|  |
| --- |
| **Travaux Pratiques**  **FI BDCC S4 : Architecture distribuée JEE** TP : Gestion des comptes bancaires |

# Introduction :

La banque doit pouvoir atteindre plusieurs objectifs dans le cadre de la gestion quotidienne des comptes bancaires. Ces objectifs sont bien sûr complémentaires. Il lui faut :

Maîtriser les risques associés aux opérations bancaires réalisées sur le compte.

Améliorer la qualité des services rendus aux clients.

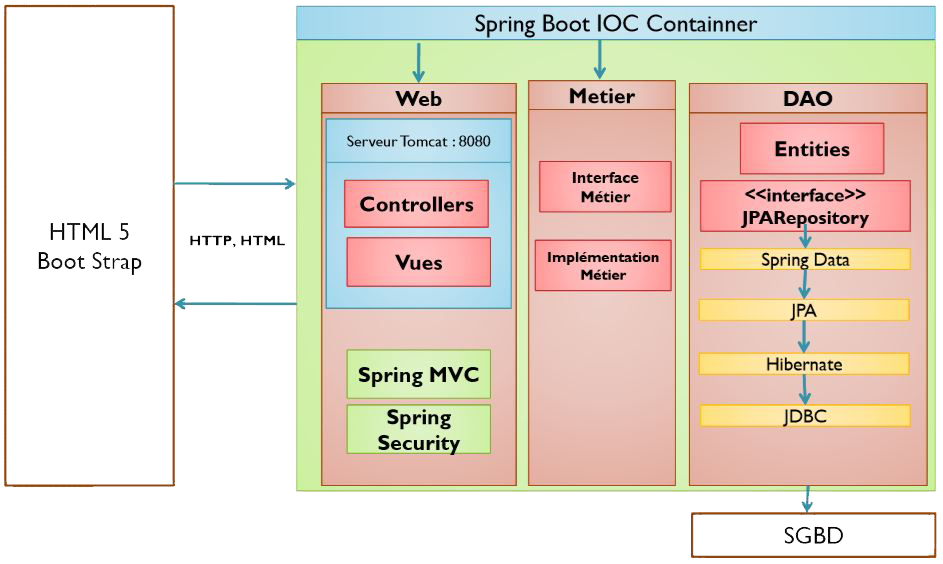
Rentabiliser ses comptes en équipant la clientèle des produits et services de la gamme.

# Enoncé :

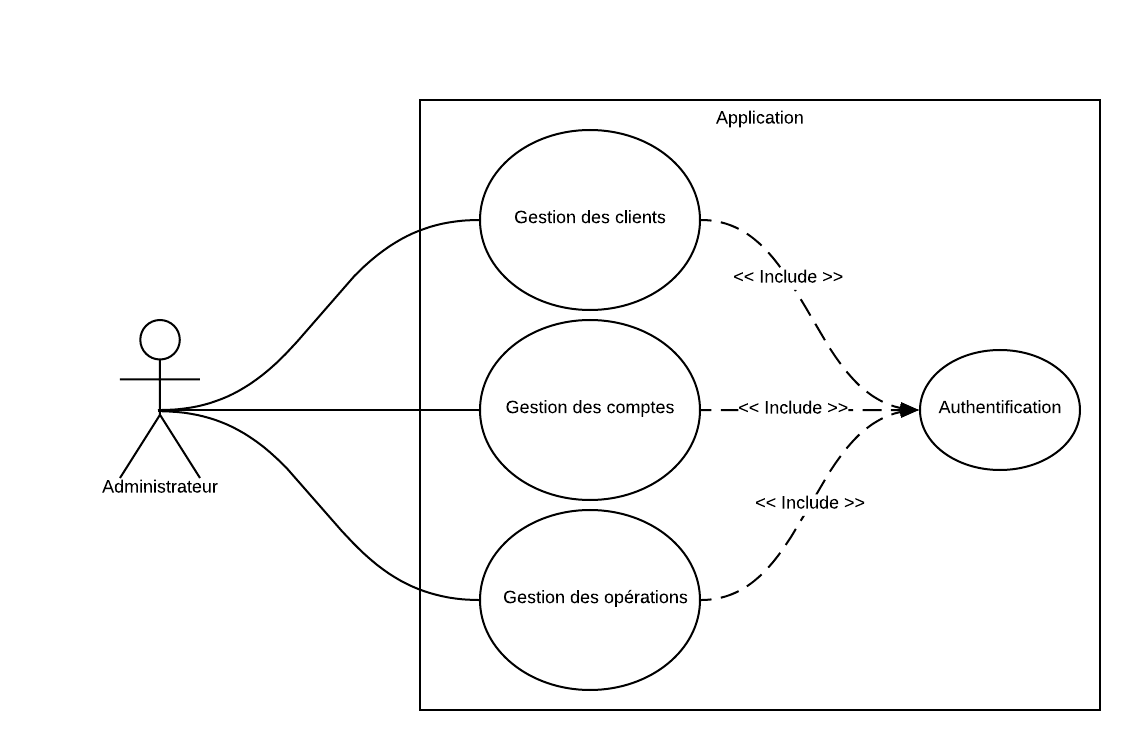
On souhaite créer une application qui permet de gérer des comptes bancaires en respectant les exigences fonctionnelles : Gérer les clients, Gérer les comptes, Gérer les opérations et les exigences techniques : les données sont stockées dans une base de données MySQL, l’application se compose de trois couches (la couche DAO, la couche Métier, la couche Web basée sur MVC coté serveur en utilisant Thymeleaf, la sécurité est basée sur Spring Security).

