BankDash.

Application développée avec Spring Boot, MongoDB, Angular, JWT, Docker, Jenkins, et AWS

Solution complète et moderne pour créer des tableaux de bord administratifs professionnels

---

Mené par :
NEJIB Hamza

Projet réalisé entre 14 Novembre 2024 et 30 Novembre 2024

Village de l'Emploi

Table des matières

Table des figures

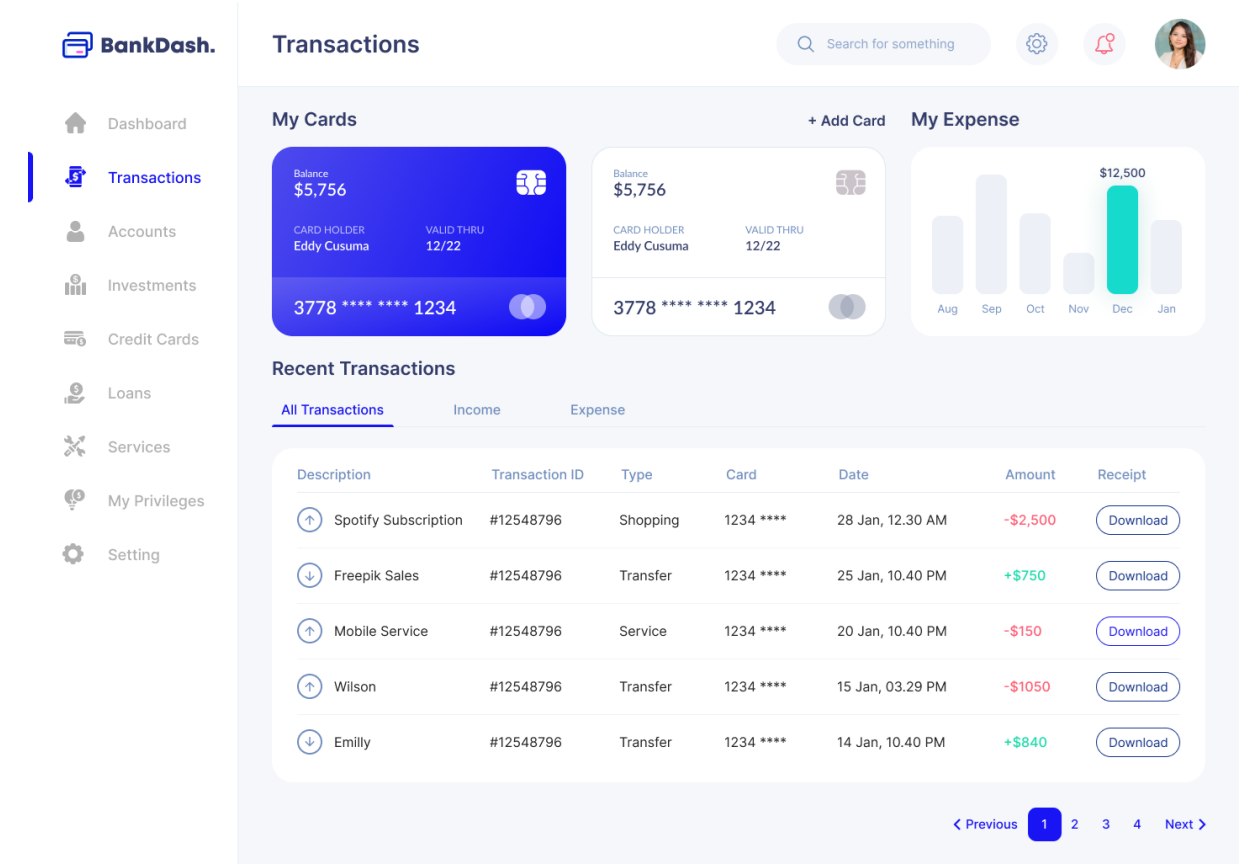
1 Résumé

BankDash est une application de gestion bancaire développée à partir d'une maquette récupérée sur Figma, l'application permet aux utilisateurs de visualiser leurs comptes, suivre leurs transactions, gérer leurs cartes de crédit, et effectuer des transferts rapides. L'application est divisée en une interface utilisateur (frontend) développée avec Angular, un backend en Spring Boot avec une base de données MongoDB, et des services de déploiement et CI/CD avec Docker, Jenkins, et AWS. J'ai récupéré la maquette sur Figma via le lien suivant : BankDash

