

# Projet JEE Spring Angular Digital Banking

**Realiseé Par : Otmani Idrissi Mokhtar**

**Encadré par :**

* Pr. Mohamed YOUSSFI

Année Universitaire : 2022 - 2023

# Plan

[Introduction 3](#_TOC_250005)

1. [Spécification des besoins 4](#_TOC_250004)
2. [Conception 5](#_TOC_250003)
3. [Test et documentation 6](#_TOC_250002)
4. [Demonstration 9](#_TOC_250001)

[Conclusion 13](#_TOC_250000)

## Introduction

Ce rapport présente le développement d'une application de banque numérique dans le cadre du module d'Architecture JEE et Middlewares. L'objectif de ce projet passionnant était de créer une application robuste et conviviale pour gérer les comptes bancaires des clients en utilisant les frameworks Spring Boot et Angular.

L'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 débuté par la création d'un projet Spring Boot en appliquant les meilleures pratiques de conception pour les 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. Cette approche m'a permis d'interagir efficacement avec la base de données et d'assurer une gestion optimale des informations bancaires.

La qualité de l'application a été renforcée grâce à l'utilisation de tests unitaires approfondis. J'ai veillé à valider la couche DAO 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 d'éventuelles erreurs.

La couche de services a été développée en utilisant des DTO (Data Transfer Objects) et des mappers, permettant une gestion efficace des opérations métier et la conversion des 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 facilite la communication entre les clients et l'application, leur permettant de profiter pleinement 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 étant une priorité essentielle pour toute application bancaire, j'ai intégré Spring Security et utilisé JSON Web Tokens (JWT) pour l'authentification et l'autorisation des utilisateurs. Ces mesures de sécurité garantissent la confidentialité et l'intégrité des informations personnelles et financières des clients.

Ce rapport fournit les répertoires Backend et Frontend contenant les codes sources complets de l'application. Vous y trouverez également une vidéo de démonstration de 5 minutes mettant en lumière les principales fonctionnalités de l'application.

En respectant consciencieusement les délais du projet, 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 banque numérique répond aux besoins fonctionnels et non fonctionnels spécifiés, offrant une expérience utilisateur intuitive

1 Besoins fonctionnels :

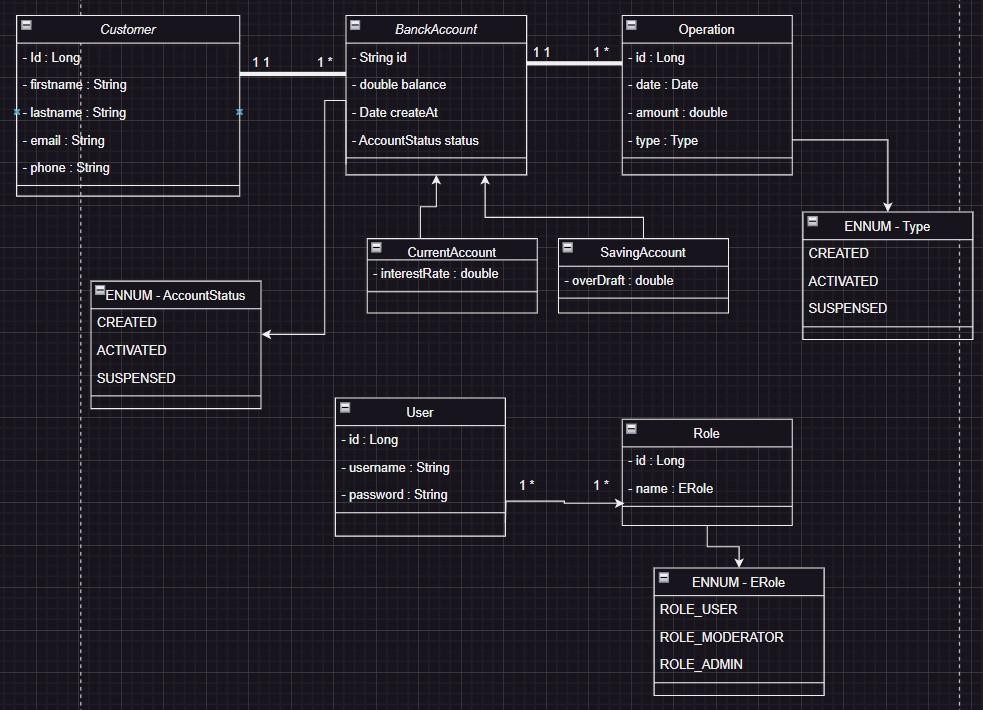
* Gestion des comptes bancaires : L'application doit permettre aux clients de créer, consulter, modifier et supprimer des comptes bancaires.
* Opérations de débit et de crédit : Les utilisateurs doivent pouvoir effectuer des opérations de débit et de crédit sur leurs comptes bancaires.
* Catégorisation des comptes : Les comptes bancaires doivent être classés en comptes courants et comptes épargne, avec des fonctionnalités spécifiques pour chaque type.
* Gestion des clients : L'application doit permettre la gestion des informations clients, y compris la création, la consultation et la modification des données personnelles.
* Historique des opérations : Les utilisateurs doivent pouvoir consulter l'historique des opérations effectuées sur leurs comptes bancaires, avec des détails tels que la date, le montant et le type d'opération.

