepository clientRepository;
    @Autowired
    CompteRepository compteRepository;

    @Override
    public Client consulterClient(Long codeClt) {

        return clientRepository.findOne(codeClt);
    }

    @Override
    public void supprimerClient(Long codeClt) {
        clientRepository.delete(codeClt);
    }

    @Override
    public Client ajouter(Client c) {
        return clientRepository.save(c);
    }

    @Override
    public Page<Compte> listComptes(Long codeClt, int page, int size) {
        return compteRepository.listComptes(codeClt, new PageRequest(page,
size));
    }
}

```

### 3) Couche web

*Package com.sid.web*

- **banqueController**

```

package org.sid.web;

import org.sid.entities.Compte;
import org.sid.entities.Operation;
import org.sid.metier.IBanqueMetier;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.stereotype.Controller;

```

```

import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class banqueController {

    @Autowired
    private IBanqueMetier banqueMetier;

    @RequestMapping("/operations")
    public String index(){
        return "comptes";
    }

    @RequestMapping("/consultercompte")
    public String consulter(Model model,String codeCte,
                           @RequestParam(defaultValue="0") int page,
                           @RequestParam(defaultValue="4")int size){

        try {
            Compte c=banqueMetier.consulterCompte(codeCte);
            Page<Operation>
operations=banqueMetier.listOperation(codeCte,page, size);
            model.addAttribute("codeCte",codeCte);
            model.addAttribute("compte",c);
            model.addAttribute("pageCourant",page);
            model.addAttribute("operations",operations.getContent());
            // System.out.println(operations.getContent());
        } catch (Exception e) {
            model.addAttribute("exception",e);
        }

        return "comptes";
    }

    @RequestMapping(value="/saveOperation",method=RequestMethod.POST)
    public String saveOperation(Model model,String codeCte,String
codeCteDes,String typeOperation,double montant){
        try {
            if(typeOperation.equals("Retrait"))
                banqueMetier.retirer(codeCte, montant);
            else if(typeOperation.equals("Versement"))
                banqueMetier.verser(codeCte, montant);
            else if (typeOperation.equals("Virement"))
                banqueMetier.virement(codeCte, codeCteDes, montant);

        } catch (Exception e) {
            model.addAttribute("error" ,e);
            return
"redirect:/consultercompte?codeCte="+codeCte+"&error="+e.getMessage();
        }

        return "redirect:/consultercompte?codeCte="+codeCte;
    }
}

- ClientController
package org.sid.web;

```

```

import org.sid.entities.Client;
import org.sid.entities.Compte;
import org.sid.metier.IClientMetier;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class ClientController {

    @Autowired
    IClientMetier clientMetier;

    @RequestMapping(value="/clients")
    public String client(){
        return "formClients";
    }

    @RequestMapping("/consulterclient")
    public String consulter(Model model,Long codeClt,
                           @RequestParam(defaultValue="0") int page,
                           @RequestParam(defaultValue="4")int size){

        try {
            Client c=clientMetier.consulterClient(codeClt);
            Page<Compte> comptes=clientMetier.listComptes(codeClt,page,
size);

            model.addAttribute("codeClt",codeClt);
            model.addAttribute("client",c);
            model.addAttribute("pageCourant",page);
            model.addAttribute("comptes",comptes.getContent());
            // System.out.println(operations.getContent());
        } catch (Exception e) {
            model.addAttribute("exception",e);
        }

        return "formClients";
    }
}

- WebInitializer
package org.sid.web;

import org.sid.Application;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.support.SpringBootServletInitializer;

public class WebInitializer extends SpringBootServletInitializer{

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder
builder) {
        return builder.sources(Application.class)
            ;
    }
}

```

## *Package com.sid.sec*

### - SecurityConfig

```
package org.sid.sec;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.encoding.Md5PasswordEncoder;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    private DataSource dataSource;
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        /* auth.inMemoryAuthentication().withUser("admin")
           .password("0000").roles("ADMIN","USER");
        auth.inMemoryAuthentication().withUser("user")
           .password("0000")
        .roles("USER");
        */
        auth.jdbcAuthentication().dataSource(dataSource)
            .usersByUsernameQuery("select username as principal,password as credentials
,active from users where username=?")
            .authoritiesByUsernameQuery("select username as principal,roles as role from
users_roles where username=?")
            .rolePrefix("ROLE_")
            .passwordEncoder(new Md5PasswordEncoder());
    }
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.formLogin().loginPage("/login");

        http.authorizeRequests().antMatchers("/operations","/consultercompte").hasRole("USER");
        http.authorizeRequests().antMatchers("/saveOperation").hasRole("ADMIN");
        http.exceptionHandling().accessDeniedPage("/403");
    }
}
```

### - SecurityController

```
package org.sid.sec;
```



```

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class SecurityController {

    @RequestMapping(value="/login")
    public String login(){

        return "login";
    }

    @RequestMapping(value="/")
    public String home(){
        return "redirect:/operations";
    }

