



**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**
Membre de **HONORIS UNITED UNIVERSITIES**

Projet JEE Spring Angular

Digital Banking

Realisé Par : Mohmmmed Jihad

Encadré par :

- Pr. Mohamed YOUSSEFI

Année Universitaire : 2022 - 2023

Plan

Introduction.....	3
I. Spécification des besoins.....	4
II. Conception.....	5
III. Test et documentation.....	6
IV. Demonstration.....	9
Conclusion.....	13

Introduction

Dans le cadre du module d'Architecture JEE et Middlewares, j'ai entrepris un projet passionnant visant à développer une application de banque numérique. En utilisant les frameworks Spring Boot et Angular, j'ai réussi à créer une application robuste et conviviale pour gérer les comptes bancaires des clients.

Mon application offre une large gamme de fonctionnalités permettant aux utilisateurs de gérer facilement leurs comptes. Ils peuvent effectuer des opérations de débit et de crédit, tout en profitant de la flexibilité d'avoir à leur disposition des comptes courants et des comptes épargne.

Pour garantir une implémentation solide, j'ai commencé par créer un projet Spring Boot, en utilisant les meilleures pratiques de conception pour la création des entités JPA. Les entités telles que Customer, BankAccount, Saving Account, CurrentAccount et AccountOperation ont été soigneusement conçues pour représenter les éléments clés du système bancaire.

Afin de faciliter l'accès et la manipulation des données, j'ai utilisé les interfaces JPA Repository fournies par Spring Data. Cela m'a permis d'interagir de manière efficace avec la base de données et d'assurer une gestion optimale des informations bancaires.

La qualité de mon application a été renforcée par l'utilisation de tests unitaires approfondis. J'ai veillé à valider la couche DAO de l'application en m'assurant du bon fonctionnement de chaque entité et de chaque interface Repository. Cette approche rigoureuse m'a permis de détecter et de corriger rapidement les éventuelles erreurs.

La couche de services a été développée en utilisant des DTO (Data Transfer Objects) et des mappers. Cela m'a permis de gérer efficacement les opérations métier et de convertir les objets pour les rendre compatibles avec les API. Ainsi, j'ai pu offrir une expérience utilisateur fluide et cohérente.

Pour exposer les fonctionnalités de l'application via des API REST, j'ai développé la couche Web en utilisant les RestControllers. Cela permet aux clients de communiquer facilement avec l'application et de profiter de toutes les fonctionnalités offertes par la banque numérique.

L'interface utilisateur attrayante et réactive a été réalisée grâce à Angular. En combinant les meilleures pratiques de conception et une attention particulière aux détails, j'ai pu offrir une expérience utilisateur agréable et intuitive.

La sécurité est une priorité essentielle pour toute application bancaire. C'est pourquoi j'ai intégré Spring Security et utilisé JSON Web Tokens (JWT) pour l'authentification et l'autorisation des utilisateurs. Grâce à ces mesures de sécurité, les clients peuvent avoir confiance en la confidentialité et l'intégrité de leurs informations personnelles et financières.

Dans le cadre de ce rapport, je fournis les répertoires Backend et Frontend contenant les codes sources complets de mon application. Vous y trouverez également une vidéo de démonstration de 5 minutes qui met en lumière les principales fonctionnalités de l'application.

J'ai consciencieusement respecté les délais du projet et je suis fier du résultat obtenu. Ce projet démontre ma capacité à concevoir et à développer une application JEE de qualité, en utilisant les meilleures technologies disponibles.

En conclusion, ce projet de banSpécification des besoins

1) Besoins fonctionnel :

- Gestion des comptes bancaires : L'application doit permettre la création, la consultation, la modification et la suppression des comptes bancaires pour les clients.
- Opérations de débit et de crédit : Les utilisateurs doivent pouvoir effectuer des opérations de débit et de crédit sur les comptes bancaires.
- Catégorisation des comptes : Les comptes bancaires doivent être catégorisés en comptes courants et comptes épargne, avec des fonctionnalités spécifiques pour chaque type.
- Gestion des clients : L'application doit permettre la gestion des clients, incluant la création, la consultation et la modification des informations personnelles.
- Historique des opérations : Les utilisateurs doivent pouvoir consulter l'historique des opérations effectuées sur un compte bancaire, y compris les détails tels que la date, le montant et le type d'opération.

