

Ministère de l'Enseignement Supérieur et de la Recherche
Scientifique

Ecole Polytechnique Internationale Privée de Tunis



Projet de fin d'année : JEE

Ingénieur en INFORMATIQUE, RÉSEAUX ET MULTIMÉDIA

Java JEE

Présenté et réalisé par :

ALOUÏ Houcem

HAMMOUDA Ons

Application de Gestion de Comptes Bancaires

Année Universitaire 2024 - 2025

Remerciements

Nous tenons à exprimer notre sincère gratitude à toutes les personnes et institutions qui ont contribué à la réussite de notre projet de fin d'année intitulé « Gestion de Comptes Bancaires ».

En premier lieu, nous remercions chaleureusement notre encadrant pour son accompagnement constant, ses conseils précieux et la qualité de ses enseignements. Son expertise, sa disponibilité et sa capacité à nous guider ont été des atouts essentiels pour surmonter les défis techniques et méthodologiques tout au long de ce projet.

Nous souhaitons également souligner l'importance de notre collaboration en binôme, qui a été déterminante pour la réussite de ce projet. Cette coopération nous a permis de combiner nos compétences respectives, d'échanger nos idées et de renforcer notre esprit d'équipe. La complémentarité de nos efforts et notre communication efficace ont joué un rôle clé dans l'atteinte de nos objectifs. Nos remerciements s'adressent aussi à l'ensemble du corps enseignant de l'École Polytechnique Internationale Privée de Tunis pour la qualité de leur enseignement et leur soutien tout au long de notre parcours. Leur engagement a grandement contribué à notre formation et à l'acquisition des compétences nécessaires à la réalisation de ce projet.

Enfin, nous exprimons notre profonde reconnaissance à nos familles et nos proches pour leur soutien inébranlable, leur patience et leurs encouragements, qui ont été des moteurs précieux tout au long de cette expérience.

Table des matières

Introduction générale	1
1 Contexte général de projet	2
1.1 Introduction	3
1.2 Présentation du projet	3
1.2.1 Cadre du projet	3
1.2.2 Problématique	3
1.2.3 Solution proposée	3
1.3 Conclusion	4
2 Spécification des besoins et modélisation	5
2.1 Introduction	6
2.2 Définition des acteurs	6
2.3 Besoins fonctionnels	6
2.4 Méthodologie Scrum	8
2.5 Principes de la méthodologie Agile	8
2.6 Les rôles Scrum	9
2.6.1 Product Owner	9
2.6.2 Scrum Master	9
2.6.3 L'Équipe de Développement	9
2.7 Backlog Produit	9
2.8 Besoins non fonctionnels	11
2.9 Modélisation	12
2.9.1 Diagramme de classes	12
2.9.2 Diagramme de séquence	13
2.10 Conclusion	14
3 Réalisation	15
3.1 Introduction	16
3.2 Environnement matériel	16
3.3 Environnement logiciel	16
3.3.1 La partie locale	16

3.3.2	La partie Web	18
3.4	Architecture logicielle de l'application :	20
3.4.1	Architecture MVC :	21
3.4.2	Les trois couches du modèle MVC :	22
3.5	Conclusion	22
4	Mise en oeuvre	23
4.1	Introduction	24
4.2	Interface application web	24
4.2.1	Sign IN	24
4.2.2	Login	24
4.2.3	Page d'accueil	25
4.2.4	Gestion de client	26
4.2.5	Gestion de Compte	27
4.2.6	Gestion de Carte	29
4.2.7	Historique de transactions	29
4.3	Conclusion	30
	Conclusion générale	31

Table des figures

2.1	Diagramme de cas d'utilisation globale	7
2.2	Diagramme de classe	13
2.3	Diagramme de séquence	14
3.1	Spring Framework	17
3.2	Thymeleaf	17
3.3	Hibernate (JPA)	17
3.4	DBC (MyPHPadmin)	18
3.5	JSP (Java Server Pages)	18
3.6	HTML	18
3.7	Servlets	19
3.8	Architecture logicielle de l'application	20
3.9	Architecture MVC	21
4.1	login page	24
4.2	login page	25
4.3	le dashboard de l'admin	25
4.4	Dashboard de l'utilisateur (client)	26
4.5	Gestion de client	26
4.6	Ajout client	26
4.7	Ajout client dans la base	27
4.8	Gestion de compte	27
4.9	Ajout de compte	28
4.10	Ajout compte dans la base	28
4.11	Gestion de carte	29
4.12	Historique de transactions	29
4.13	Ajout d'une transaction	30

Liste des tableaux

2.2	Planification des Sprints	11
3.1	Spécifications de l'appareil 1	16
3.2	Spécifications de l'appareil 2	16

Introduction générale

Le secteur bancaire connaît une profonde mutation à l'ère du numérique. Face à des clients de plus en plus connectés et exigeants, les établissements financiers doivent proposer des services digitaux accessibles, sécurisés et performants. La gestion en ligne des comptes, des cartes, des opérations financières ou encore des crédits est devenue une nécessité stratégique, permettant à la fois d'améliorer l'expérience utilisateur et d'optimiser les processus internes.

L'application repose sur une architecture logicielle basée sur le modèle MVC (Modèle-Vue-Contrôleur), favorisant la séparation des responsabilités, la modularité et la maintenabilité du code. Le projet est mené selon la méthodologie agile Scrum, avec une organisation en sprints permettant un développement itératif, une meilleure réactivité aux besoins fonctionnels, et une collaboration continue entre les membres de l'équipe. Ce rapport présente l'ensemble des phases du projet, depuis l'analyse des besoins jusqu'à la réalisation finale, en détaillant la conception technique, le développement, les choix technologiques et les résultats obtenus. :

Le premier chapitre pose le cadre général du projet. Il présente la problématique abordée, analyse les solutions bancaires existantes, et détaille la méthodologie adoptée pour concevoir et développer notre application.

Le chapitre 2 est dédié à l'identification des besoins, tant fonctionnels que techniques, ainsi qu'à la modélisation des processus essentiels à la mise en œuvre de la solution.

Le chapitre 3 décrit l'environnement matériel et logiciel utilisé, en exposant les technologies retenues et les justifications de ces choix au regard des exigences du projet.

Le chapitre 4 offre une vue détaillée de l'application développée, en mettant l'accent sur l'interface utilisateur, les fonctionnalités principales et l'expérience offerte à l'utilisateur final.

Enfin, le rapport se termine par une synthèse des résultats obtenus, une analyse des perspectives d'évolution, ainsi qu'un retour critique sur les enseignements tirés tout au long de cette expérience projet.

CONTEXTE GÉNÉRAL DE PROJET

Plan

1	Introduction	3
2	Présentation du projet	3
3	Conclusion	4

1.1 Introduction

L'étude du projet constitue une étape cruciale pour définir le contexte et les objectifs principaux de notre démarche. Dans un premier temps, nous aborderons les enjeux de la gestion numérique des comptes bancaires, ainsi que la pertinence d'une telle solution. Nous présenterons ensuite le projet, en exposant la problématique à résoudre, une analyse des solutions existantes et la réponse apportée par notre solution pour répondre aux besoins identifiés.

1.2 Présentation du projet

1.2.1 Cadre du projet

Ce projet se concentre sur la conception et le développement d'une plateforme de gestion bancaire en ligne. Cette plateforme permettra aux utilisateurs de gérer leurs comptes de manière efficace, d'effectuer des transactions, de suivre leurs opérations financières et d'accéder à des tableaux de bord interactifs pour analyser leurs données en temps réel. La digitalisation croissante du secteur bancaire rend cette initiative d'autant plus pertinente, en réponse aux attentes toujours plus fortes en matière de simplicité et d'accessibilité des services.

1.2.2 Problématique

La gestion bancaire traditionnelle présente encore de nombreuses limites, telles qu'un manque d'automatisation, une accessibilité restreinte et une expérience utilisateur souvent peu optimisée. Les établissements bancaires doivent évoluer pour offrir des services numériques complets tout en garantissant la sécurité et la fiabilité des transactions. Le principal défi réside dans la création d'une solution qui permette une gestion efficace et sécurisée des comptes, tout en répondant aux besoins divers des utilisateurs (clients et gestionnaires) dans un environnement intuitif et performant.

1.2.3 Solution proposée

La solution proposée consiste à développer une application web basée sur le framework Spring MVC. Cette plateforme offrira plusieurs fonctionnalités clés :

Gestion des comptes : création, consultation et mise à jour des comptes bancaires.

Suivi des opérations : consultation de l'historique des transactions et gestion des opérations financières.

Tableaux de bord : visualisation des données financières à travers des graphiques interactifs

pour un suivi en temps réel.

Interface utilisateur conviviale : une expérience fluide et intuitive, adaptée aux besoins des utilisateurs.

La plateforme s'appuiera sur des technologies modernes telles que les Servlets, JSP, Spring MVC et JPA, garantissant ainsi une architecture robuste, évolutive et sécurisée. L'évaluation de la solution sera réalisée en tenant compte des critères de performance, d'ergonomie et de sécurité, afin d'assurer une application fiable et conforme aux attentes des utilisateurs.