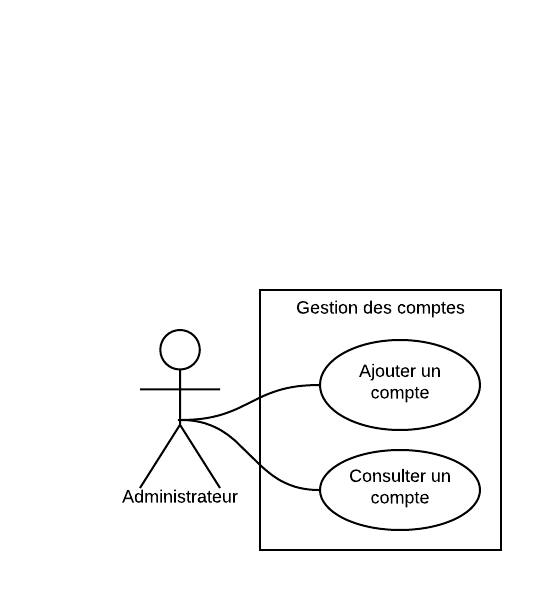
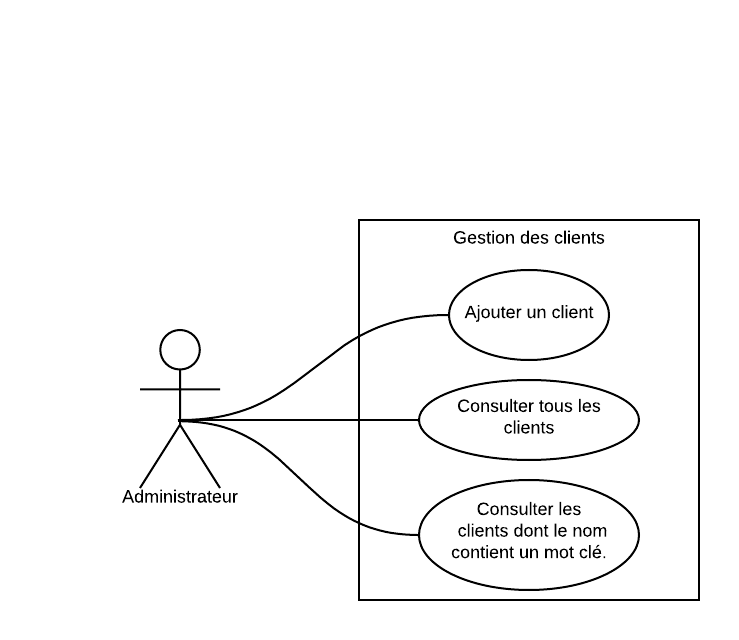
# Architecture et conception :