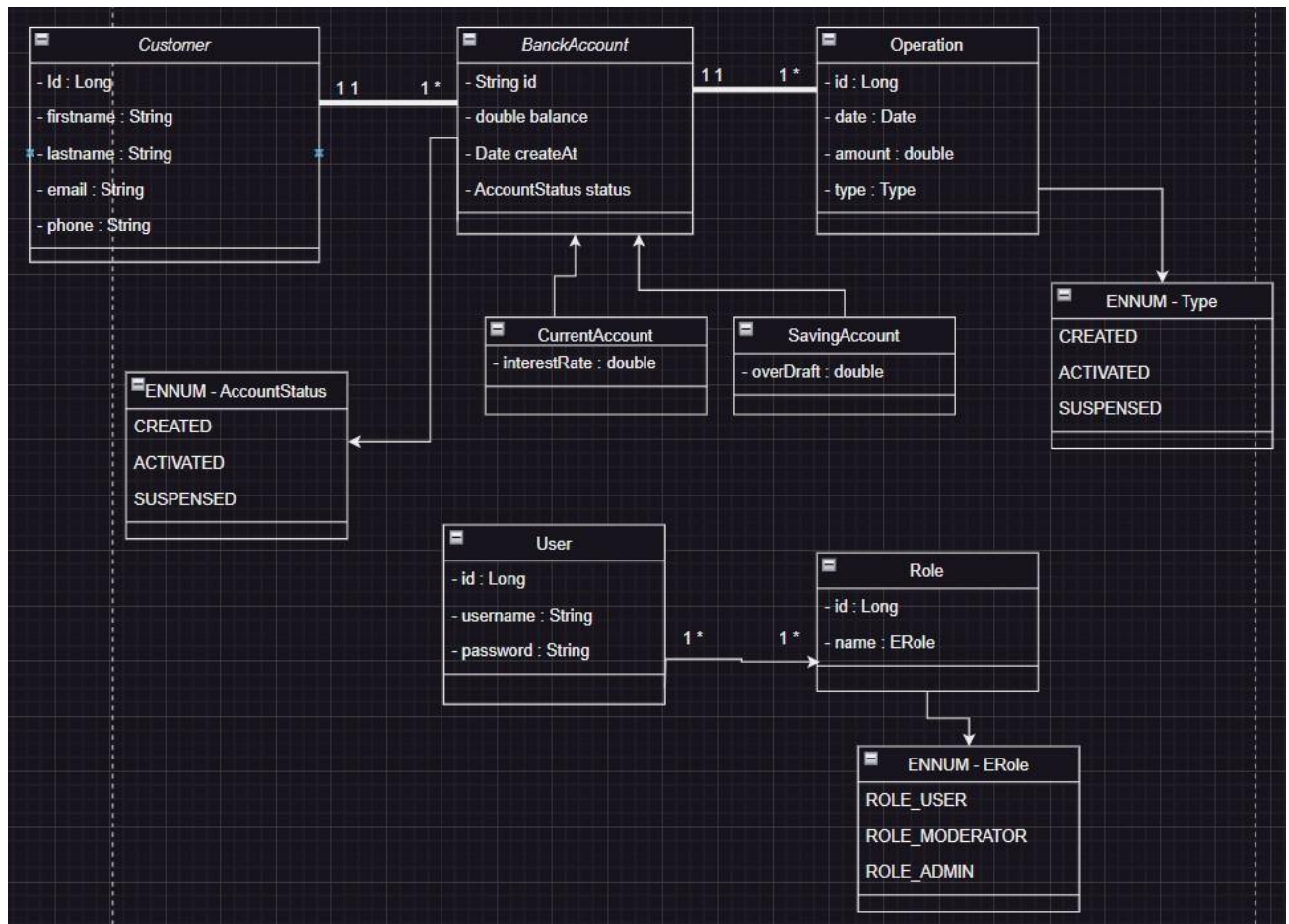
2) Besoins non fonctionnel :

- Sécurité : L'application doit garantir la sécurité des données sensibles, notamment en mettant en place un système d'authentification et de contrôle d'accès.
- Performance : L'application doit être performante, en assurant une réponse rapide aux requêtes des utilisateurs et en gérant efficacement les opérations sur les comptes bancaires.
- Convivialité de l'interface utilisateur : L'interface utilisateur doit être conviviale, intuitive et facile à utiliser, afin de permettre aux utilisateurs de naviguer et d'effectuer des opérations sans difficulté.
- Scalabilité : L'application doit être conçue de manière à pouvoir gérer un nombre croissant de clients et de transactions, en permettant une évolutivité facile.