1.3 Conclusion

En conclusion, ce projet vise à développer une solution innovante pour répondre aux besoins de gestion bancaire numérique. Grâce à la conception d'une application web performante et intuitive, nous facilitons l'accès à des services bancaires modernes, contribuant ainsi à l'amélioration des pratiques de gestion financière dans le cadre de la transformation numérique du secteur.

SPÉCIFICATION DES BESOINS ET MODÉLISATION

Plan

1	Introduction	6
2	Définition des acteurs	6
3	Besoins fonctionnels	6
4	Méthodologie Scrum	8
5	Principes de la méthodologie Agile	8
6	Les rôles Scrum	9
7	Backlog Produit	9
8	Besoins non fonctionnels	11
9	Modélisation	12
10	Conclusion	14

2.1 Introduction

Ce chapitre a pour objectif de formaliser les éléments préliminaires à la conception de notre application de gestion bancaire. Nous y présentons la structure fonctionnelle du système, les différents acteurs impliqués ainsi que les interactions qu'ils entretiennent avec la plateforme. La modélisation permet d'obtenir une vision claire et structurée de l'architecture du projet, facilitant ainsi les étapes ultérieures de développement.

2.2 Définition des acteurs

Dans une application de gestion bancaire, les acteurs représentent les entités qui interagissent avec le système pour réaliser des opérations spécifiques. Ces acteurs peuvent être humains ou externes. Dans le cadre de GESTBANCAIRE, nous avons identifié deux catégories principales d'acteurs :

Le client bancaire (USER) : Il s'agit de l'utilisateur final de la plateforme, qui utilise l'application pour gérer ses comptes bancaires, consulter ses soldes, effectuer des virements, visualiser l'historique de ses opérations, et recevoir des alertes ou des notifications.

L'administrateur du système (ADMIN) : C'est l'utilisateur ayant des privilèges étendus, chargé de la gestion globale du système. Il peut créer ou supprimer des comptes, superviser les opérations des clients, générer des rapports financiers, et assurer la sécurité et la maintenance des données.

Le diagramme de cas d'utilisation global (Figure 2.1) présente de manière synthétique l'ensemble des interactions possibles entre ces acteurs et la plateforme GESTBANCAIRE.

2.3 Besoins fonctionnels

Les besoins fonctionnels représentent les fonctionnalités principales que le système doit offrir. Notre application web GESTBANCAIRE propose les services suivants :

Gestion des comptes bancaires

- Consultation des comptes avec détails : solde, historique, informations générales.
- Création, modification et suppression de comptes par l'administrateur.
- Visualisation d'un tableau de bord synthétique avec indicateurs clés.

Transactions bancaires

- Offrir aux clients la possibilité d'effectuer des virements entre leurs propres comptes ou vers des comptes tiers enregistrés.

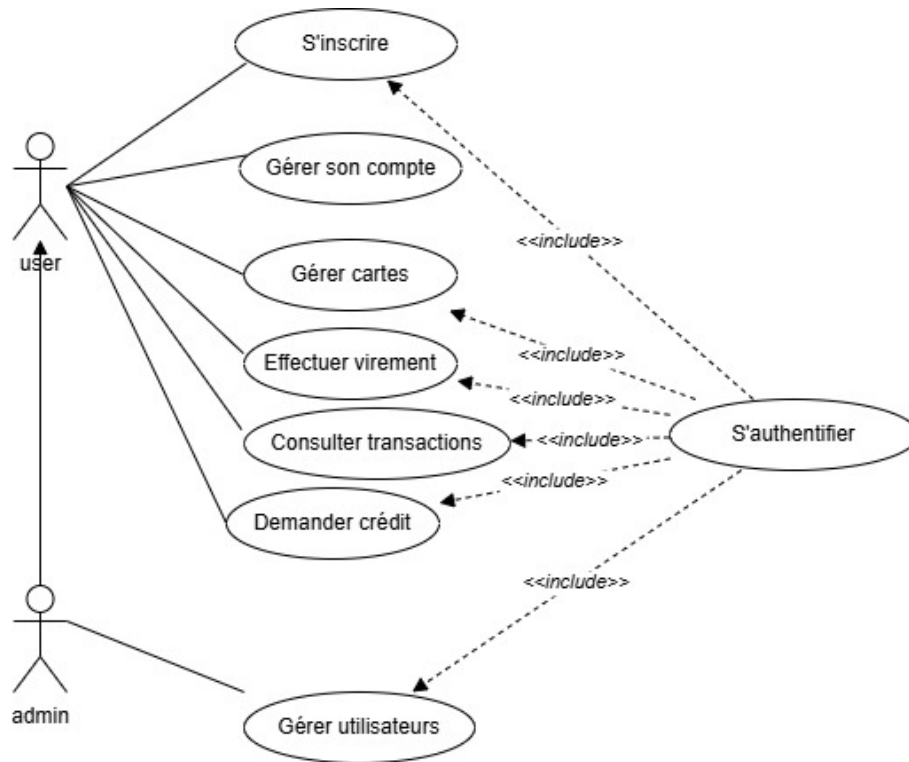


FIGURE 2.1 : Diagramme de cas d'utilisation globale

- Mettre en place une fonctionnalité de paiement de factures (eau, électricité, internet...) via l'application.
- Implémenter un système de notifications (email ou in-app) à chaque transaction effectuée pour garantir la transparence.
- Afficher l'historique des opérations financières avec des filtres par date, type d'opération, montant, ou bénéficiaire.

Sécurité et Gestion des utilisateurs

- Authentification sécurisée avec identifiant et mot de passe, et option d'authentification à deux facteurs pour renforcer la sécurité.
- Gestion complète des utilisateurs par l'administrateur : création de profils, modification d'informations, suppression de comptes.
- Mise en place d'un système de rôles pour différencier les droits d'accès entre clients et administrateurs.

Visualisation et rapports financiers

- Intégration d'un tableau de bord interactif affichant les indicateurs financiers : solde global, dépenses mensuelles, revenus, etc.
- Génération automatique de rapports financiers périodiques : hebdomadaires, mensuels ou annuels.

- Affichage de graphiques comparatifs pour aider à l'analyse des flux financiers (par catégorie, période, ou compte).

Sauvegarde et Récupération de données

- Sauvegarde automatique et régulière des données critiques (transactions, comptes, utilisateurs).
- Mise en œuvre d'un mécanisme de restauration des données pour pallier toute éventuelle perte suite à une panne ou à un incident technique.

Notification et alertes

- Envoi de notifications automatiques pour informer l'utilisateur d'événements importants (transaction en attente, paiement imminent, seuil de solde atteint, etc.).
- Alertes de sécurité en cas de tentative de connexion suspecte ou d'activité inhabituelle sur le compte.

2.4 Méthodologie Scrum

La méthodologie Scrum est un cadre de travail agile utilisé pour la gestion de projet. Il repose sur des itérations courtes et régulières appelées "sprints", permettant ainsi une livraison incrémentale des fonctionnalités du produit. Scrum favorise la collaboration étroite entre les membres de l'équipe et les parties prenantes, tout en permettant une grande flexibilité face aux changements.

Dans le cadre de ce projet, Scrum a été appliqué pour organiser le travail de développement en plusieurs sprints. Chaque sprint a une durée de deux semaines, avec des objectifs clairement définis à atteindre à la fin de chaque itération.

2.5 Principes de la méthodologie Agile

La méthodologie agile repose sur plusieurs principes clés qui visent à améliorer la flexibilité, la collaboration et la réactivité au changement dans le développement logiciel. Voici les principaux principes de la méthodologie agile :

- **Livraison incrémentale et itérative** : Le travail est découpé en petites itérations (sprints), permettant ainsi des livraisons régulières et un retour rapide sur l'avancement du projet.
- **Collaboration avec les parties prenantes** : Les développeurs et les parties prenantes (clients, utilisateurs) collaborent étroitement tout au long du projet.
- **Adaptation au changement** : La méthodologie agile permet d'adapter le projet en fonction des retours des parties prenantes et des progrès réalisés pendant les sprints, garantissant que les

besoins du client sont toujours au cœur du développement.

- **Autonomie des équipes :** Les équipes Scrum sont auto-organisées et responsables de la gestion du travail, ce qui permet une plus grande réactivité.
- **Simplicité et efficacité :** L'accent est mis sur la simplicité du code et la recherche de solutions efficaces.

2.6 Les rôles Scrum

Scrum définit trois rôles principaux qui sont essentiels au bon déroulement du projet :

2.6.1 Product Owner

Le Product Owner (PO) est responsable de la définition des priorités du produit et de la gestion du backlog produit. Ce rôle est crucial pour s'assurer que le travail effectué correspond aux attentes des parties prenantes et aux objectifs du projet. Le PO est le garant de la vision du produit et s'assure que les fonctionnalités développées répondent aux besoins du client.

2.6.2 Scrum Master

Le Scrum Master est responsable de la bonne application de Scrum au sein de l'équipe. Il veille à ce que l'équipe suive les principes et les pratiques Scrum, élimine les obstacles qui pourraient nuire à la productivité, et protège l'équipe des distractions extérieures. Le Scrum Master est également un facilitateur des réunions Scrum et aide à maintenir une communication fluide au sein de l'équipe.