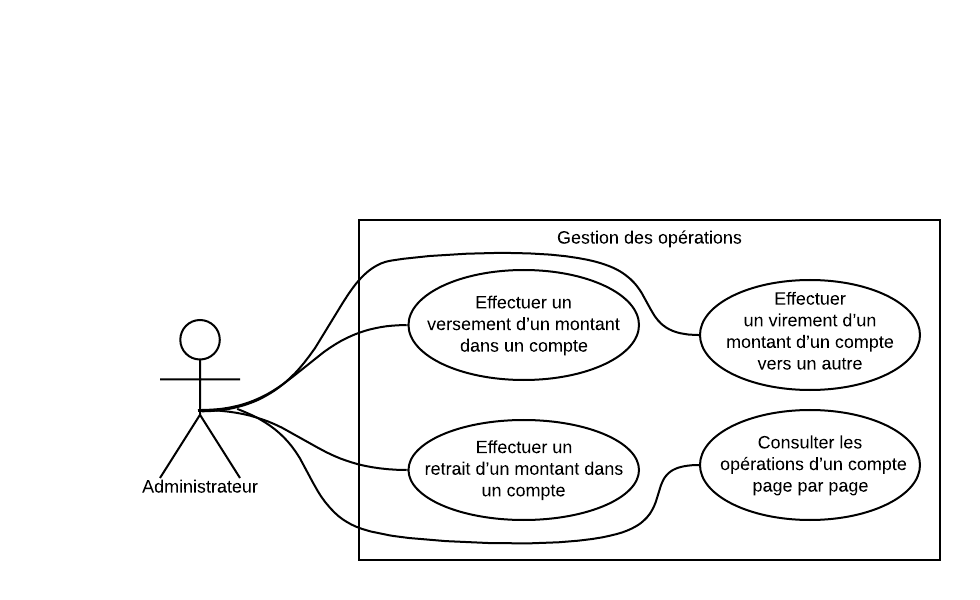
* + Architecture technique :



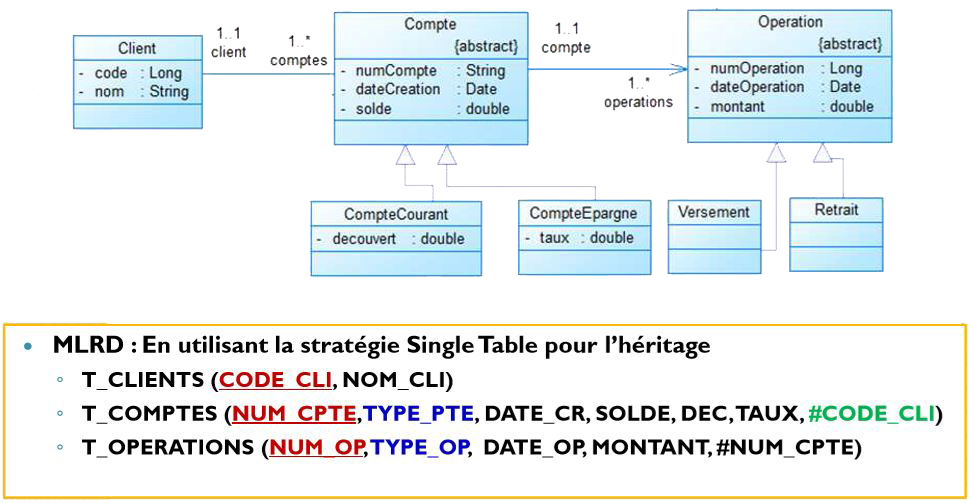
* + Diagramme de cas d’utilisation :







* + Diagramme de classes des entités et MLDR :