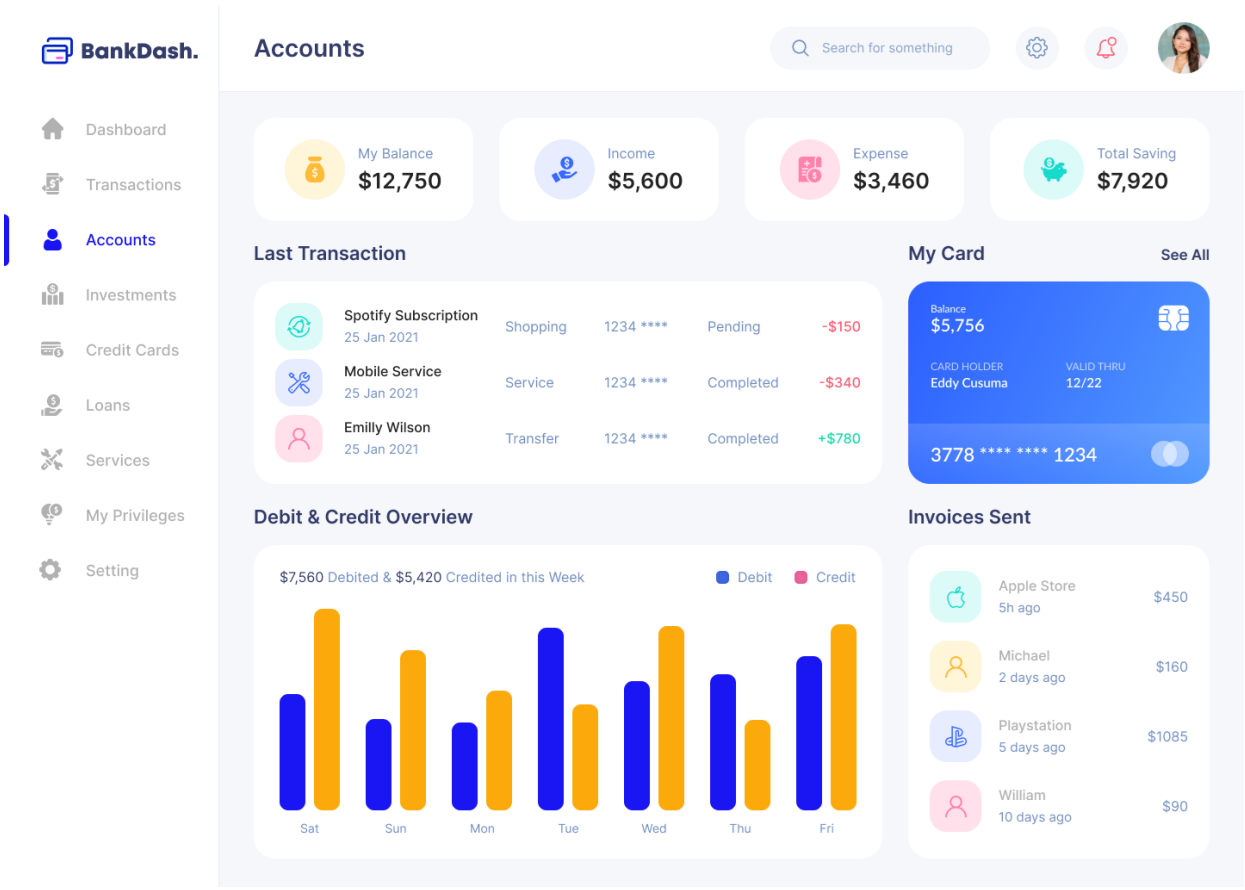
2 Pages de la maquette

Page d'accueil (Dashboard)