2.6.3 L'Équipe de Développement

L'équipe de développement est composée de professionnels ayant les compétences nécessaires pour réaliser le travail du projet. Cette équipe est auto-organisée et collabore pour atteindre les objectifs fixés. L'équipe est multidisciplinaire et chaque membre est responsable de son domaine d'expertise tout en travaillant ensemble pour atteindre l'objectif du sprint.

2.7 Backlog Produit

Le backlog produit est une liste priorisée de toutes les fonctionnalités, améliorations et corrections à apporter au produit. Les user stories sont classées par ordre de priorité et assignées aux différents sprints. Ce backlog est constamment mis à jour en fonction des retours des parties prenantes et des progrès réalisés pendant les sprints.

ID	User Story	Priorité	Sprint
US01	En tant qu'utilisateur, je veux pouvoir m'enregistrer avec un rôle défini	Haute	Sprint 1
US02	En tant qu'utilisateur, je veux pouvoir me connecter et obtenir un token JWT	Haute	Sprint 1
US03	En tant qu'utilisateur, je veux que l'authentification sécurise mes requêtes	Haute	Sprint 1
US04	En tant qu'administrateur, je veux pouvoir gérer les rôles des utilisateurs	Moyenne	Sprint 1
US05	En tant que développeur, je veux permettre les requêtes front grâce au CORS	Moyenne	Sprint 1
US06	En tant qu'administrateur, je veux pouvoir consulter tous les utilisateurs	Moyenne	Sprint 2
US07	En tant qu'utilisateur, je veux créer un compte bancaire	Haute	Sprint 2
US08	En tant qu'utilisateur, je veux consulter la liste de mes comptes bancaires	Haute	Sprint 2
US09	En tant qu'utilisateur, je veux supprimer un de mes comptes bancaires	Moyenne	Sprint 2
US10	En tant qu'utilisateur, je veux créer une carte bancaire liée à un compte	Haute	Sprint 3
US11	En tant qu'utilisateur, je veux visualiser mes cartes bancaires	Moyenne	Sprint 3
US12	En tant qu'utilisateur, je veux effectuer une transaction entre deux comptes bancaires	Haute	Sprint 3
US13	En tant qu'utilisateur, je veux consulter l'historique de mes transactions	Moyenne	Sprint 3
US14	En tant qu'utilisateur, je veux pouvoir demander un crédit	Haute	Sprint 4
US15	En tant qu'administrateur, je veux valider ou refuser un crédit	Moyenne	Sprint 4

TABLEAU 2.2 : Planification des Sprints

Sprint	Durée	Objectif
Sprint 1	2 semaines	Mettre en place l'infrastructure du projet, sécuriser l'accès via JWT, permettre l'inscription et la connexion des utilisateurs.
Sprint 2	2 semaines	Implémenter la gestion des utilisateurs côté administrateur et la création des comptes bancaires.
Sprint 3	2 semaines	Gérer les cartes bancaires et implémenter le système de transaction entre comptes.
Sprint 4	2 semaines	Ajouter la gestion des crédits bancaires : demande par l'utilisateur et validation par l'administrateur.

2.8 Besoins non fonctionnels

Outre les fonctionnalités principales, le système doit respecter un ensemble d'exigences non fonctionnelles essentielles au bon fonctionnement et à la qualité globale de l'application.

Performance

- Réactivité élevée même en cas de trafic important.
- Optimisation des requêtes et utilisation du cache pour les données fréquentes.
- Base de données performante pour un accès rapide aux informations.

Scalabilité

- L'architecture de l'application doit permettre l'ajout progressif de nouvelles fonctionnalités (gestion de crédits, prêts, placements. ...).
- Il doit être possible d'augmenter la capacité du serveur et de la base de données pour supporter une croissance du nombre d'utilisateurs.
- La plateforme doit être conçue pour s'adapter à l'évolution des besoins sans devoir être reconstruite intégralement.

Portabilité

- L'application doit pouvoir être déployée sur divers environnements : serveurs locaux, cloud, conteneurs (Docker).
- L'utilisation de fichiers de configuration paramétrables permettra une adaptation facile aux différents contextes (développement, test, production).

Facilité de Maintenance

- Le code doit être structuré en modules indépendants et réutilisables afin de faciliter les évolutions et la correction des bugs.
- L'utilisation des logs détaillés doit être mis en place pour suivre les actions critiques et identifier rapidement les anomalies.
- Une documentation claire, technique et fonctionnelle, doit accompagner le projet pour faciliter sa reprise ou son extension par d'autres développeurs.

Ergonomie et accessibilité

- L'interface utilisateur doit être intuitive, claire et adaptée à tous les profils (clients et administrateurs).
- La navigation entre les différentes fonctionnalités doit être fluide, avec des menus bien organisés et une hiérarchisation logique des informations.
- L'application doit être responsive et accessible sur tous types de supports : ordinateurs, tablettes, et smartphones.

2.9 Modélisation

2.9.1 Diagramme de classes

La figure 2.3 suivante présente le diagramme de classe, qui représente la structure du système en illustrant les classes, leurs attributs, méthodes et les relations entre elles.

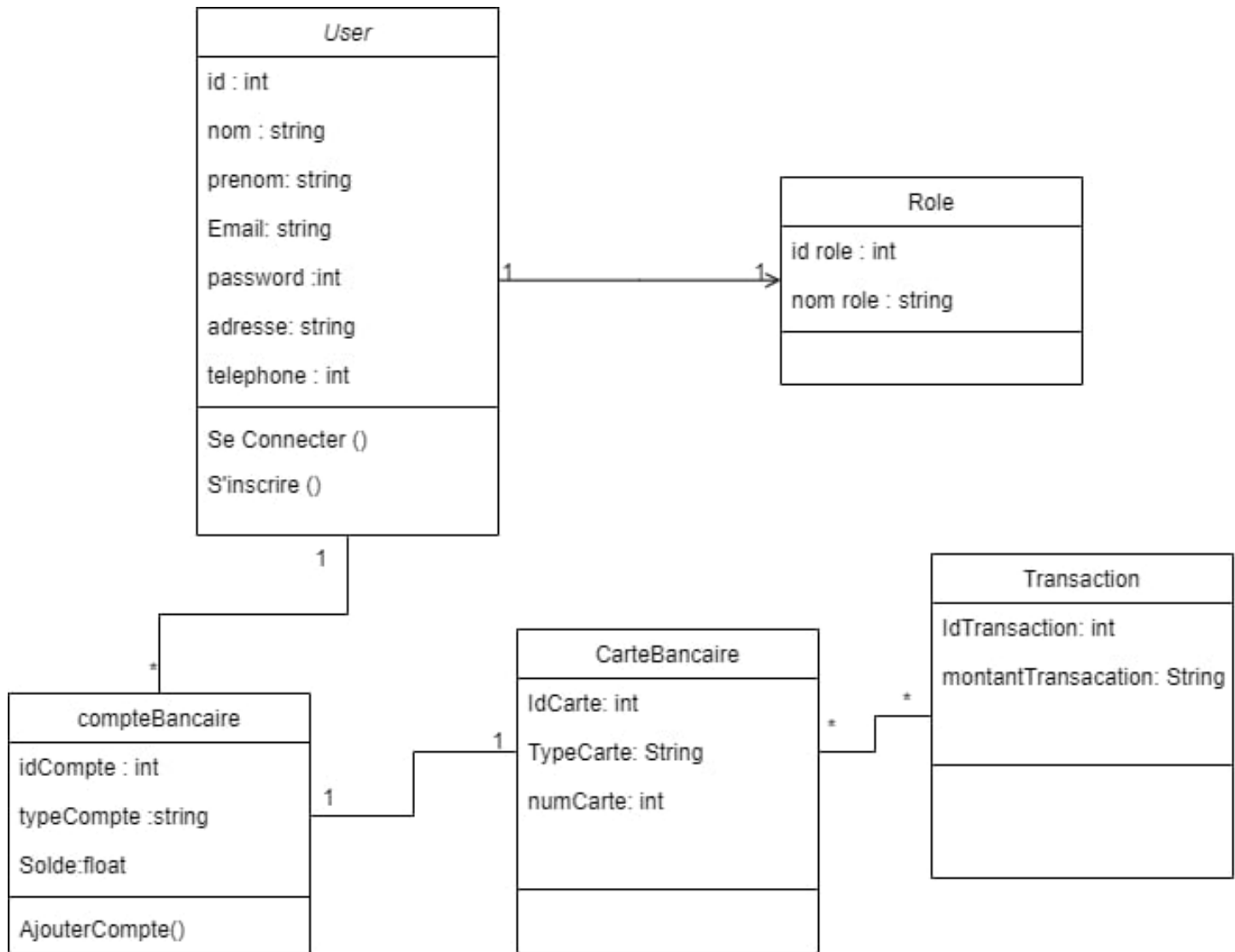
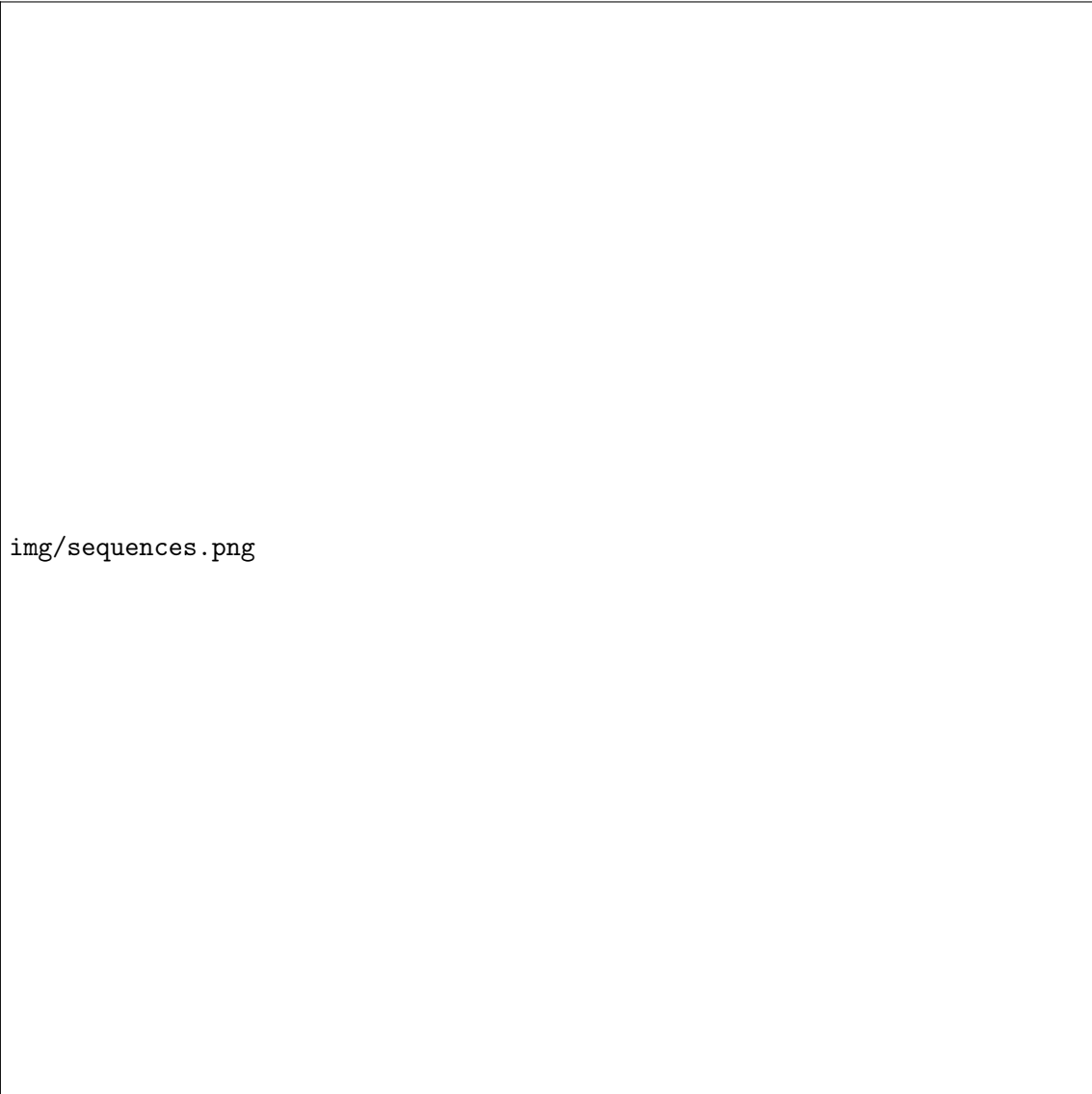