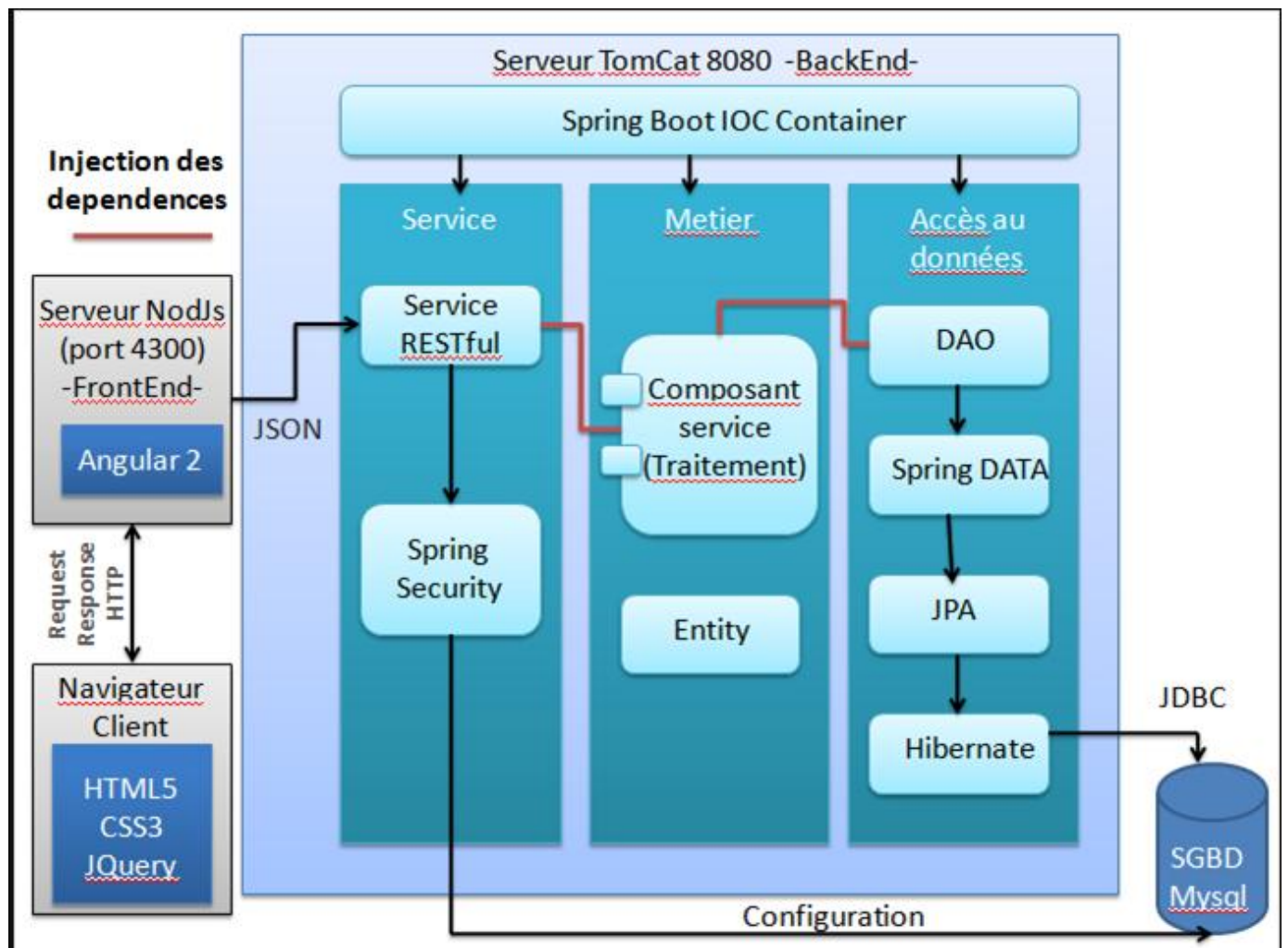
- Fiabilité : L'application doit être fiable et stable, minimisant les risques de perte de données ou de dysfonctionnements.

II. Conception

1) Diagramme de classe.



2) Diagramme d'architecture technique du projet



III. Test et documentation

Pour la documentation du projet, j'ai utilisé Swagger au niveau du backend et Compodoc au niveau du frontend. Swagger est un outil populaire pour la documentation des API REST. Il permet de générer automatiquement une documentation détaillée des API, en incluant les routes, les paramètres, les réponses attendues, ainsi que des exemples d'utilisation. J'ai intégré Swagger dans mon projet backend, développé avec Spring Boot, afin de fournir une documentation complète et facilement accessible pour les développeurs et les utilisateurs.