2 Besoins non fonctionnels :

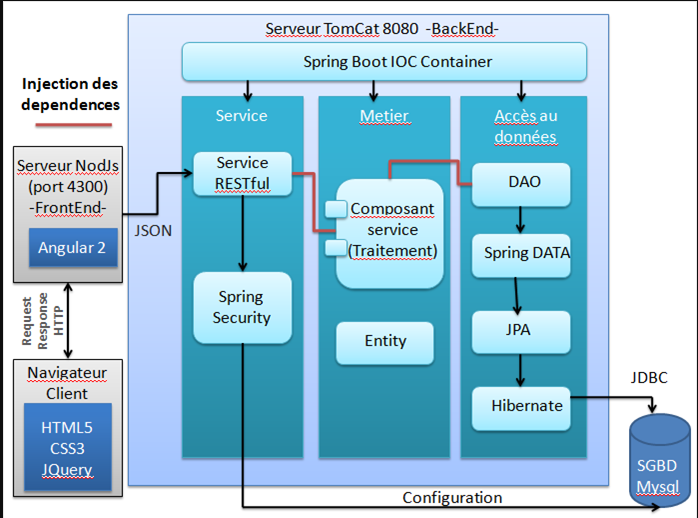
* Sécurité : L'application doit garantir la sécurité des données sensibles en mettant en place un système d'authentification et de contrôle d'accès.
* Performance : L'application doit être performante, en assurant des temps de réponse rapides aux requêtes des utilisateurs et en gérant efficacement les opérations sur les comptes bancaires.
* Convivialité de l'interface utilisateur : L'interface utilisateur doit être conviviale, intuitive et facile à utiliser, permettant 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..

### Conception

* 1. Diagramme de classe.



* 1. Diagramme d'architecture technique du projet



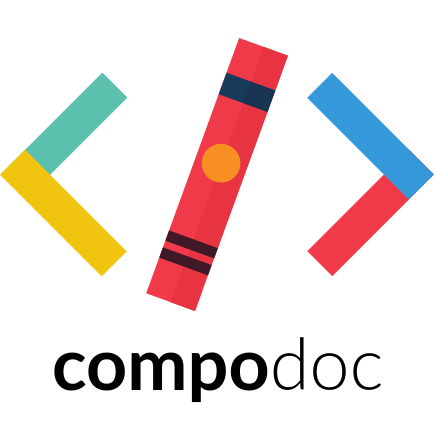
### Test et documentation

J'ai utilisé Swagger pour documenter le backend de mon projet et Compodoc pour documenter le frontend. Swagger est un outil largement utilisé pour documenter les API REST. Il génère automatiquement une documentation détaillée des API, incluant les routes, les paramètres, les réponses attendues et des exemples d'utilisation. J'ai intégré Swagger à mon projet backend développé avec Spring Boot pour offrir une documentation complète et facilement accessible aux développeurs et aux utilisateurs.

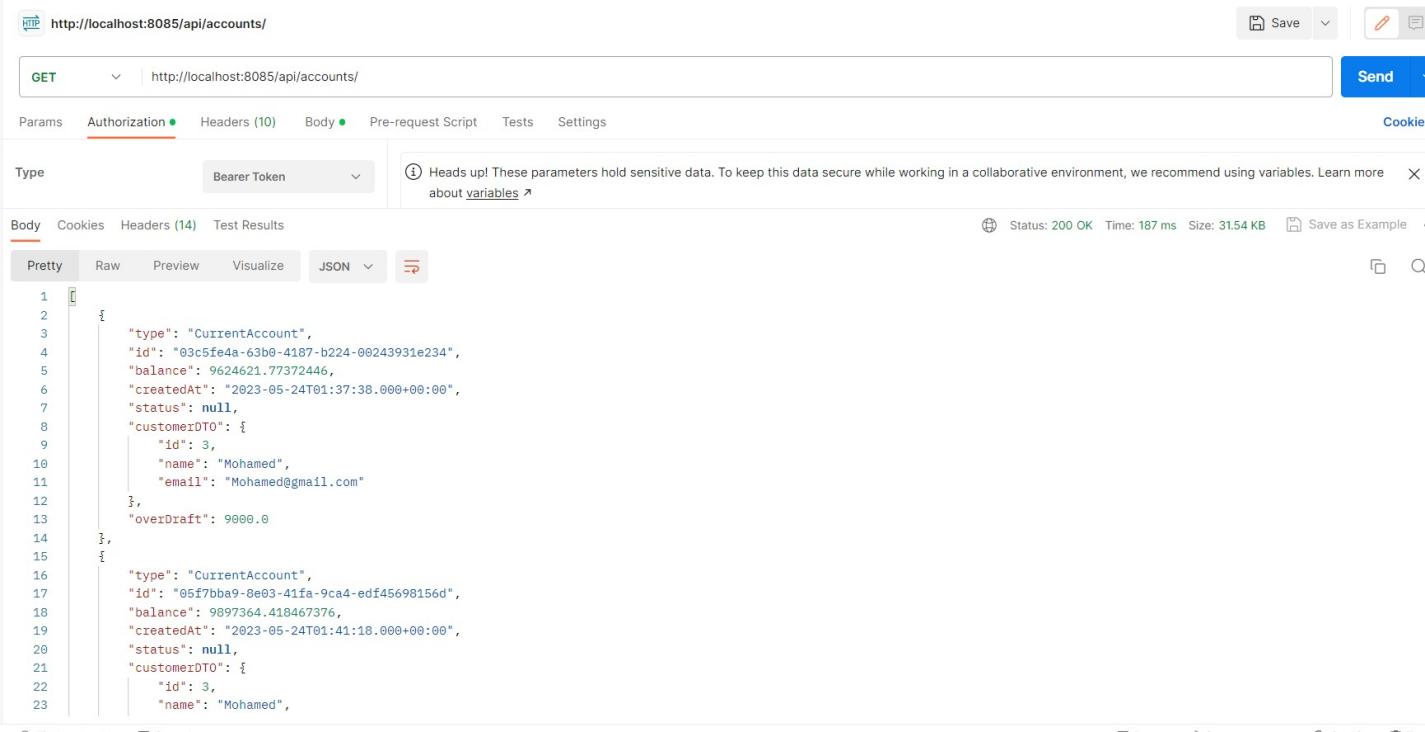
Pour le frontend, j'ai opté pour Compodoc, un outil spécifique à Angular, afin de générer la documentation du code source. Compodoc facilite la documentation des composants, modules, services et autres éléments du code Angular. Il produit une documentation interactive et bien structurée, qui aide à comprendre 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 de leur bon fonctionnement. Cela m'a permis de vérifier les différentes routes, paramètres et réponses des API, contribuant ainsi à assurer la qualité et la fiabilité de mon application.