FIGURE 2.2 : Diagramme de classe

2.9.2 Diagramme de séquence

La figure 2.4 suivante illustre le diagramme de séquence, qui détaille les interactions dynamiques entre les objets du système au fil du temps, en mettant en évidence l'ordre des messages échangés pour réaliser une fonctionnalité spécifique.



img/sequences.png

FIGURE 2.3 : Diagramme de séquence

2.10 Conclusion

Dans ce chapitre, nous avons défini en détail les besoins fonctionnels et non fonctionnels de notre application de gestion bancaire. Ces spécifications nous ont permis de structurer clairement les fonctionnalités attendues par les utilisateurs, ainsi que les exigences techniques à respecter pour garantir performance, sécurité, maintenabilité et évolutivité. Cette étape constitue un socle essentiel pour la phase de modélisation et de conception technique, que nous aborderons dans le chapitre suivant à travers les diagrammes UML illustrant l'architecture du système.

RÉALISATION

Plan

1	Introduction	16
2	Environnement matériel	16
3	Environnement logiciel	16
4	Architecture logicielle de l'application :	20
5	Conclusion	22

3.1 Introduction

Ce chapitre expose les principales étapes de la mise en œuvre du projet, en détaillant l’environnement matériel utilisé ainsi que les solutions logicielles choisies pour le développement et le déploiement de notre application de vision par ordinateur.

3.2 Environnement matériel

L’environnement matériel est détaillée dans le tableau

TABLEAU 3.1 : Spécifications de l’appareil 1

Caractéristiques	Détails
Propriétaire	Hammouda Ons
Processeur	AMD Ryzen 5 5600H avec Radeon Graphics
Mémoire RAM installée	16,0 Go
Type du système	Système d’exploitation 64 bits, processeur x64

TABLEAU 3.2 : Spécifications de l’appareil 2

Caractéristiques	Détails
Nom de l’appareil	Aloui Houcem
Processeur	13th Gen Intel(R) Core(TM) i7-13700H 2.40 GHz
Mémoire RAM installée	32 Go
Type du système	Système d’exploitation 64 bits, processeur x64

3.3 Environnement logiciel

3.3.1 La partie locale

Dans la partie locale du projet, nous avons choisi d’utiliser des technologies issues de l’écosystème Java pour développer les fonctionnalités principales de l’application de gestion bancaire. Ces outils ont été essentiels pour garantir une application stable, évolutive et performante.

Spring Framework : Ce framework a été adopté pour orchestrer la logique métier de l’application et assurer une intégration fluide des différents composants. Grâce à Spring, nous avons pu mettre en œuvre le modèle architectural MVC (Model-View-Controller), ce qui a facilité la séparation des couches métier, présentation et traitement, tout en améliorant la maintenabilité et la clarté du code.



FIGURE 3.1 : Spring Framework

Thymeleaf : Thymeleaf est un moteur de templates pour Java, compatible avec les environnements web (servlets) ainsi que les environnements non web. Il est particulièrement bien adapté pour générer du contenu XHTML/HTML5 dans la couche de présentation des applications web basées sur le modèle MVC. Toutefois, il est également capable de traiter tout type de fichier XML, y compris en mode hors ligne.



FIGURE 3.2 : Thymeleaf

Hibernate (JPA) : Hibernate a été utilisé pour assurer le mapping objet-relationnel (ORM), facilitant ainsi la gestion des échanges avec la base de données. Cet outil nous a permis de manipuler les données sous forme d'objets Java tout en garantissant leur enregistrement et leur persistance dans la base de données de manière transparente.



FIGURE 3.3 : Hibernate (JPA)

MyPHPAdmin (Java Database Connectivity) : Cette technologie a permis la connexion directe à la base de données pour des opérations spécifiques ne nécessitant pas d'abstraction ORM. Nous l'avons utilisée principalement pour des requêtes optimisées.



FIGURE 3.4 : DBC (MyPHPAdmin)

3.3.2 La partie Web

a partie web de ce projet a été développée pour fournir une interface utilisateur intuitive et responsive. Elle permet aux utilisateurs d'interagir avec les fonctionnalités bancaires via un navigateur.

JSP (Java Server Pages) : JSP a été utilisé pour générer des pages dynamiques en récupérant les données fournies par les servlets. Grâce à JSTL (JavaServer Pages Tag Library), nous avons simplifié l'intégration de la logique métier dans les vues.



FIGURE 3.5 : JSP (Java Server Pages)

HTML : Langage de balisage utilisé pour structurer les pages web de l'interface utilisateur. HTML a permis de définir les sections de l'interface, comme le champ de téléversement d'images et la zone d'affichage des résultats.



FIGURE 3.6 : HTML

Servlets : les servlets ont joué un rôle central dans le traitement des requêtes HTTP, en transmettant les données entre la couche de présentation et la couche de service.



FIGURE 3.7 : Servlets

3.4 Architecture logicielle de l'application :

fin de représenter de manière symbolique et schématique les différents composants de notre système, ainsi que leurs interrelations et interactions, nous avons adopté une architecture garantissant la stabilité et l'efficacité de notre application. L'illustration de l'architecture de l'application est présentée dans la figure 3.1.