Concernant le frontend, j'ai utilisé Compodoc, un outil spécifique à Angular, pour générer la documentation du code source. Compodoc facilite la documentation des composants, des modules, des services et autres éléments du code Angular. Il permet de générer une documentation interactive et bien structurée, qui aide à comprendre l'architecture et le fonctionnement du frontend.

Pour les tests, j'ai utilisé Postman. Postman est un outil de développement d'API qui permet d'envoyer des requêtes HTTP et de vérifier les réponses. J'ai utilisé Postman pour tester mes API au niveau du backend et m'assurer qu'elles fonctionnent correctement. Il m'a permis de vérifier les différentes routes, les paramètres et les réponses des API, ce qui a contribué à assurer la qualité et la fiabilité de mon application.

En résumé, j'ai utilisé Swagger pour la documentation du backend, Compodoc pour la documentation du frontend et Postman pour les tests au niveau du backend. Ces outils ont joué un rôle essentiel dans le développement, la documentation et la vérification de mon projet de banque numérique, en assurant sa cohérence, sa qualité et sa facilité d'utilisation.



1) Le test

The screenshot shows a REST client interface with the following details:

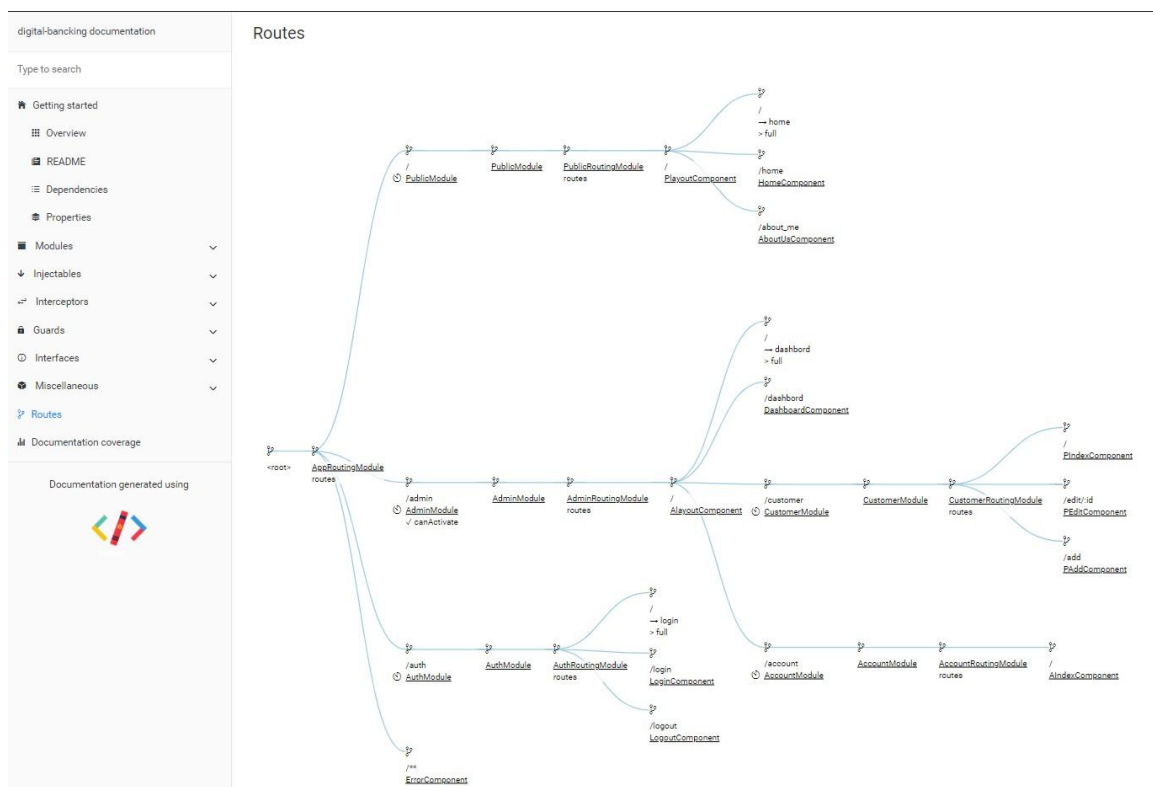
- URL: `http://localhost:8085/api/accounts/`
- Method: `GET`
- Authorization: `Bearer Token`
- Status: `200 OK`, Time: `187 ms`, Size: `31.54 KB`
- Response Body (JSON):

```
{
  "type": "CurrentAccount",
  "id": "03c5fe4a-63b0-4187-b224-00243931e234",
  "balance": 9624621.77372446,
  "createdAt": "2023-05-24T01:37:38.000+00:00",
  "status": null,
  "customerDTO": {
    "id": 3,
    "name": "Mohamed",
    "email": "Mohamed@gmail.com"
  },
  "overDraft": 9000.0
},
{
  "type": "CurrentAccount",
  "id": "05f7bba9-8e03-41fa-9ca4-edf45698156d",
  "balance": 9897364.418467376,
  "createdAt": "2023-05-24T01:41:18.000+00:00",
  "status": null,
  "customerDTO": {
    "id": 3,
    "name": "Mohamed",