    @RequestMapping(value="/403")
    public String accessDenied(){
        return "403";
    }
}

```

## Fichiers html

### - Template1.html

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head>
<meta charset="ISO-8859-1" />
<link rel="stylesheet" type="text/css" href="../static/css/bootstrap.css"
th:href="@{/css/bootstrap.css}" />
<link rel="stylesheet" type="text/css" href="../static/css/myStyle.css"
th:href="@{/css/myStyle.css}" />
<title>Gestion de Banque</title>
</head>
<body>
<header>
    <div class="navbar navbar-inverse">
        <div class="container-fluid">
            <ul class="nav navbar-nav">
                <li><a th:href="@{/comptes}">Comptes</a></li>
                <li><a th:href="@{/clients}">Clients</a></li>
                <li><a th:href="@{/operation}">Operations</a></li>
            </ul>
            <ul class="nav navbar-nav navbar-right">
                <li><a th:href="@{/Login?Logout}">Logout</a></li>
            </ul>
        </div>
    </div>
</header>
<section layout:fragment="content1"></section>
<footer>
    <div class="navbar navbar-default navbar-fixed-bottom">
        <div class="text-center">
            <small> copyright@2017 adresse:Avenue Jaich Hay Salam Meknes </small>
        </div>
    </div>

```

```

    </div>
</footer>
</body>
</html>

```

### - Compte.html

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
xmlns:sec="http://www.thymeleaf.org/extras/spring-security"
layout:decorator="template1">
<div layout:fragment="content1">
    <div class="col-md-6">
        <div class="panel panel-primary">
            <div class="panel panel-heading">Consultation d'un compte</div>
            <div class="panel panel-body">
                <form th:action="@{/consultercompte}" method="get">
                    <div class="input-group">
                        <input type="text" class="form-control" th:value="${codeCte}"
id="codeCte" name="codeCte" placeholder="taper ici" />
                        <span class="input-group-btn">
                            <button class="btn btn-default" type="submit"
name="rechercher">OK
                            <!-- <span class="glyphicon glyphicon-search" aria-
hidden="true"></span> -->
                        </button>
                    </span>
                </div>
            </form>
            <div class="text-danger" th:if="${exception}"
th:text="${exception.message}"></div>
        </div>
        <div class="panel panel-primary" th:if="${compte}">
            <div class="panel panel-heading">Informations sur le compte</div>
            <div class="panel panel-body">
                <div>
                    <label class="control-label">Client :</label>
                    <label class="control-label"
th:text="${compte.client.nom}"></label>
                </div>
                <div>
                    <label class="control-label">Code compte :</label>
                    <label class="control-label"
th:text="${compte.code}"></label>
                </div>
                <div>
                    <label class="control-label">Solde :</label>
                    <label class="control-label"
th:text="${compte.solde}"></label>
                </div>
                <div>
                    <label class="control-label">Date Creation :</label>
                    <label class="control-label"
th:text="${compte.dateCreation}"></label>
                </div>
                <div>
                    <label class="control-label">Type compte :</label>

```

```

        <label class="control-Label"
th:text="${compte.class.simpleName}"></label>
    </div>
    <div th:if="${compte.class.simpleName=='CompteCourant'}">
        <label class="control-Label" >Decouvert :</label>
        <label class="control-Label"
th:text="${compte.decouvert}"></label>
    </div>
    <div th:if="${compte.class.simpleName=='CompteEpargne'}">
        <label class="control-Label" >Taux :</label>
        <label class="control-Label"
th:text="${compte.taux}"></label>
    </div>

</div>
</div>
</div>
<div class="col-md-6" >
    <div class="panel panel-primary" sec:authorize="hasRole('ROLE_ADMIN')"
th:if="${compte}">
        <div class="panel panel-heading">Operations sur le compte</div>
        <div class="panel panel-body">
            <form th:action="@{/saveOperation}" method="post">
                <div class="form-group">
                    <label class="control-Label">compte : </label>
                    <label class="control-Label"
th:text="${compte.code}"></label>
                    <input type="hidden" class="form-control"
th:value="${codeCte}" name="codeCte" />
                </div>
                <div class="form-group">
                    <input type="radio" name="typeOperation" value="Retrait"
th:text="Retrait" checked="checked"
onchange="document.getElementById('vers').style.display='none'"/>
                    <input type="radio" name="typeOperation" value="Versement"
th:text="Versement"
onchange="document.getElementById('vers').style.display='none'"/>
                    <input type="radio" name="typeOperation" value="Virement"
th:text="Virement"
onchange="document.getElementById('vers').style.display='block'"/>
                </div>
                <div class="form-group" id="vers" style="display: none;">
                    <label class="control-Label">Vers</label>
                    <input type="text" name="codeCteDes" class="form-
control"></input>
                </div>
                <div class="form-group">
                    <label class="control-Label">Montant</label>
                    <input type="text" name="montant" class="form-
control"></input>
                </div>
                <div class="text-danger"
th:text="${#httpServletRequest.getParameter('error')}">
            </div>
            <div>
                <button class="btn btn-primary" type="submit"
name="saveOp">Save</button>
            </div>

```