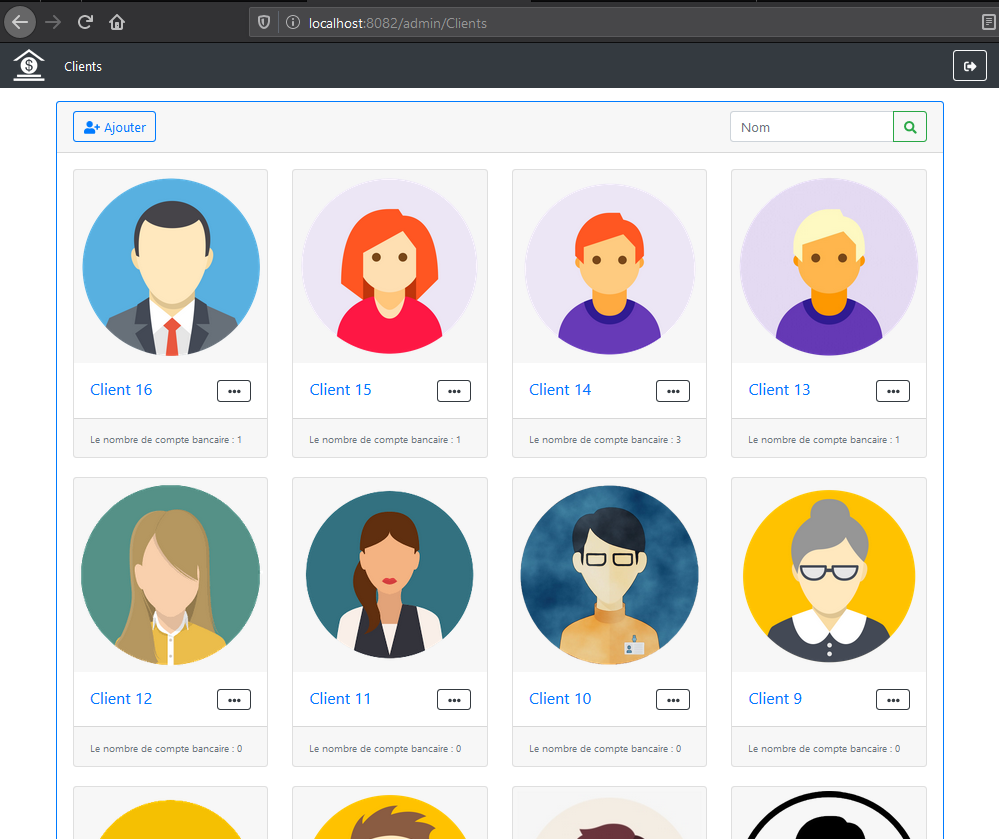
# Code source :

<https://github.com/Medomane/JEE/tree/master/Project%20Spring%20Boot%20MVC%20-%20Gestion%20des%20Comptes%20bancaires>.

# Exécution :

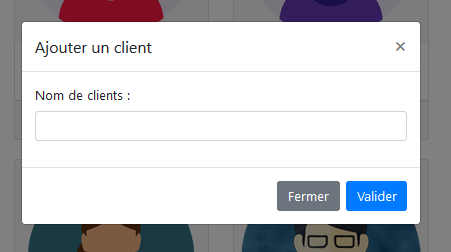
## Gestion des clients

### Consulter les clients :



### Ajouter un client :

Quand vous cliquez sur ajouter cette fenêtre va s’afficher :