Architecture d'une application Web JEE

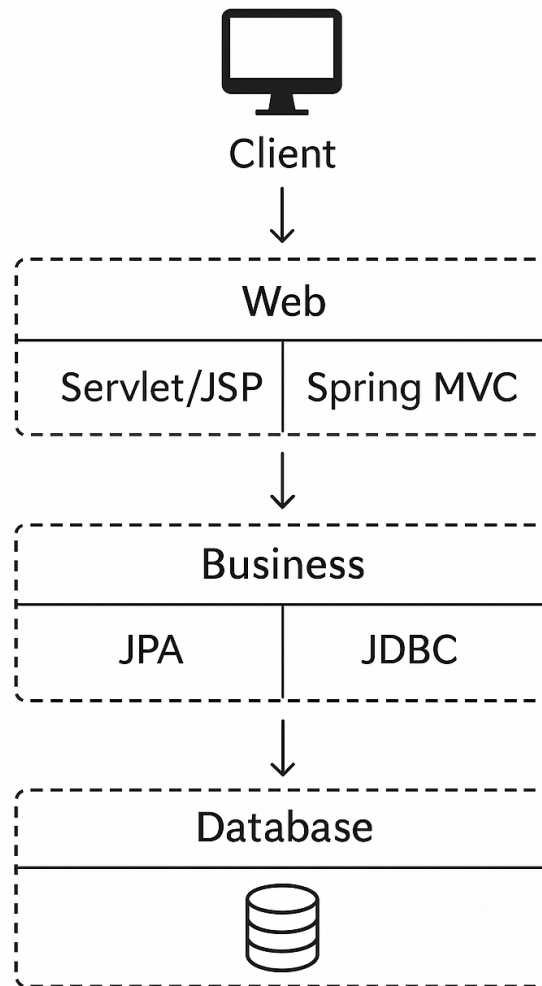


FIGURE 3.8 : Architecture logicielle de l'application

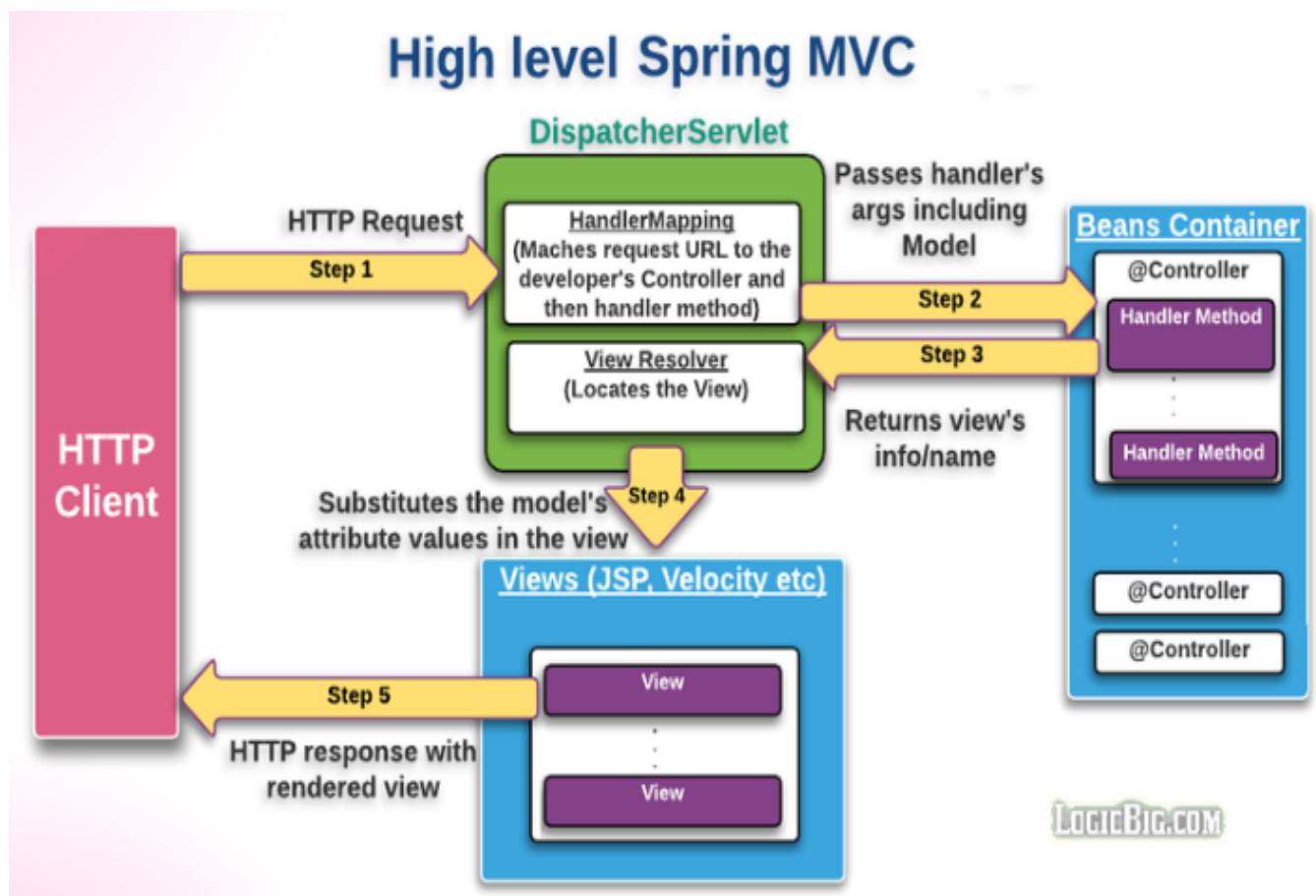
L'architecture logicielle d'une application client-serveur web en mode 3-tiers repose sur une division des fonctions de l'application en trois composants distincts : le client, le serveur d'application et le serveur de base de données. Dans ce modèle, le client envoie une requête au serveur d'application via le réseau internet, où elle est traitée par l'application. Le serveur web IIS interagit avec la base de données et le stockage des fichiers en utilisant le protocole standard HTTP. Une fois la réponse générée, elle est renvoyée du serveur de la base de données au serveur d'application, qui la transmet ensuite au client.

3.4.1 Architecture

MVC

:

Nous avons choisi d'adopter le modèle MVC, qui s'avère particulièrement adapté pour gérer l'interaction entre les différents composants de notre application web. L'architecture MVC (Modèle, Vue, Contrôleur) est une architecture à trois couches utilisée dans la programmation client/serveur et pour les interfaces graphiques. Ce modèle est très puissant grâce à son principe fondamental, qui consiste à séparer les données (modèle), l'affichage (vue) et les actions (contrôleur). Cette séparation permet une gestion claire et efficace des différentes fonctions de l'application. L'interaction entre ces trois couches est illustrée dans la figure ci-dessous :



[22]

FIGURE 3.9 : Architecture MVC

3.4.2 Les trois couches du modèle MVC :

- **Modèle** : Il correspond aux données stockées généralement dans une base de données. Dans un langage orienté objet, ces données sont exploitées sous forme de classes. Le modèle peut aussi agir sur la vue en mettant à jour ses données.
- **Vue** : Ne contenant que les informations liées à l’affichage, la vue se contente d’afficher le contenu qu’elle reçoit sans avoir connaissance des données. C’est l’interface homme-machine de l’application.
- **Contrôleur** : Le contrôleur sert de base à récupérer les informations, de les traiter en fonction des paramètres demandés par la vue (par l’utilisateur) puis de renvoyer à la vue les données afin d’être affichées. C’est l’élément qui va utiliser les données pour les envoyer à la vue.

3.5 Conclusion

Ce projet de gestion bancaire a permis d’implémenter une architecture basée sur le modèle MVC, assurant ainsi une séparation claire des responsabilités entre les différentes couches (modèle, vue et contrôleur). L’utilisation de technologies modernes telles que Spring Framework, Hibernate et JSP/Servlets a permis de concevoir une application robuste, facilement maintenable et évolutive. La partie web offre une interface utilisateur intuitive et responsive, grâce à l’intégration de Bootstrap et d’outils de visualisation comme Chart.js, facilitant ainsi la gestion des comptes et des opérations bancaires. Ce projet a également été une occasion d’acquérir une expérience pratique approfondie dans le développement d’applications Java EE, tout en explorant les meilleures pratiques en matière de développement logiciel.

MISE EN OEUVRE

Plan

1	Introduction	24
2	Interface application web	24
3	Conclusion	30

4.1 Introduction

Ce chapitre présente la mise en oeuvre de notre application web, axée sur une interface utilisateur intuitive. Nous aborderons les principales fonctionnalités, telles que la page d'accueil et l'insertion d'images, tout en mettant en avant l'équipe qui a contribué au développement de cette application.

4.2 Interface application web

4.2.1 Sign IN

La figure 4.2 suivante présente page de Sign IN.