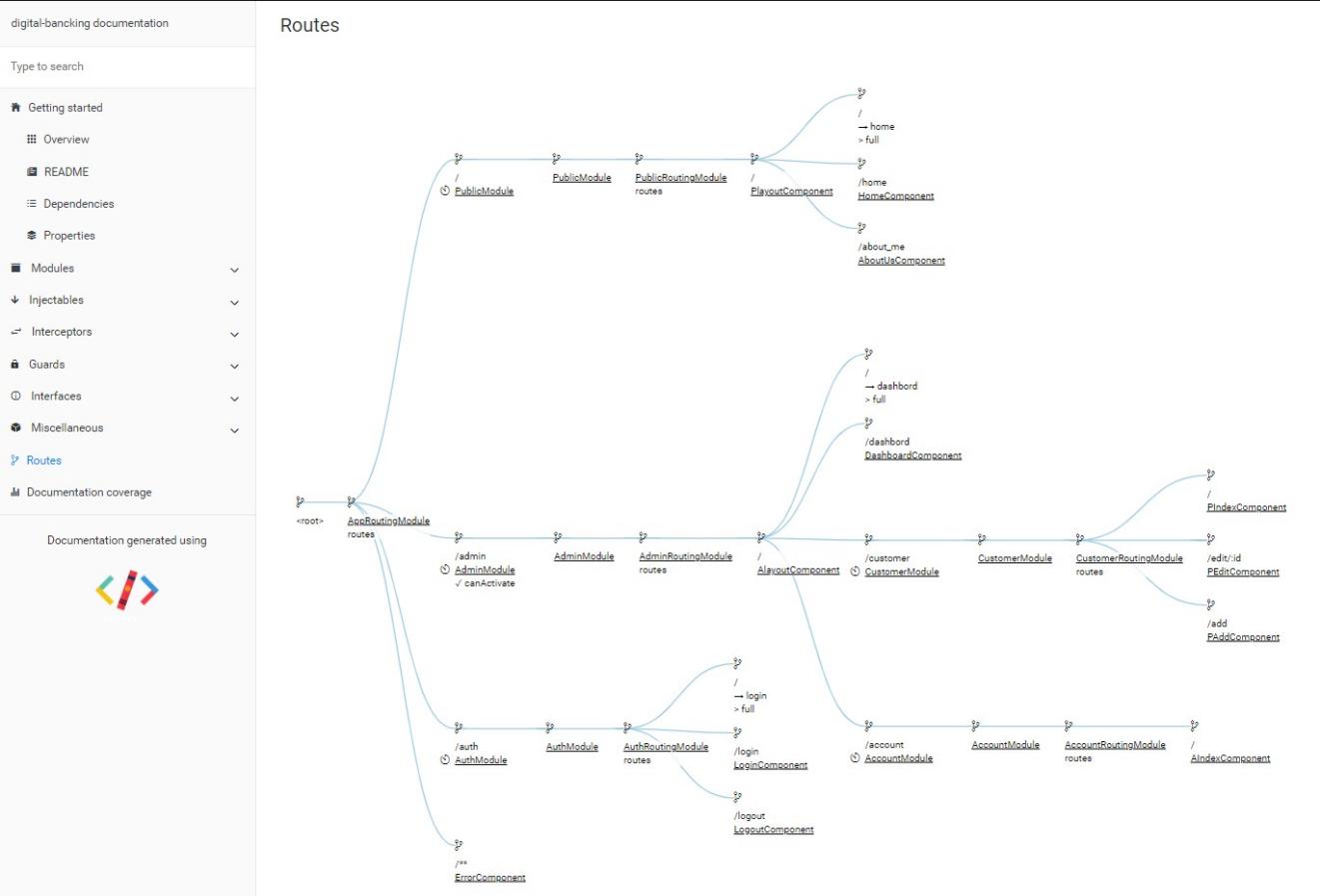
En résumé, j'ai utilisé Swagger pour la documentation du backend, Compodoc pour la documentation du frontend et Postman pour les tests au niveau du backend. Ces outils ont joué un rôle essentiel dans le développement, la documentation et la vérification de mon projet de banque numérique, garantissant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