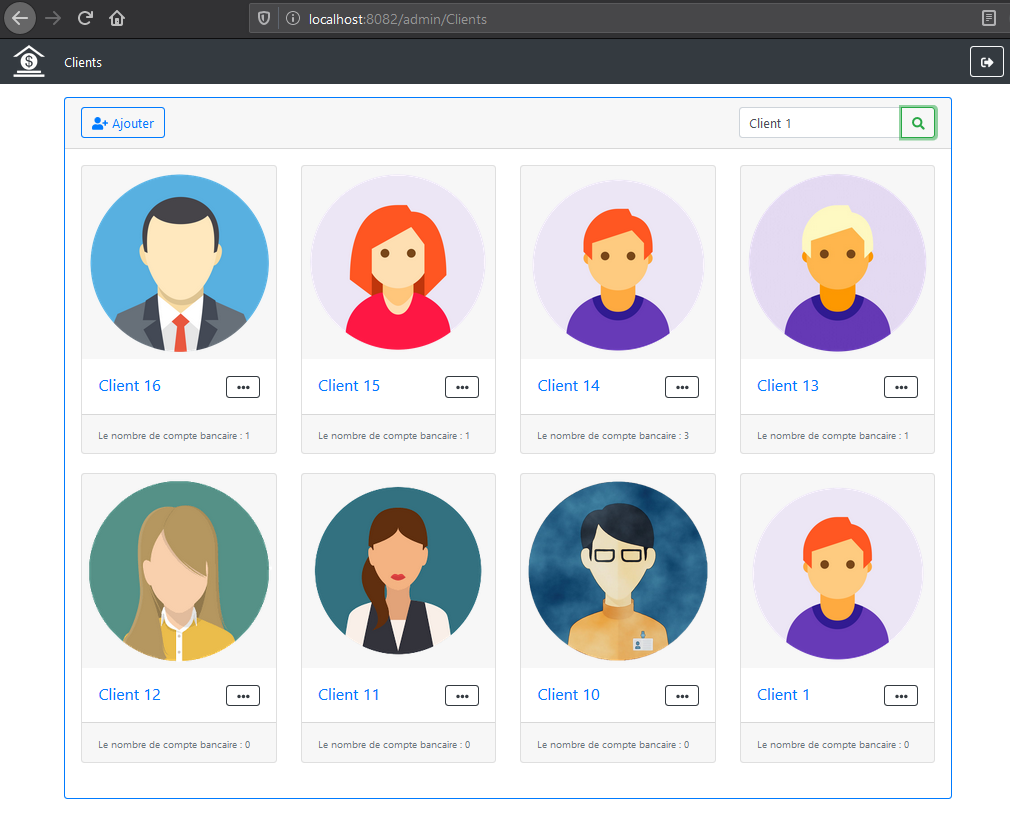
Quand vous cliquez sur valider : un nouveau client s’ajoute à la base de donnée et aussi il s’ajoute dans la liste des clients sans actualiser la page.

### Consulter les clients par nom :

Quand vous cliquez sur chercher



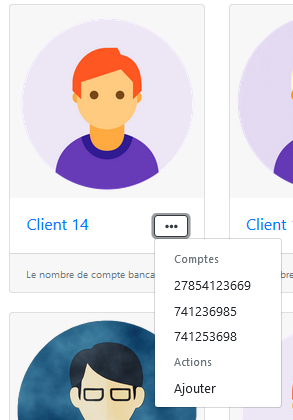
Vous récupérez tous les clients ayant un nom contient cette chaine



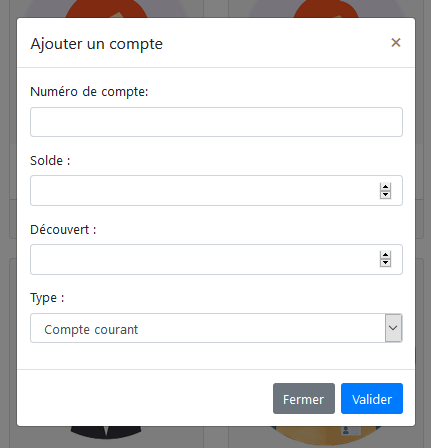
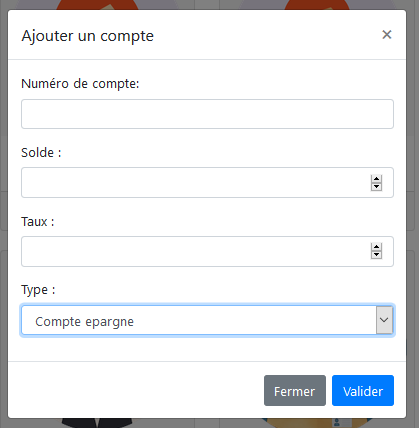
## Gestion des comptes

### Ajouter un compte :

Quand vous cliquer sur Ajouter



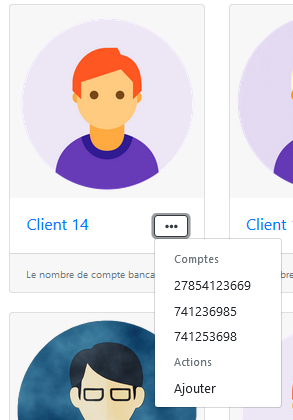
Cette fenêtre va s’affichée :

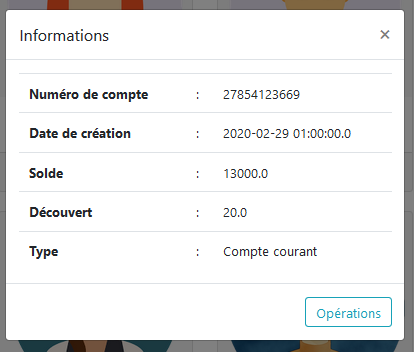
Vous pouvez choisir le type du compte puis saisir les informations, et quand vous cliquez sur valider un compte va s’ajouter au client ciblé.