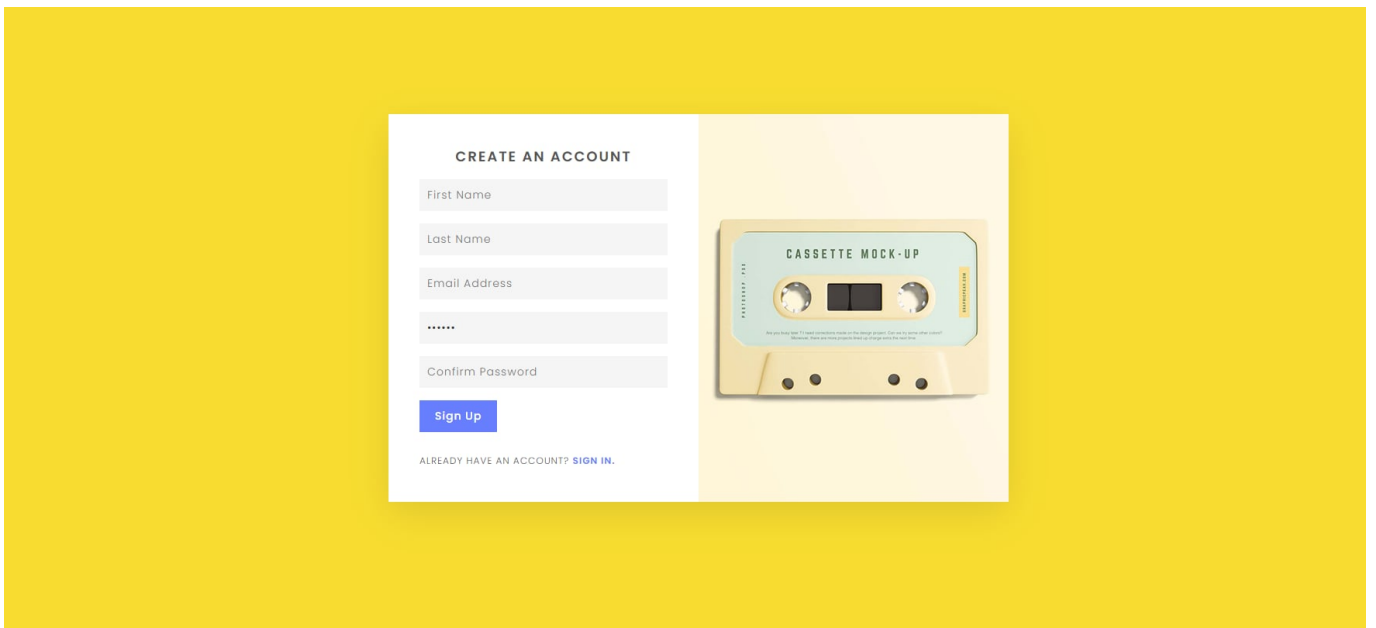


FIGURE 4.1 : login page

4.2.2 Login

La figure 4.2 suivante présente page de login.

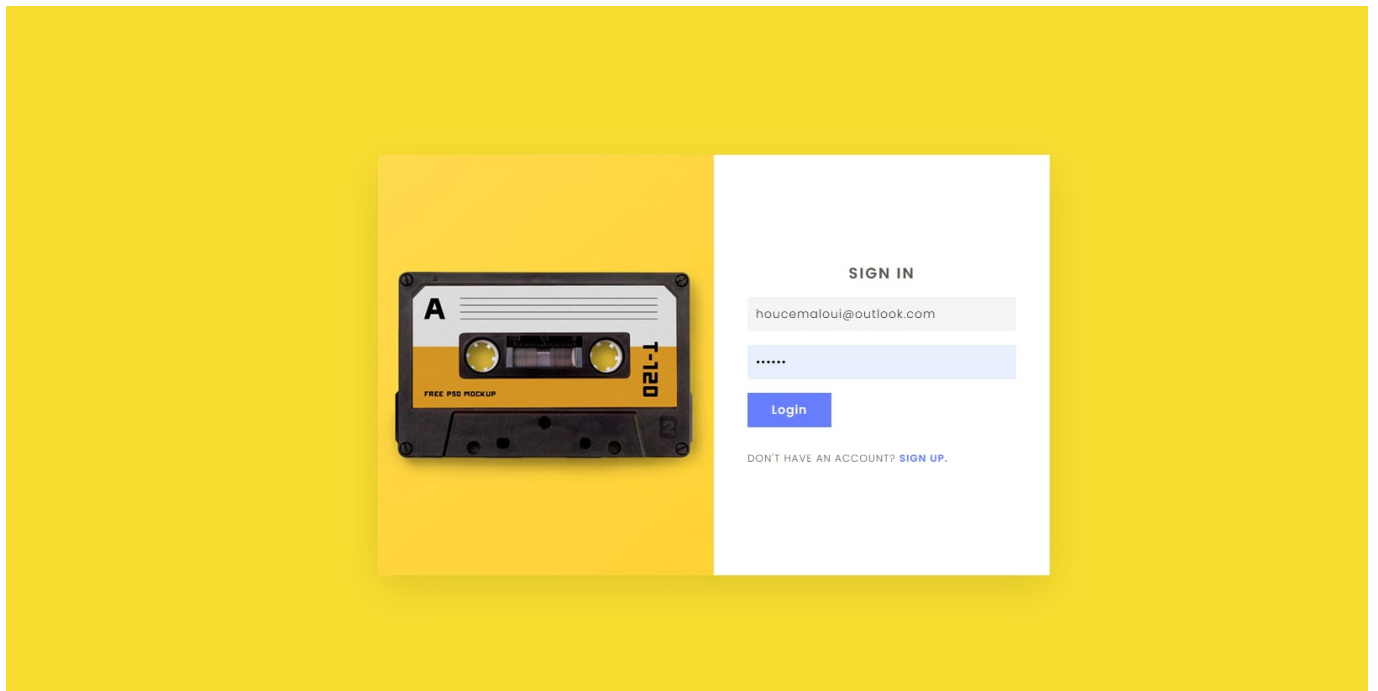


FIGURE 4.2 : login page

4.2.3 Page d'accueil

Les figures 4.1 et 4.2 suivante montrent la capture d'écran de la page d'accueil de l'application, offrant un aperçu de son interface utilisateur et de ses principales fonctionnalités.

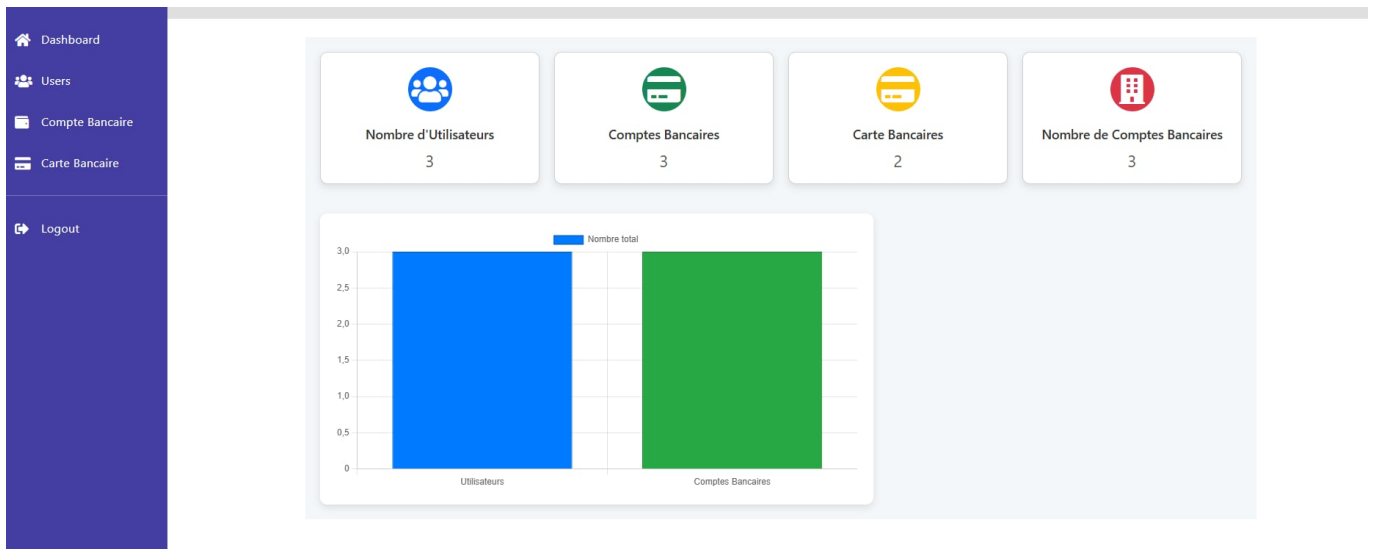


FIGURE 4.3 : le dashboard de l'admin

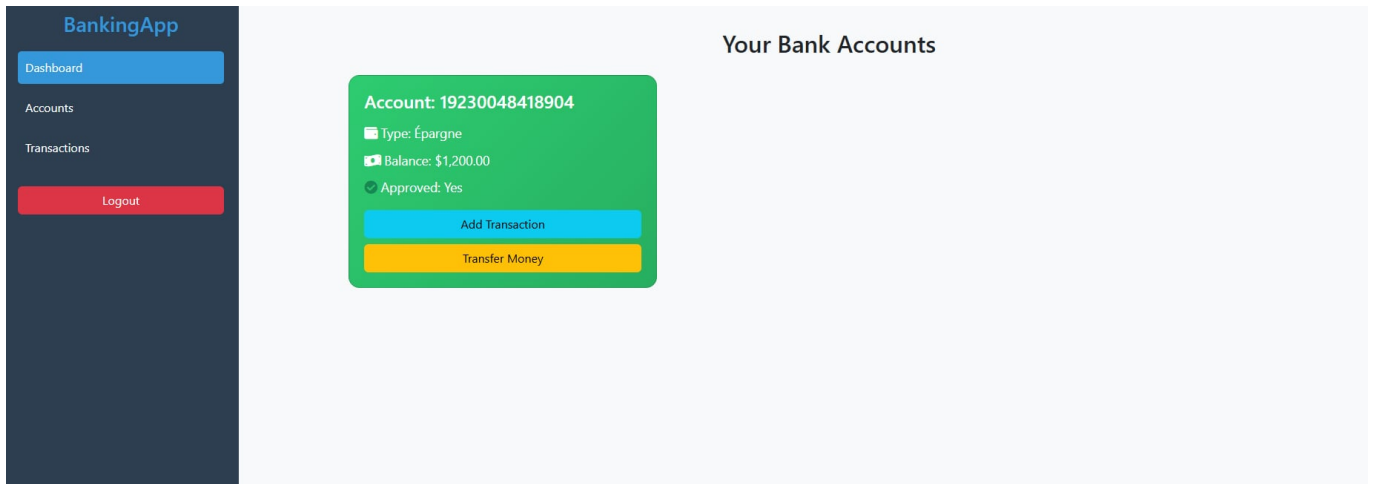


FIGURE 4.4 : Dashboard de l'utilisateur (client)