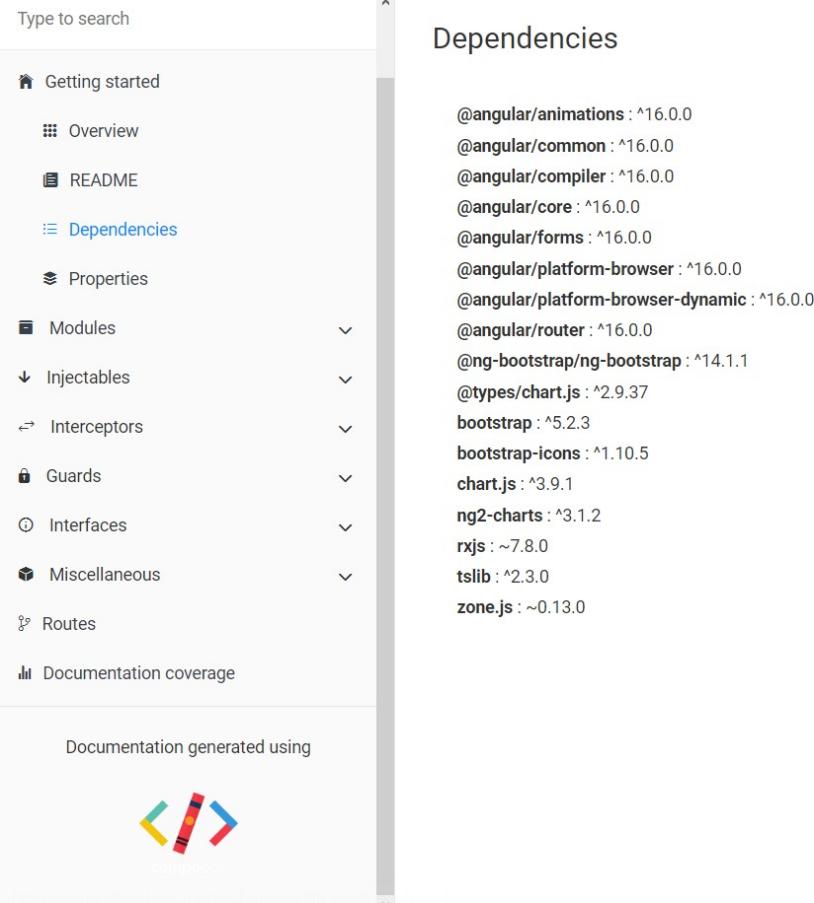
* 1. Le test



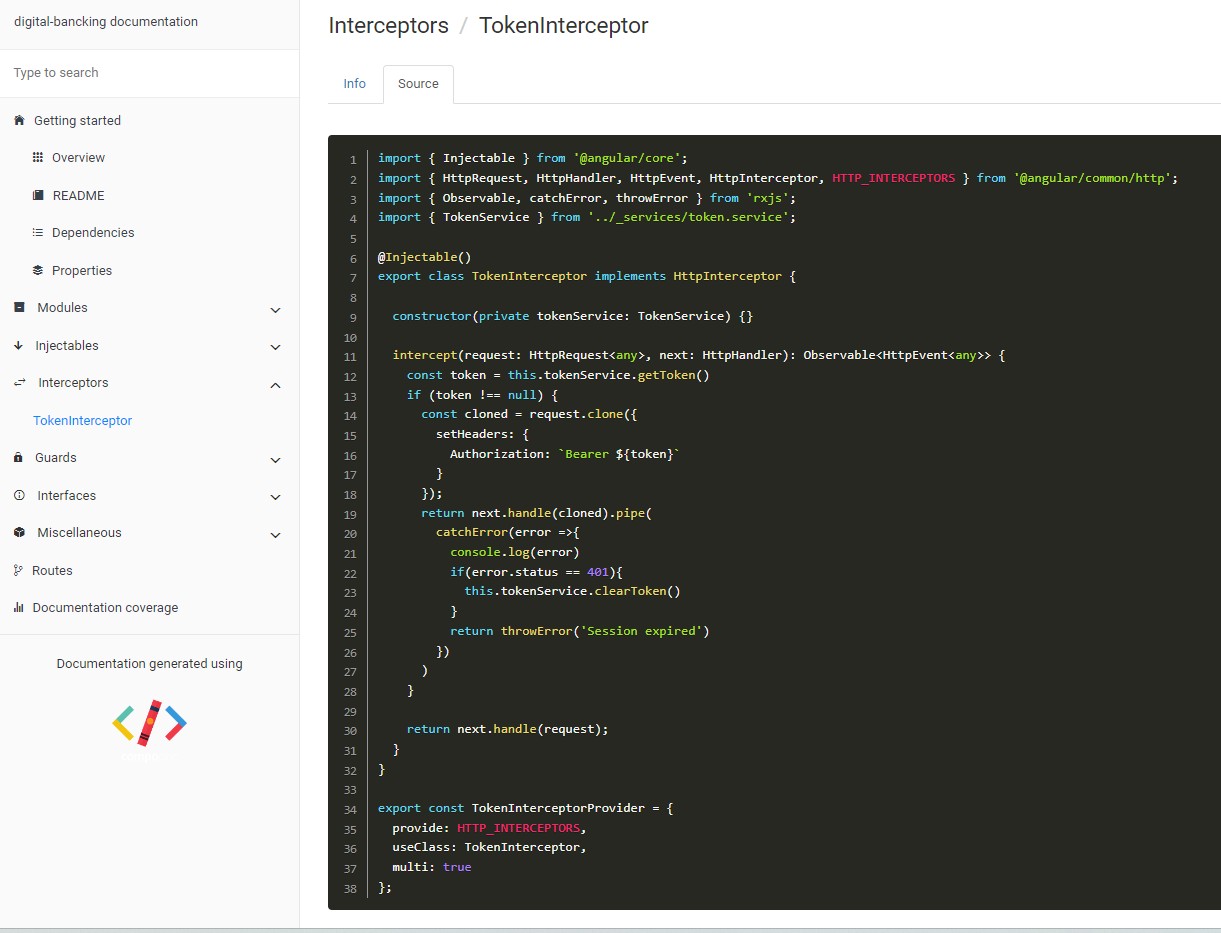
* 1. La documentation frontend avec compodoc
     + Le system de routage