```

        </form>
    </div>
</div>
<div class="panel panel-primary" th:if="${compte}">
    <div class="panel panel-heading">List des operations</div>
    <div class="panel panel-body">
        <table class="table table-striped">
            <thead>
<tr><th>Numero</th><th>Type</th><th>Date</th><th>Montant</th></tr>
            </thead>
            <tbody>
                <tr th:each="op:${operations}">
                    <td th:text="${op.numero}"></td>
                    <td th:text="${op.class.simpleName}"></td>
                    <td th:text="${op.dateOp}"></td>
                    <td th:text="${op.Montant}"></td>
                </tr>
            </tbody>
        </table>
        <div>
            <ul class="nav nav-pills" >
                <li th:each="pg,status:${operations}"
th:class="${status.index==pageCourant}?'active':''">

                    <a
th:href="@{/consultercompte(codeCte=${compte.code},page=${status.index})}"
th:text="${status.index}"></a></li>
            </ul>
        </div>
    </div>
</div>
</div>
</html>

```

#### - Login.html

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<title>Authentication</title>
<link rel="stylesheet" type="text/css" th:href="@{/css/bootstrap.css}"
href="../static/css/bootstrap.css" />
<link rel="stylesheet" type="text/css" th:href="@{/css/myStyle.css}"
href="../static/css/myStyle.css" />
</head>
<body>
    <div class="col-md-4 col-md-offset-3 espace ">
        <div class="panel panel-primary">
            <div class="panel-heading">Authentication</div>
            <div class="panel-body">
                <div th:if="${param.error}" class="red">
                    Incorrect Username or password
                </div>
                <div th:if="${param.logout}" class="red">
                    You have been logged outs
                </div>
            </div>
        </div>
    </div>

```


```

        <form th:action="@{/Login}" method="post">
            <div>
                <label>Utilisateur :</label>
                <input type="text" name="username" />
            </div>
            <div>
                <label>Password :</label>
                <input type="password" name="password" />
            </div>
            <button type="submit">Login</button>
        </form>
    </div>
</div>
</div>
</div>
</body>
</html>

```

## Démonstration de l'application

### Interface d'authentification

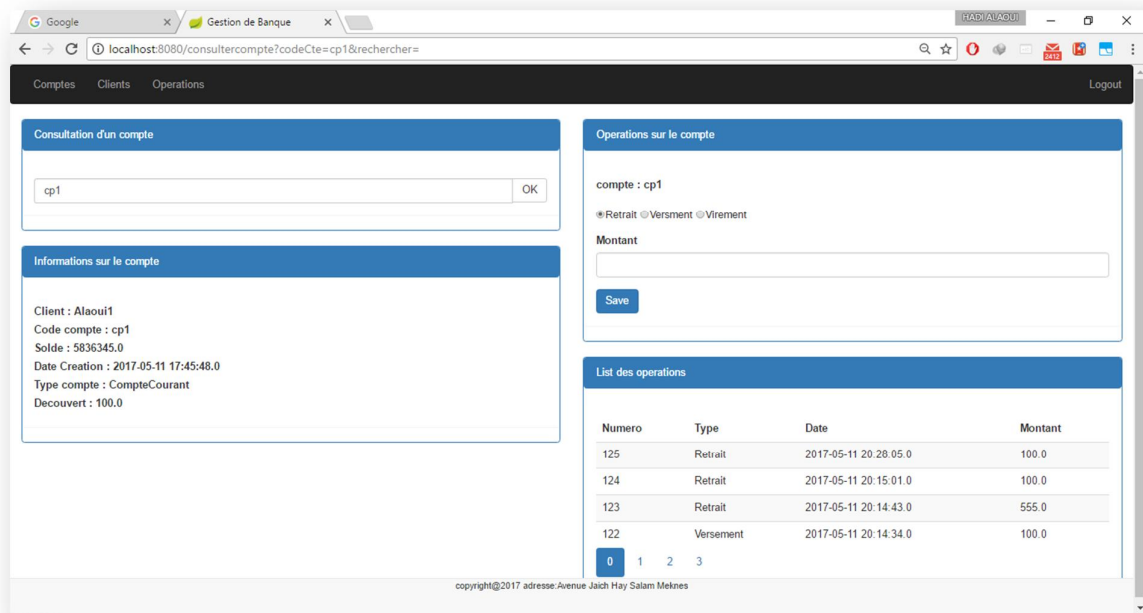


The screenshot shows a web form titled "Authentification" in a blue header. Below the header, there are two input fields: "Utilisateur :" followed by a text input box, and "Password :" followed by a password input box. Below these fields is a "Login" button.

l'authentification se fait en deux type :

- Administrateur : Il peut faire des operations
- Utilisateur : il n'a pas le droit de faire des operations

### Interface de consultation des comptes et Operations



## Interface de consultation des clients et ces comptes