Transactions

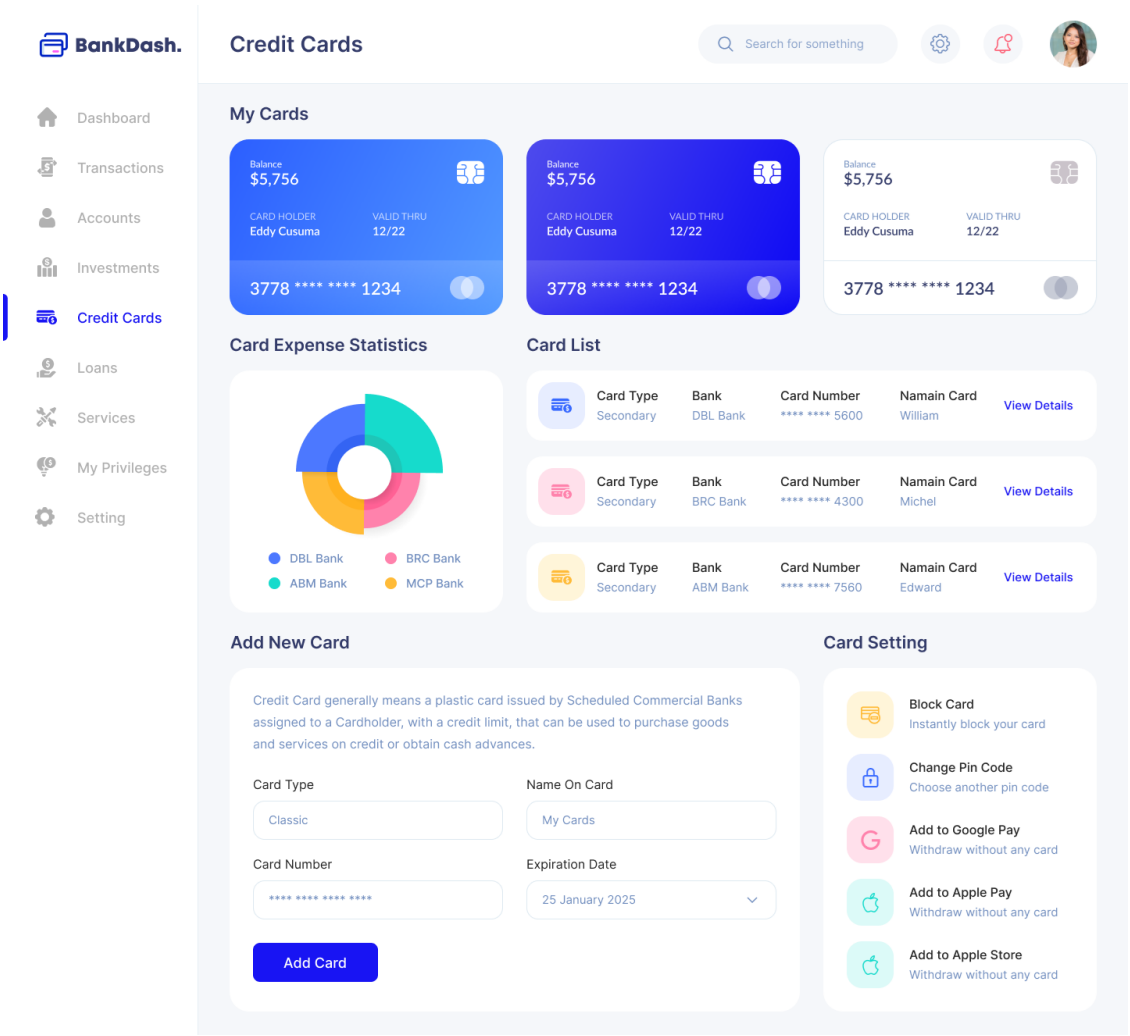


Accounts

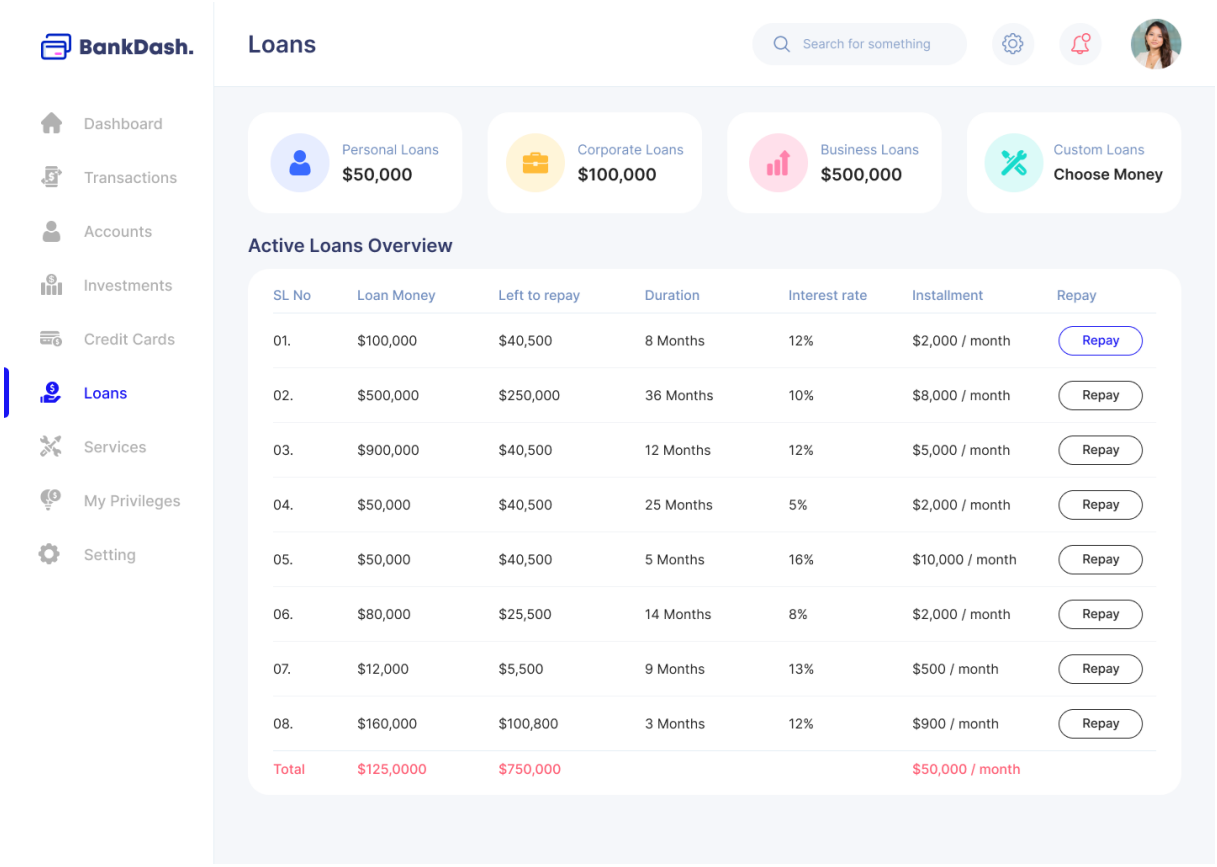


Investments

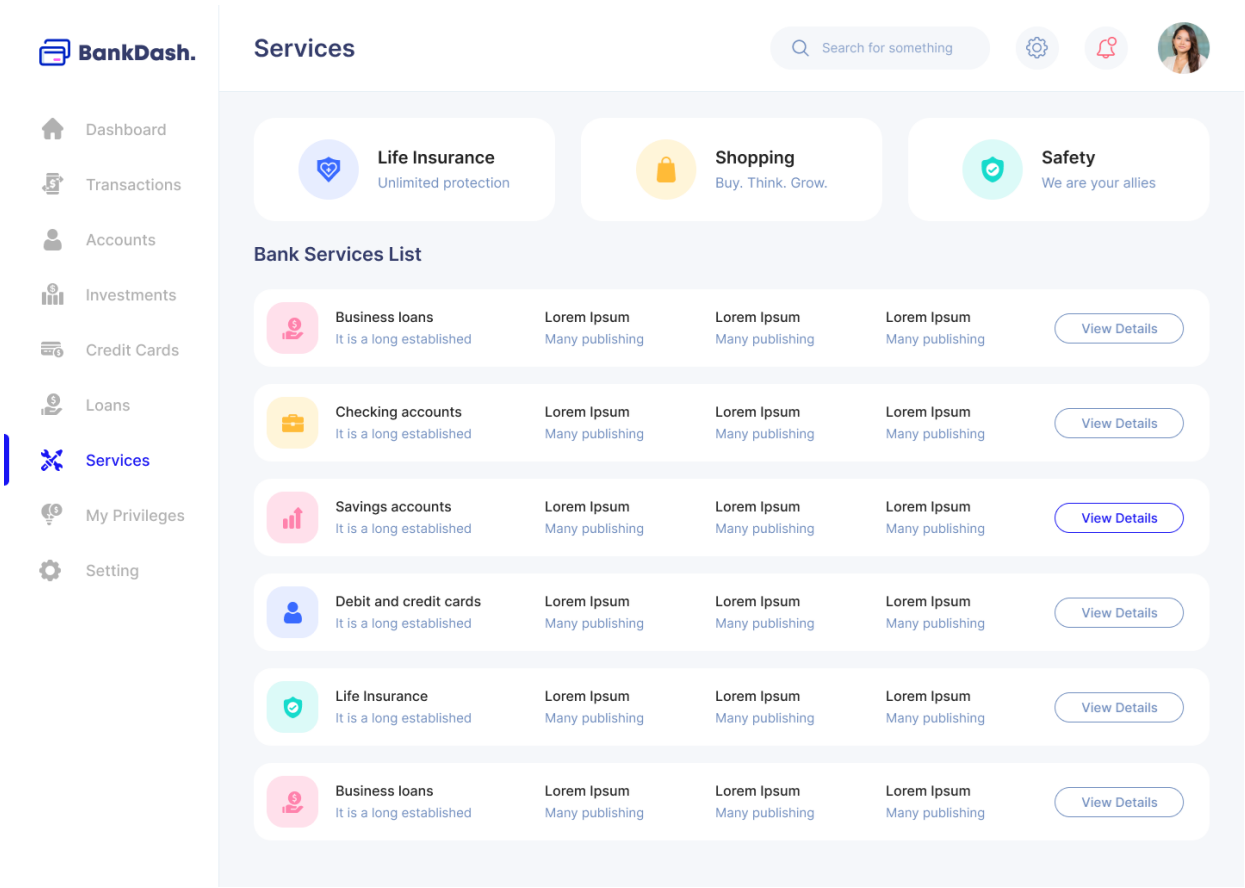
Credit Cards



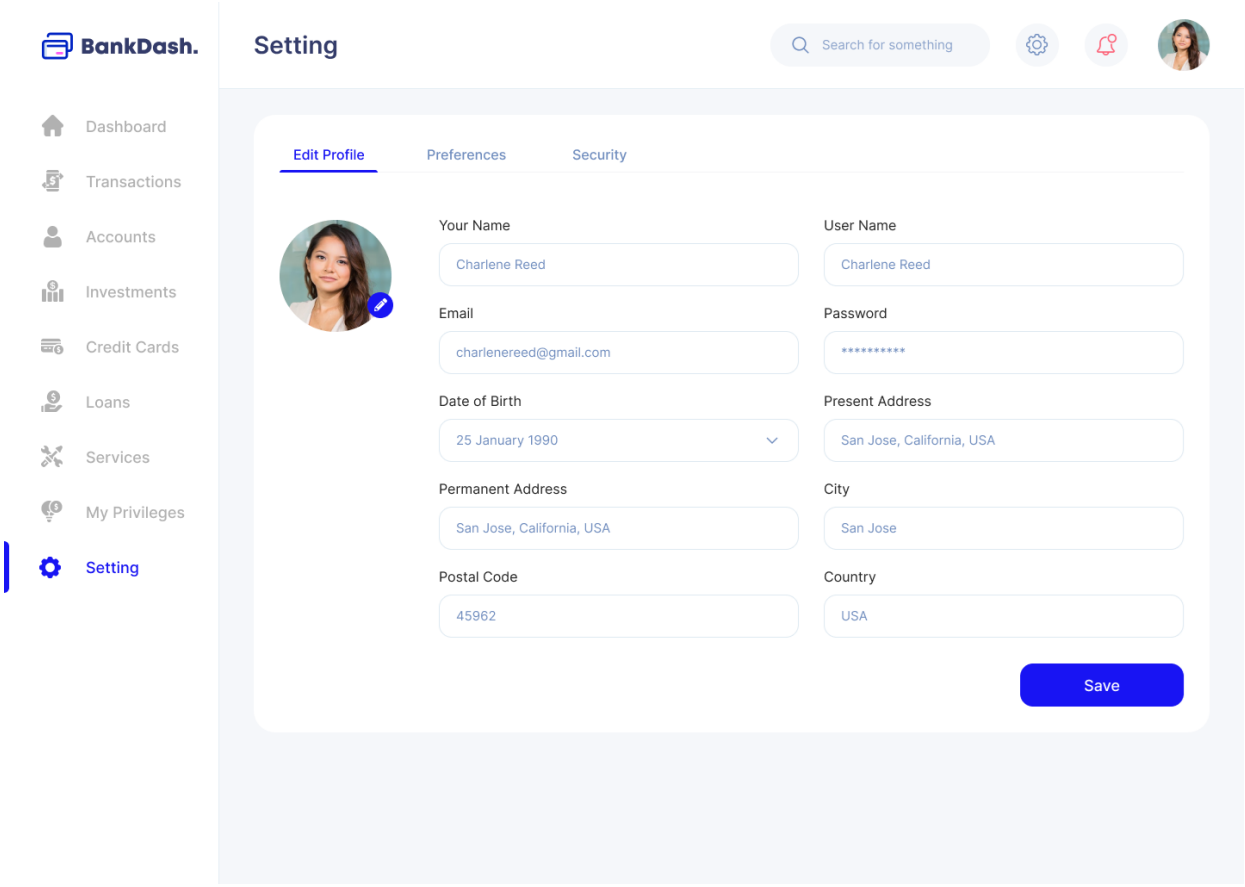
Loans



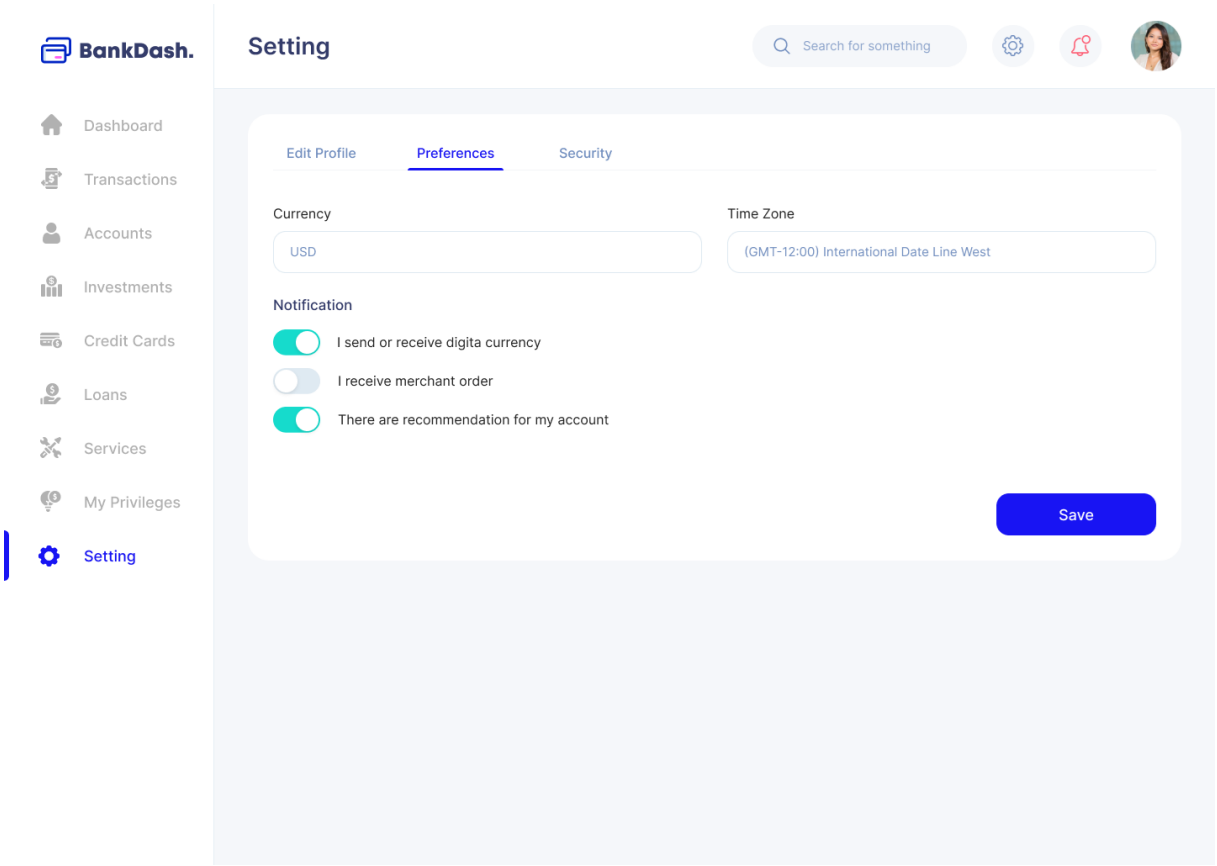
Services



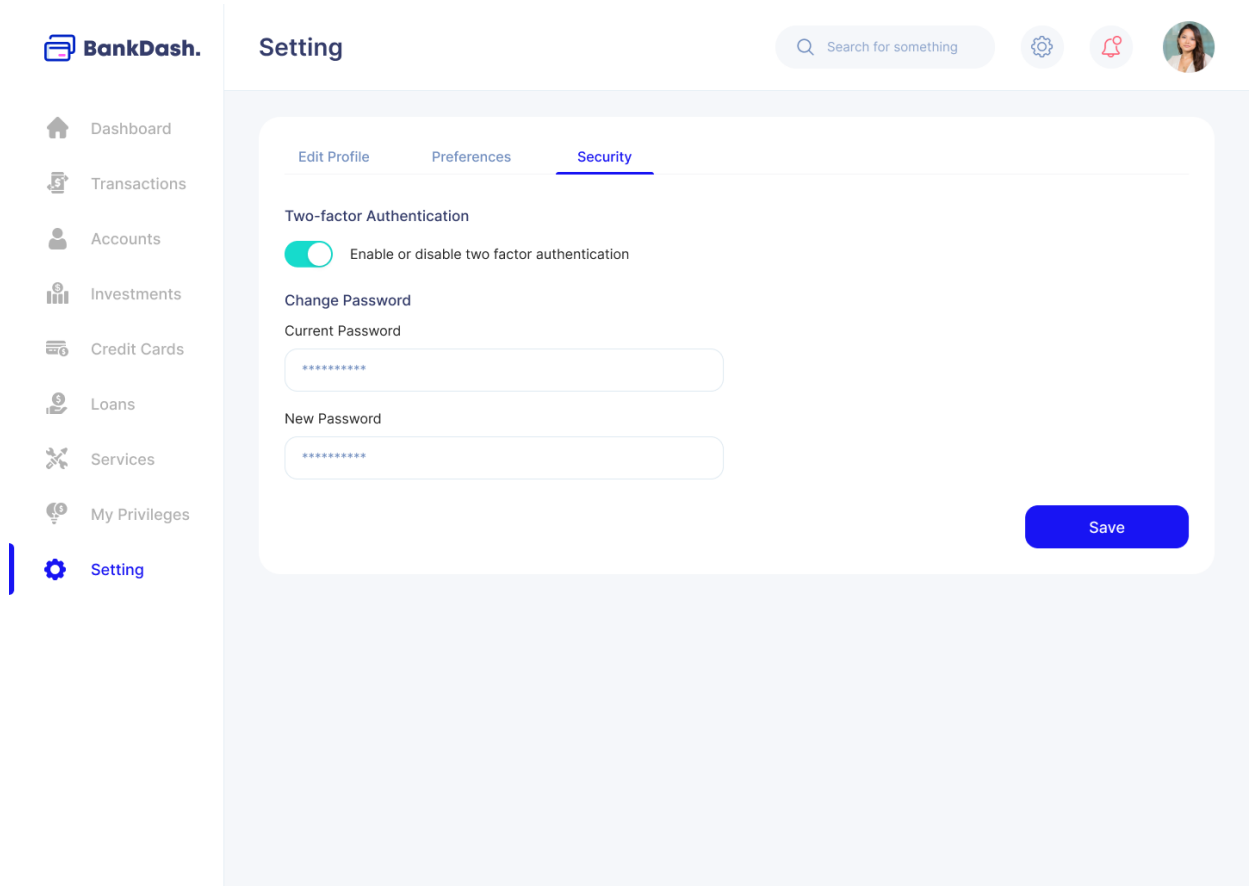
Setting Page 1



Setting Page 2



Setting Page 3



3 Fonctionnalités principales de l'application

BankDash, application bancaire moderne, doit répondre à ces attentes en offrant des fonctionnalités innovantes qui simplifient les opérations financières quotidiennes tout en garantissant une sécurité optimale.