* + - Les dependances



- Les intercepteurs



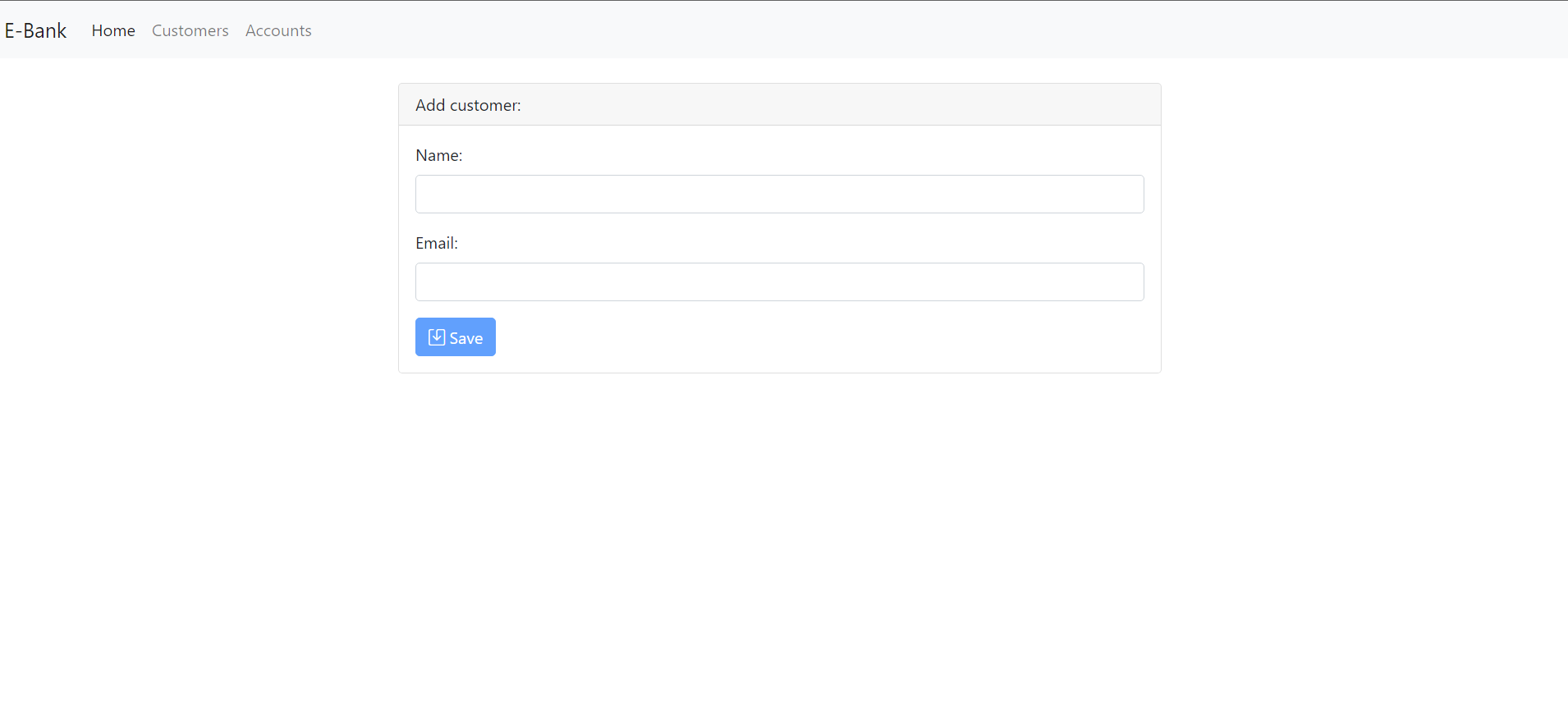