```

2) La documentation frontend avec compodoc

- Le system de routage




- Les dependances

Type to search

- Getting started
- Overview
- README
- Dependencies
- Properties
- Modules
- Injectables
- Interceptors
- Guards
- Interfaces
- Miscellaneous
- Routes
- Documentation coverage

Documentation generated using



Dependencies

```

@angular/animations : ^16.0.0
@angular/common : ^16.0.0
@angular/compiler : ^16.0.0
@angular/core : ^16.0.0
@angular/forms : ^16.0.0
@angular/platform-browser : ^16.0.0
@angular/platform-browser-dynamic : ^16.0.0
@angular/router : ^16.0.0
@ng-bootstrap/ng-bootstrap : ^14.1.1
@types/chart.js : ^2.9.37
bootstrap : ^5.2.3
bootstrap-icons : ^1.10.5
chart.js : ^3.9.1
ng2-charts : ^3.1.2
rxjs : ~7.8.0
tslib : ^2.3.0
zone.js : ~0.13.0

```


- Les intercepteurs

digital-banking documentation

Type to search

- Getting started
- Overview
- README
- Dependencies
- Properties
- Modules
- Injectables
- Interceptors
- Guards
- Interfaces
- Miscellaneous
- Routes
- Documentation coverage

Documentation generated using



Interceptors / TokenInterceptor

Info Source

```

1 import { Injectable } from '@angular/core';
2 import { HttpRequest, HttpHandler, HttpEvent, HttpInterceptor, HTTP_INTERCEPTORS } from '@angular/common/http';
3 import { Observable, catchError, throwError } from 'rxjs';
4 import { TokenService } from '../_services/token.service';
5
6 @Injectable()
7 export class TokenInterceptor implements HttpInterceptor {
8
9     constructor(private tokenService: TokenService) {}
10
11     intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
12         const token = this.tokenService.getToken()
13         if (token !== null) {
14             const cloned = request.clone({
15                 headers: {
16                     Authorization: `Bearer ${token}`
17                 }
18             });
19             return next.handle(cloned).pipe(
20                 catchError(error =>{
21                     console.log(error)
22                     if(error.status == 401){
23                         this.tokenService.clearToken()
24                     }
25                     return throwError('Session expired')
26                 })
27             )
28         }
29
30         return next.handle(request);
31     }
32 }
33
34 export const TokenInterceptorProvider = {
35     provide: HTTP_INTERCEPTORS,
36     useClass: TokenInterceptor,
37     multi: true
38 };

```

IV. Demonstration

L'application est divisée en deux parties distinctes : une partie publique qui ne nécessite pas d'authentification et une partie admin qui requiert une authentification sécurisée en utilisant un jeton JWT (JSON Web Token).

La partie publique de l'application permet aux utilisateurs d'accéder à certaines fonctionnalités sans avoir à s'authentifier. Cela peut inclure la consultation des informations générales sur les services offerts par la banque, la visualisation des taux d'intérêt, ou encore l'accès à des pages d'aide et de contact. Les utilisateurs peuvent explorer ces fonctionnalités sans avoir besoin de fournir des informations d'identification.

D'autre part, la partie admin de l'application est réservée aux administrateurs ou au personnel autorisé. Pour accéder à cette partie, une authentification est requise. L'authentification est réalisée de manière sécurisée en utilisant un jeton JWT. Lorsqu'un administrateur se connecte avec ses informations d'identification, l'application génère un jeton JWT qui est ensuite utilisé pour authentifier et autoriser les requêtes vers les fonctionnalités administratives. Ce mécanisme de sécurité garantit que seuls les utilisateurs authentifiés et autorisés peuvent accéder et effectuer des opérations dans la partie admin de l'application.