4.2.4 Gestion de client

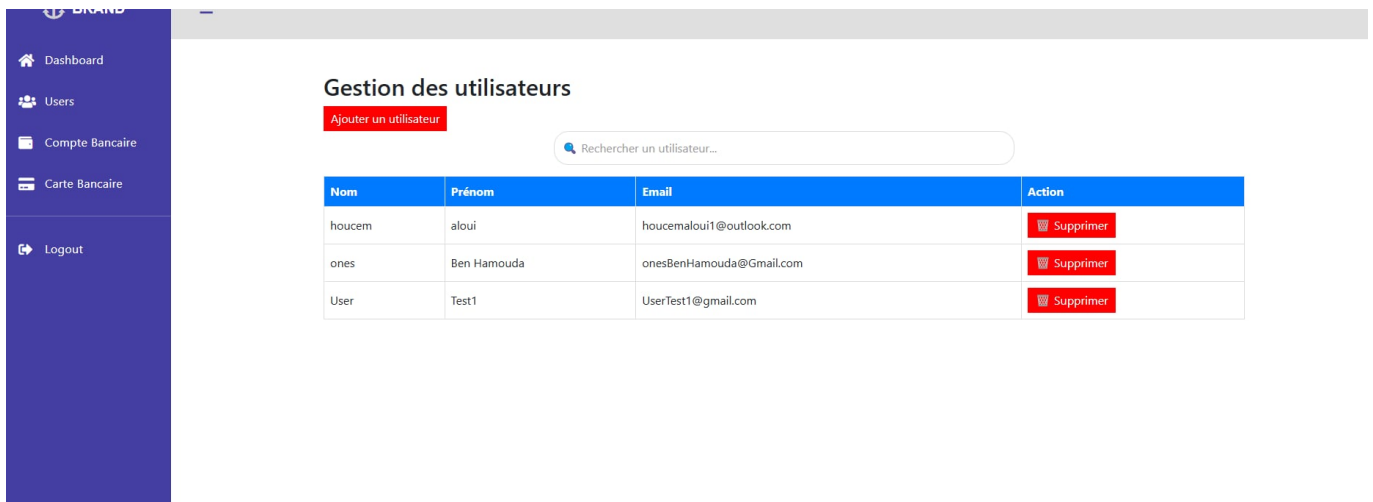


FIGURE 4.5 : Gestion de client

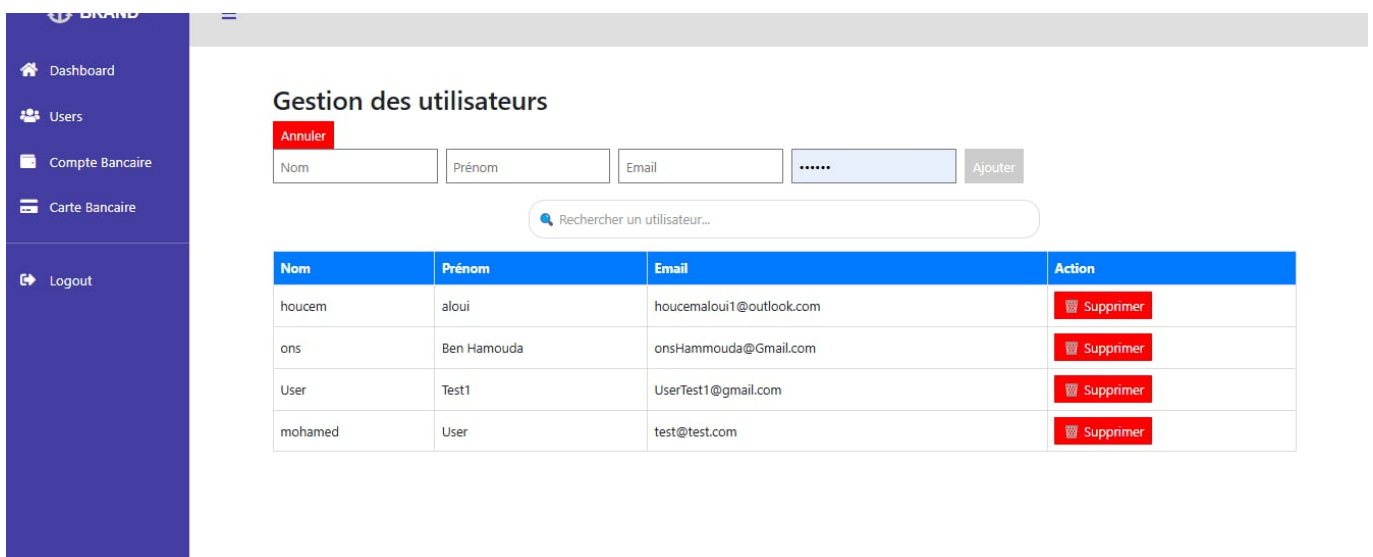


FIGURE 4.6 : Ajout client

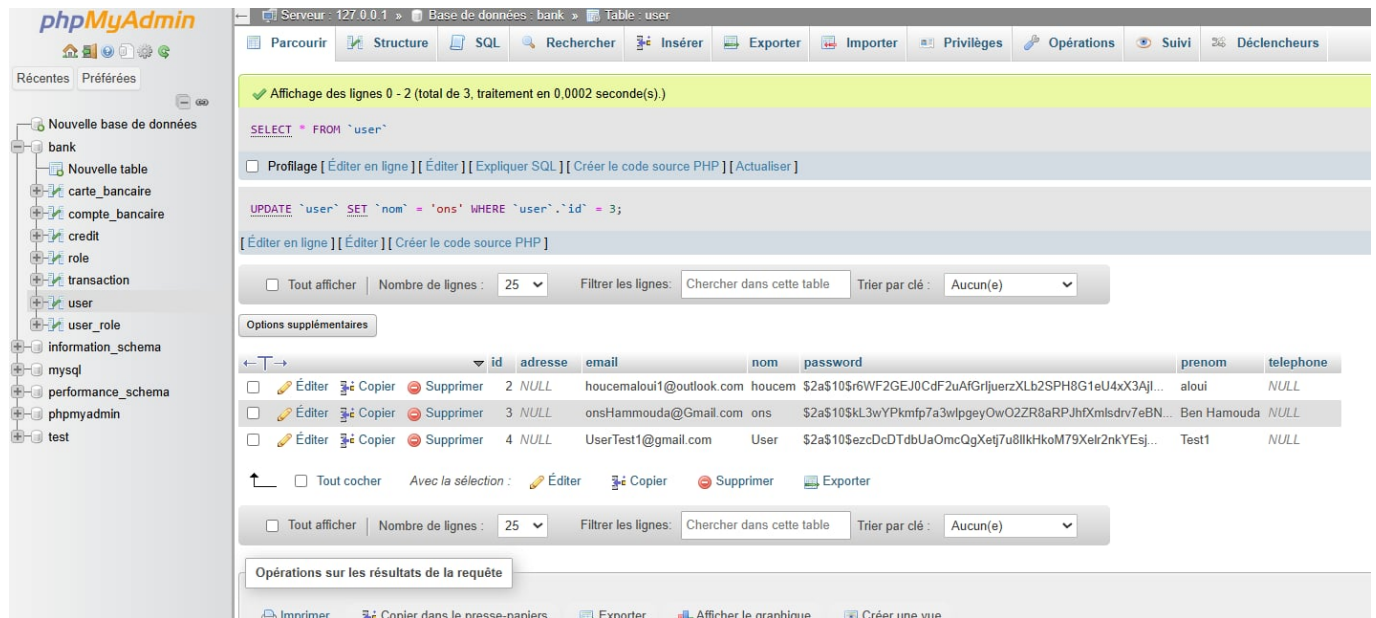


FIGURE 4.7 : Ajout client dans la base

4.2.5 Gestion de Compte

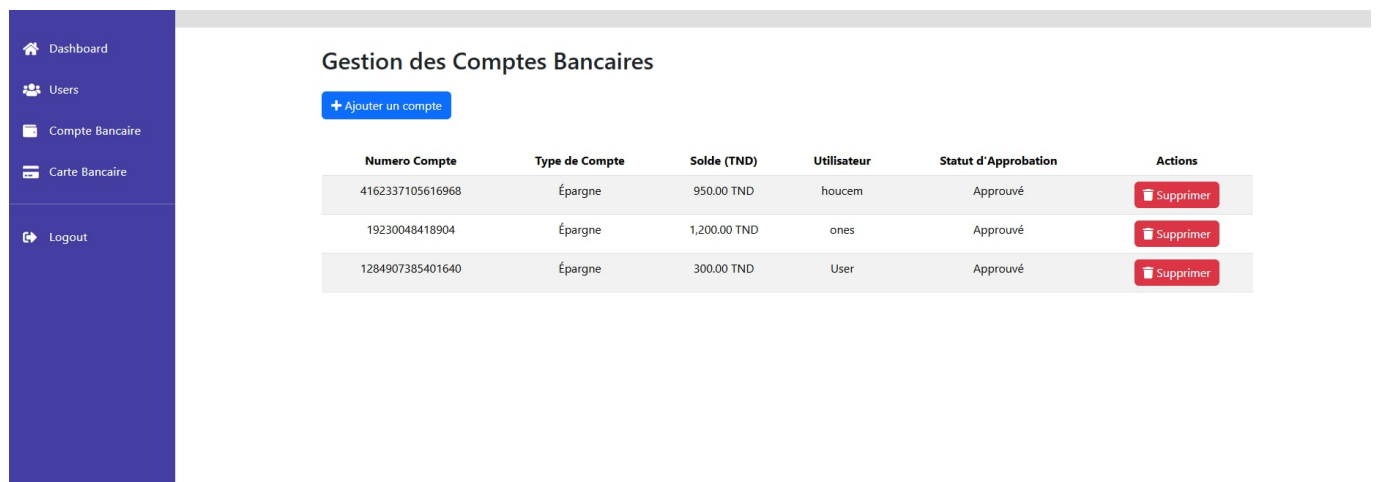


FIGURE 4.8 : Gestion de compte

Gestion des Comptes Bancaires

[+ Ajouter un compte](#)

Type de carte : Solde (TND) : Utilisateur :

[Enregistrer](#)

Numero Compte	Type de Compte	Solde (TND)	Utilisateur	Statut d'Approbation	Actions
4162337105616968	Épargne	950.00 TND	houcem	Approuvé	Supprimer
19230048418904	Épargne	1,200.00 TND	ons	Approuvé	Supprimer
1284907385401640	Épargne	300.00 TND	User	Approuvé	Supprimer

FIGURE 4.9 : Ajout de compte

phpMyAdmin

Récents Préférées

Nouvelle base de données bank

Nouvelle table

carte_bancaire

compte_bancaire

credit

role

transaction

user

user_role

information_schema

mysql

performance_schema

Serveur : 127.0.0.1 Base de données : bank Table : compte_bancaire

Parcourir Structure SQL Rechercher Insérer Exporter Importer Privileges Opérations Suivi Déclencheurs

Affichage des lignes 0 - 2 (total de 3, traitement en 0,0002 seconde(s).)

SELECT * FROM `compte_bancaire`

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

Tout afficher Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table Trier par clé : Aucun(e)

Options supplémentaires

			id_compte	approved	num_compte	solde	typecompte	user_id
<input type="checkbox"/>	Éditer	Copier	Supprimer	1	1	4162337105616968	950 Épargne	2
<input type="checkbox"/>	Éditer	Copier	Supprimer	2	1	19230048418904	1200 Épargne	3
<input type="checkbox"/>	Éditer	Copier	Supprimer	3	1	1284907385401640	300 Épargne	4

Tout cocher Avec la sélection : Éditer Copier Supprimer Exporter

FIGURE 4.10 : Ajout compte dans la base

4.2.6 Gestion de Carte

Gestion des Cartes Bancaires

Type de carte :

Choisir un type

Date d'expiration :

jj/mm/aaaa

Statut :

Numero du compte :

Ajouter

Numero Carte	Type	Expiration	Statut	Actions
7613668678062762	Épargne	2025-04-19	Active	Supprimer
8225234666715365	Épargne	2025-04-18	Active	Supprimer

FIGURE 4.11 : Gestion de carte

4.2.7 Historique de transactions

BankingApp

Dashboard

Accounts

Transactions

Logout

Transactions

Filter by Account Number:

Type	Amount	From Account	To Account	Date
↑ Sent	\$100.00	19230048418904	19230048418904	
↑ Sent	\$1,500.00	19230048418904	19230048418904	
↑ Sent	\$1,500.00	19230048418904	19230048418904	
↑ Sent	\$150.00	19230048418904	19230048418904	
↑ Sent	\$150.00	19230048418904	4162337105616968	
↑ Sent	\$150.00	19230048418904	1284907385401640	
↓ Received	\$100.00	19230048418904	19230048418904	