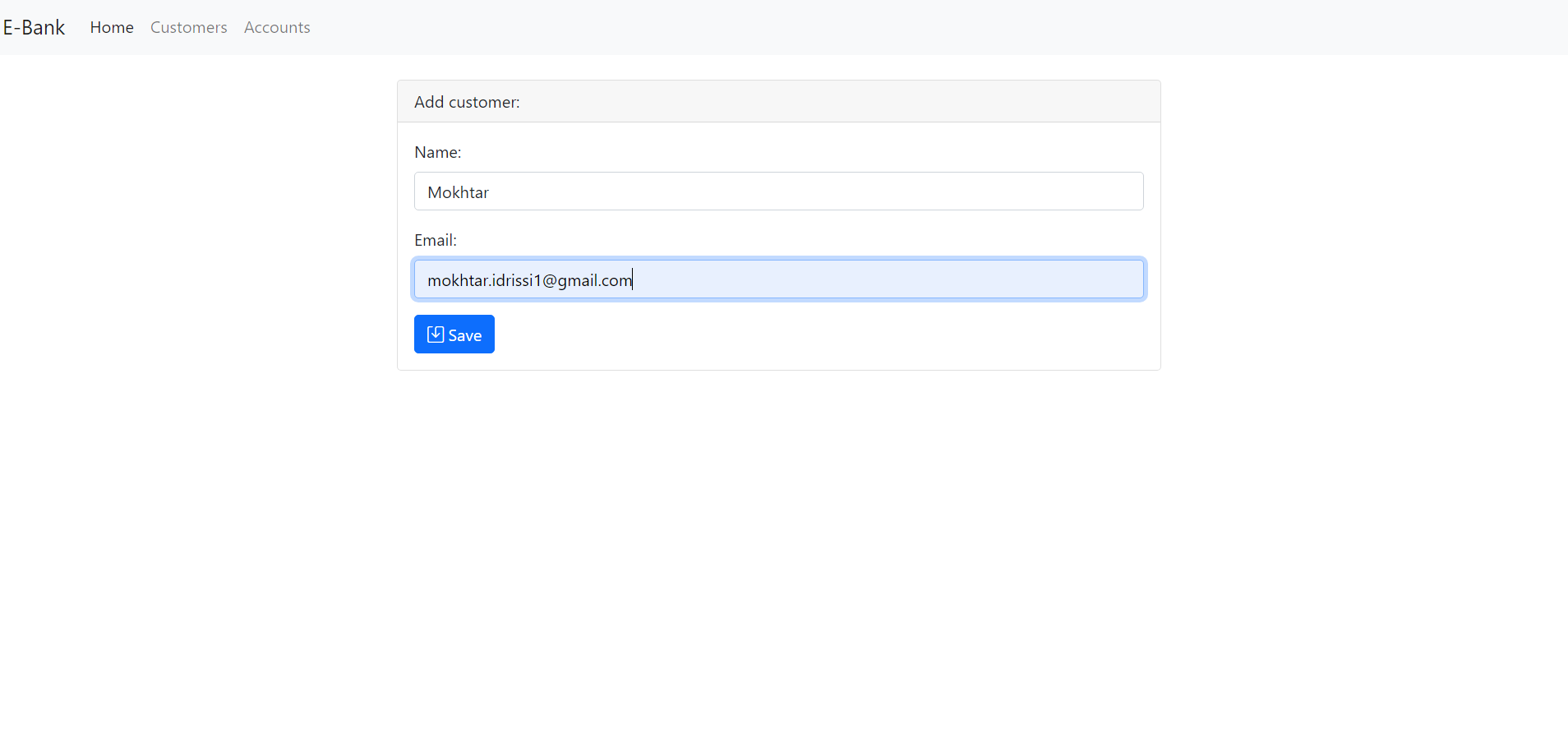
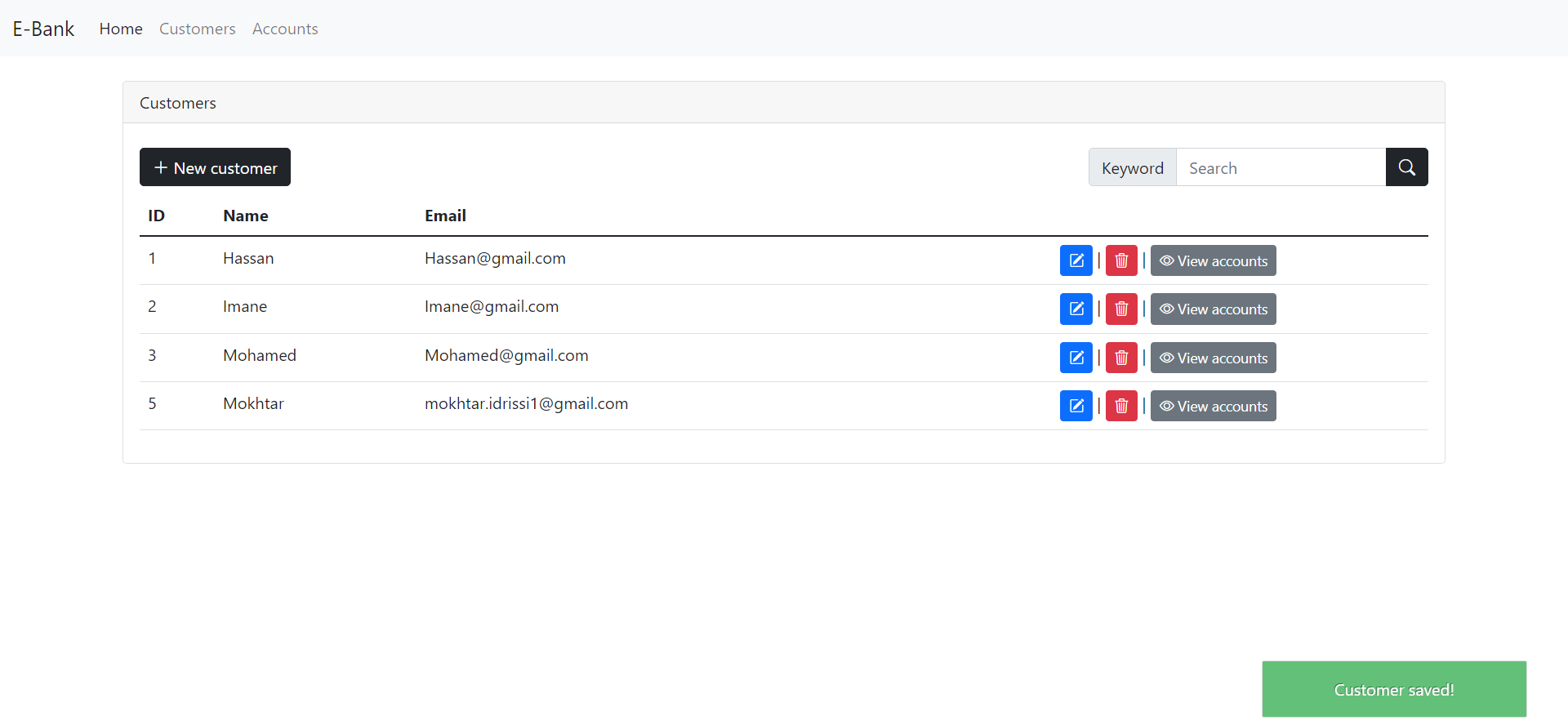
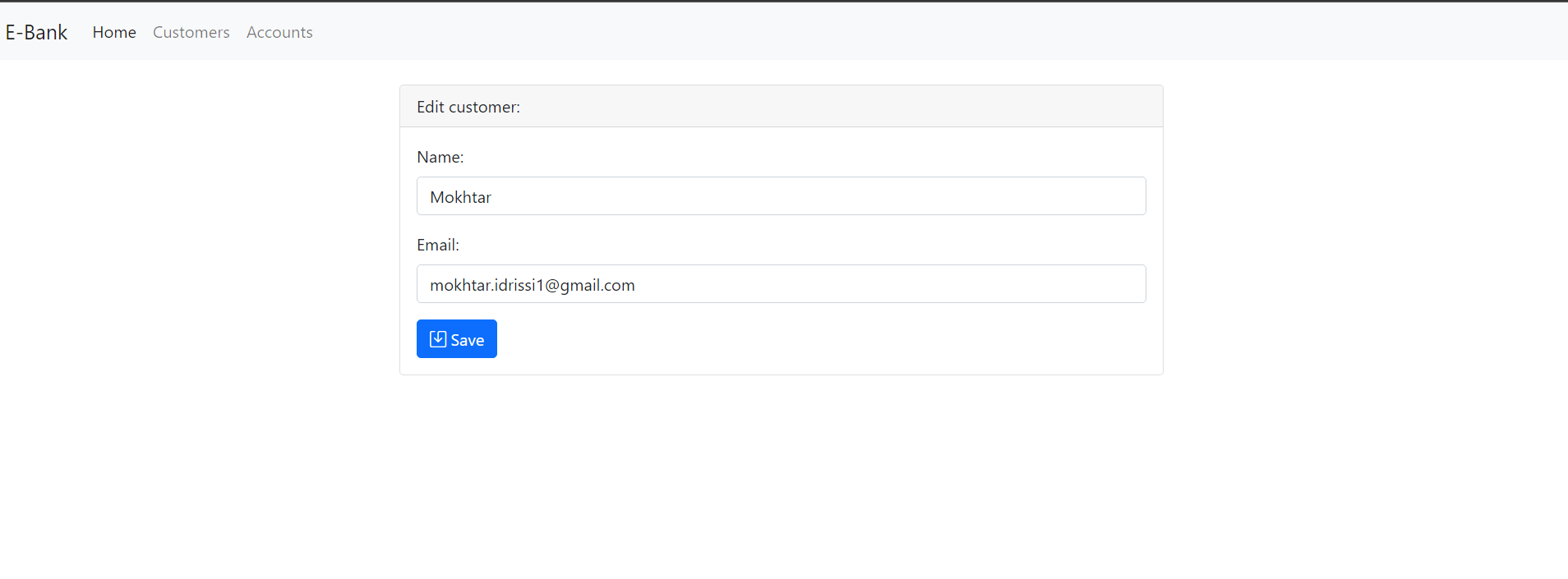