3.1 Affichage des comptes bancaires et cartes de crédit

La gestion efficace des finances commence par une vision claire et organisée de ses comptes et cartes. Une application bancaire moderne doit fournir aux utilisateurs une interface intuitive qui centralise ces informations essentielles, facilitant ainsi le suivi et la prise de décisions financières. Voici les principales fonctionnalités liées à l'affichage des comptes et cartes :

Liste claire et organisée des comptes et cartes associés à l'utilisateur

- Présentation intuitive et hiérarchisée pour une consultation rapide

Informations détaillées pour chaque compte bancaire

- Numéro de compte (partiellement masqué pour la sécurité).
- Type de compte : épargne, courant, ou autre.
- Solde disponible pour les transactions et solde bloqué (le cas échéant).

Détails des cartes de crédit ou débit

- Affichage du numéro masqué (**** **** **** 1234) pour une confidentialité renforcée.
- Type de carte : crédit, débit ou prépayée.

- Date d'expiration pour éviter les oublis.
- Statut de la carte : active, bloquée, ou en attente d'activation.

3.2 Visualisation des transactions récentes et du solde en temps réel

La gestion de ses finances personnelles nécessite une transparence totale et une réactivité immédiate. Pour répondre à ces attentes, l'application permet aux utilisateurs de suivre en temps réel toutes leurs transactions bancaires, offrant ainsi une visibilité complète sur leurs finances. Chaque mouvement est mis à jour instantanément, permettant une gestion fluide et un contrôle total des soldes. Voici les principales fonctionnalités liées à la visualisation des transactions et du solde en temps réel :

Accès instantané à l'historique des transactions

Les utilisateurs peuvent consulter rapidement leurs dernières transactions, triées par ordre chronologique ou selon des filtres personnalisés (par montant, date, bénéficiaire, etc.).

Mise à jour en temps réel des soldes

Dès qu'une transaction est effectuée, le solde des comptes est mis à jour immédiatement, garantissant ainsi une vision exacte et instantanée des finances.

Détails complets des transactions

Pour chaque opération, les utilisateurs peuvent accéder aux informations détaillées : montant, bénéficiaire ou payeur, date, statut de la transaction (en attente, validé, échoué), ainsi que des notes éventuelles associées.

Notifications push ou par e-mail

Des alertes en temps réel sont envoyées pour informer les utilisateurs des transactions importantes ou suspectes, renforçant ainsi la sécurité et la réactivité face à toute activité inhabituelle.