↓ Received	\$1,500.00	19230048418904	19230048418904	
↓ Received	\$1,500.00	19230048418904	19230048418904	
↓ Received	\$150.00	19230048418904	19230048418904	

FIGURE 4.12 : Historique de transactions

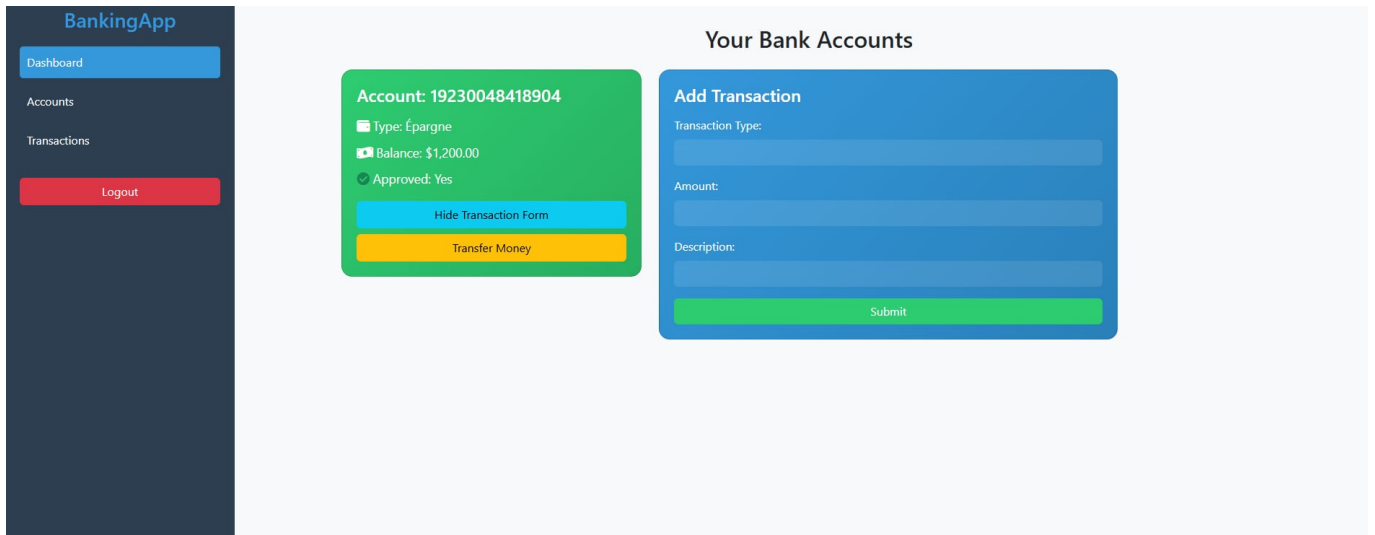


FIGURE 4.13 : Ajout d'une transaction

4.3 Conclusion

En conclusion, notre application web propose une interface intuitive et performante, favorisant une interaction fluide avec les données. Fruit d'un travail d'équipe collaboratif, elle répond aux attentes des utilisateurs tout en demeurant évolutive et ouverte à de futures améliorations.

Conclusion générale

En conclusion, ce projet de gestion bancaire basé sur une architecture web nous a permis de concevoir et de développer une solution logicielle robuste et fonctionnelle pour la gestion des comptes et des opérations bancaires. En adoptant les principes du pattern MVC, nous avons assuré une séparation claire des responsabilités entre les couches de modèle, de vue et de contrôleur, renforçant ainsi la maintenabilité et l'évolutivité de l'application.

En somme, ce projet de gestion bancaire fondé sur une architecture web nous a permis de concevoir et de développer une solution logicielle à la fois robuste et fonctionnelle, dédiée à la gestion des comptes et des opérations bancaires. L'adoption du modèle architectural MVC a facilité une séparation nette des responsabilités entre les couches modèle, vue et contrôleur, contribuant ainsi à la maintenabilité et à l'évolutivité de l'application.

L'intégration de technologies modernes telles que Spring Framework, Hibernate (JPA), ainsi que JSP et Servlets, nous a permis de renforcer notre maîtrise des outils de développement Java et du mapping objet-relationnel. Ce projet nous a également permis d'améliorer nos compétences dans la conception d'interfaces utilisateur dynamiques, ergonomiques et adaptées aux besoins concrets des utilisateurs.

Les résultats obtenus sont satisfaisants et démontrent que l'application répond efficacement aux exigences fonctionnelles d'un système bancaire tout en offrant une expérience utilisateur fluide. Néanmoins, des pistes d'amélioration restent envisageables, telles que l'ajout de nouvelles fonctionnalités, l'optimisation des performances ou encore l'intégration de services avancés comme l'analyse de données en temps réel.

Cette expérience a constitué une excellente opportunité de mettre en pratique nos acquis théoriques dans le cadre d'un projet concret et collaboratif. Elle a également renforcé notre compréhension des bonnes pratiques en ingénierie logicielle et nous prépare à relever des défis technologiques plus ambitieux à l'avenir.