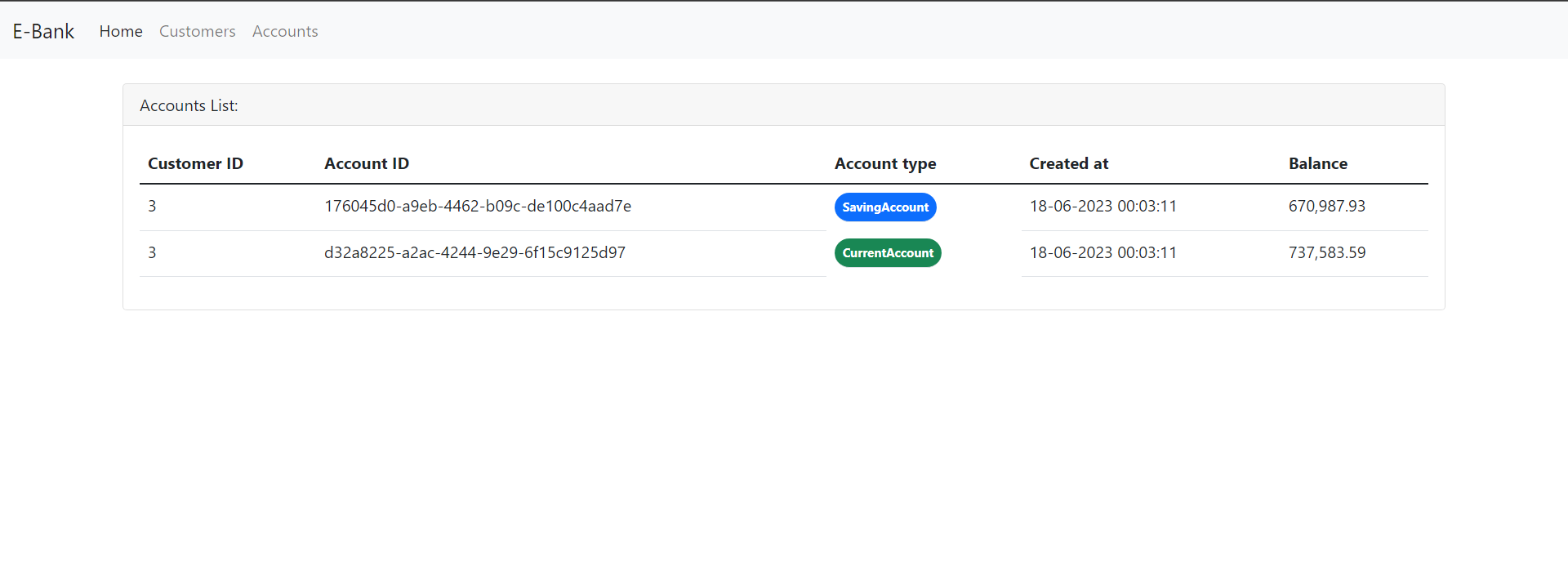
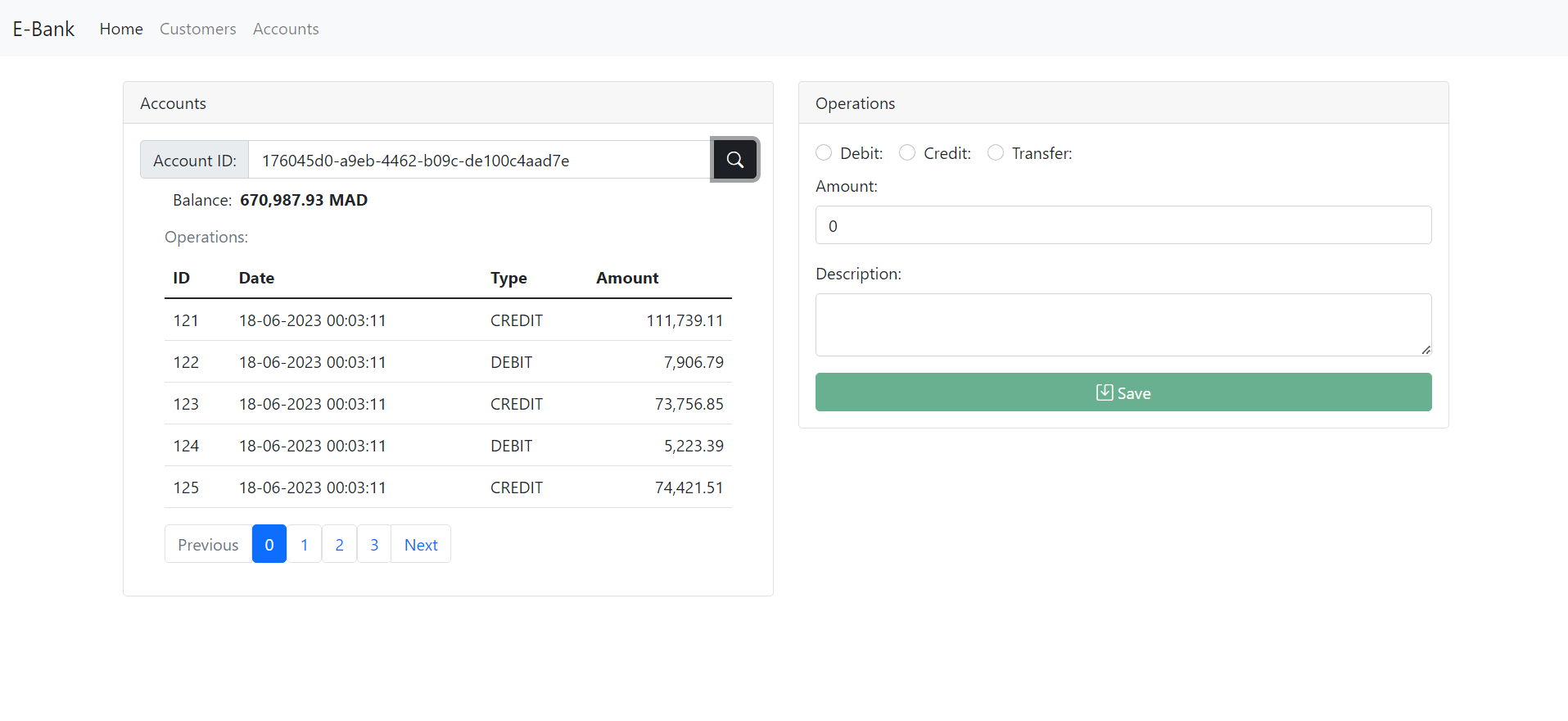
### Demonstration

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
  2. 

## 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.