En divisant l'application en deux parties avec des exigences d'authentification différentes, nous assurons une expérience utilisateur conviviale pour les utilisateurs publics, tout en garantissant la sécurité et la confidentialité des fonctionnalités administratives.

1) Interface public

2) Interface Administrateur

- la page customer

E-Bank Home Customers Accounts

Add customer:

Name:

Mohamed jihad
















Name is required.

Email:

JIHAD_MOHAMMED@EMSI-EDU.MA

Save

E-Bank Home Customers Accounts

Customers			
+ New customer		Keyword	Search
ID	Name	Email	
1	Hassan	Hassan@gmail.com	   View accounts
2	Imane	Imane@gmail.com	   View accounts
3	Mohamed	Mohamed@gmail.com	   View accounts
4	jihad	jihad@gmail.com	   View accounts
5	Mohamedjihad	JIHAD_MOHAMMED@EMSI-EDU.MA	   View accounts

Customer saved!

- la page View account

Customers			
+ New customer		Keyword	jihad <input type="button" value="Q"/>
ID	Name	Email	
4	jihad	jihad@gmail.com	✎ ✖ View accounts
5	Mohamedjihad	JIHAD_MOHAMMED@EMSI-EDU.MA	✎ ✖ View accounts

- la page Operation account

Accounts

Account ID:0ca3b97a-f5a6-43ef-b5f7-f2d179c6343c

Balance: 1,812,554.08 MAD

Operations:

ID	Date	Type	Amount
132	17-06-2023 22:23:34	DEBIT	2,364.08
133	17-06-2023 22:23:34	CREDIT	13,144.82
134	17-06-2023 22:23:34	DEBIT	7,028.53
135	17-06-2023 22:23:34	CREDIT	105,604.69
136	17-06-2023 22:23:34	DEBIT	4,308.59

Previous

0

1

2

3

4

5

6

7

8

9

10

11

Next

Operations

☐ Debit:

☐ Credit:

☐ Transfer:

Amount:

0

Description:

Save

- Recherche Customer

Customers			
+ New customer		Keyword	jihad <input type="button" value="Q"/>
ID	Name	Email	
4	jihad	jihad@gmail.com	✎ ✖ View accounts
5	Mohamedjihad	JIHAD_MOHAMMED@EMSI-EDU.MA	✎ ✖ View accounts


- Modifier customer

E-Bank Home Customers Accounts

Edit customer:

Name:


Email:
















 Save

E-Bank Home Customers Accounts

Customers

[+ New customer](#)

Keyword Search 

ID	Name	Email	
1	Hassanjihad	Hassan@gmail.com	   View accounts
2	Imane	Imane@gmail.com	   View accounts
3	Mohamed	Mohamed@gmail.com	   View accounts
4	jihad	jihad@gmail.com	   View accounts
5	Mohamedjihad	JIHAD_MOHAMMED@EMSI-EDU.MA	   View accounts

Customer edited!

- Ajoute Customer

Conclusion

En conclusion, ce projet de banque numérique réalisé avec succès en utilisant Spring Boot pour le backend et Angular pour le frontend a permis la mise en place des fonctionnalités essentielles de gestion des comptes bancaires, avec une distinction claire entre la partie publique et la partie administration.

Cependant, pour améliorer encore davantage notre application et la rendre plus performante, évolutive et résiliente face à une charge croissante, il serait judicieux de considérer une approche basée sur l'architecture de microservices. Les microservices offrent une architecture logicielle dans laquelle une application est décomposée en services indépendants, pouvant être développés, déployés et mis à l'échelle de manière autonome.

L'adoption d'une architecture de microservices nous permettrait de découpler les différentes fonctionnalités de l'application, telles que la gestion des clients, la gestion des comptes et les opérations bancaires, en services distincts. Chaque service pourrait être développé et déployé de manière indépendante, ce qui faciliterait la maintenance, la mise à l'échelle et la gestion des services.

De plus, en utilisant des technologies telles que Kubernetes pour l'orchestration des conteneurs et la gestion des déploiements, nous pourrions bénéficier d'une mise à l'échelle automatique, d'une résilience améliorée et d'une gestion simplifiée de l'infrastructure.

En somme, l'adoption d'une approche basée sur les microservices offrirait une flexibilité accrue, une évolutivité optimisée et une meilleure gestion des charges de travail pour notre application de banque numérique. Cela nous permettrait de répondre aux exigences croissantes des utilisateurs tout en facilitant la maintenance et le développement continu de l'application.