### Consulter un compte :

Quand vous cliquez sur un compte :



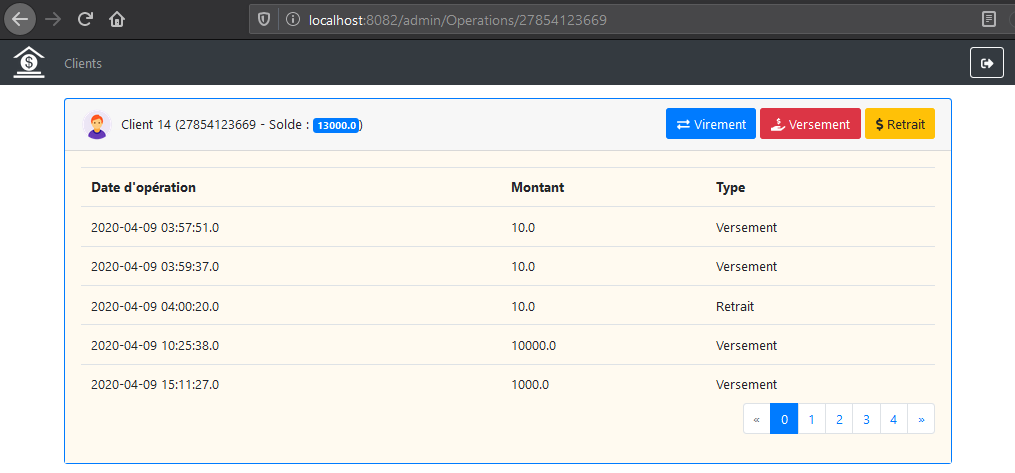
Une fenêtre de détail va s’afficher :



## Gestion des opérations

### Consulter les opérations page par page :

Quand vous cliquer sur Opérations :



Une liste des opérations du compte ciblé va s’afficher avec quelques informations supplémentaires, vous pouvez consulter page par page avec 5 éléments par page.

### Effectuer un versement d’un montant :

Quand vous cliquez sur Versement cette fenêtre va s’afficher :



Vous saisissez un montant puis vous cliquez sur valider, un versement va s’effectuer vers le compte ciblé.

### Effectuer un retrait d’un montant :

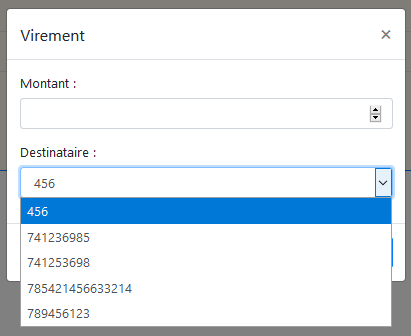
Quand vous cliquez sur Retrait cette fenêtre va s’afficher :



Vous saisissez un montant puis vous cliquez sur valider, un retrait va s’effectuer depuis le compte ciblé.

### Effectuer un virement d’un montant vers un autre compte :

Quand vous cliquez sur Virement cette fenêtre va s’afficher :

Vous saisissez un montant, vous choisissez un destinataire puis vous cliquez sur valider, un virement va s’effectuer depuis le compte ciblé vers le compte choisi.

# Conclusion :

Dans ce projet j’ai utilisé les modules de Spring Boot MVC et j’ai répondu aux exigences du projet, aussi vous allez voir que toutes les opérations dans le site s’exécutent sans l’actualisation de la page car j’ai utilisé les Fragments pour actualiser juste une partie de la page avec l’utilisation du Ajax.

Dans de Repository du projet vous trouverez une démonstration vidéo.