3.3 Graphiques de dépenses par catégorie et historique des soldes.

Dans un environnement financier en constante évolution, il est essentiel pour les utilisateurs de pouvoir visualiser l'impact de leurs dépenses et d'analyser leurs habitudes. Grâce aux graphiques interactifs, l'application offre une vue détaillée et compréhensible de la répartition des dépenses par catégorie. Cela permet de mieux gérer son budget, d'identifier les domaines d'optimisation et de suivre l'évolution des économies. Voici les principales fonctionnalités liées aux graphiques de dépenses par catégorie et à l'historique des soldes :

Diagrammes et graphiques interactifs

Des graphiques clairs et dynamiques permettant de visualiser les dépenses selon différentes catégories (alimentation, logement, loisirs, transport, etc.), offrant ainsi une vue d'ensemble facile à interpréter.

Catégories de dépenses personnalisées

Les utilisateurs peuvent consulter leurs dépenses classées dans des catégories spécifiques, comme les courses alimentaires, les loisirs, les factures de logement, etc., afin de comprendre rapidement où va leur argent.

Comparaison mensuelle ou hebdomadaire des dépenses

Possibilité de comparer les dépenses sur différentes périodes (hebdomadaires, mensuelles, etc.), afin d'évaluer les variations dans les habitudes de consommation.

Affichage des tendances sur le long terme

Visualisation de l'évolution des soldes au fil du temps, avec des graphiques montrant l'augmentation ou la diminution des économies, permettant ainsi aux utilisateurs d'anticiper et de planifier leur avenir financier.

Export des graphiques et rapports

Possibilité d'exporter les graphiques, les rapports de dépenses et l'historique des soldes au format PDF ou CSV, facilitant ainsi le suivi financier et l'archivage des informations importantes.

3.4 Transferts rapides d'argent entre utilisateurs

Les transferts d'argent rapides et sécurisés sont essentiels dans une application bancaire moderne. Pour répondre à cette exigence, l'application permet des transferts instantanés, tant entre comptes bancaires qu'entre utilisateurs de la plateforme. Ces transferts sont simples à réaliser, avec une expérience fluide et intuitive, offrant à l'utilisateur la possibilité de transférer de l'argent en quelques clics, tout en garantissant la sécurité des transactions. Voici les principales fonctionnalités liées aux transferts rapides dargent entre utilisateurs :

Transferts instantanés entre comptes bancaires ou utilisateurs

L'application permet de réaliser des transferts d'argent instantanés, que ce soit entre comptes bancaires externes ou au sein de la plateforme entre utilisateurs.

Recherche de bénéficiaires par numéro de compte, adresse e-mail ou identifiant utilisateur

Pour simplifier le processus, les utilisateurs peuvent rechercher des bénéficiaires par différents critères (numéro de compte bancaire, e-mail ou identifiant unique au sein de la plateforme).

Confirmation en temps réel des transferts

Après chaque transfert, l'utilisateur reçoit une confirmation immédiate avec un résumé détaillé de l'opération (montant, bénéficiaire, date, etc.), assurant une transparence totale sur la transaction.

Option d'ajout de notes ou motifs pour chaque transfert

Les utilisateurs ont la possibilité d'ajouter des commentaires ou des motifs pour chaque transfert, ce qui peut être utile pour clarifier l'objet de la transaction.

Enregistrement des bénéficiaires pour des transferts fréquents

Pour faciliter les transferts futurs, l'application permet d'enregistrer les bénéficiaires fréquemment utilisés, réduisant ainsi les étapes nécessaires lors de prochains transferts.

3.5 Authentification sécurisée avec JWT

Dans le cadre de la sécurité des informations bancaires sensibles, une authentification robuste est cruciale. L'application adopte une méthode d'authentification basée sur JSON Web Token (JWT), un standard moderne permettant de garantir l'intégrité des sessions utilisateur et de sécuriser les communications. Cette approche renforce la sécurité tout en offrant une expérience fluide et rapide pour l'utilisateur. Voici les principales fonctionnalités liées à l'authentification sécurisée avec JWT :

Authentification basée sur JSON Web Token (JWT)

Chaque utilisateur se connecte de manière sécurisée grâce à un JWT, qui assure une gestion fiable et sécurisée des sessions, permettant de vérifier l'identité de l'utilisateur tout au long de sa navigation.

Support de l'authentification multi-facteurs (MFA)

Pour ajouter une couche de sécurité supplémentaire, l'application prend en charge l'authentification multi-facteurs (MFA) via un code temporaire (OTP) envoyé par SMS ou e-mail, ou en utilisant une application d'authentification dédiée (ex. : Google Authenticator).

Gestion automatique de l'expiration des tokens

Les tokens JWT ont une durée de vie limitée, ce qui garantit que les sessions ne restent pas ouvertes indéfiniment. Lorsqu'un token expire, l'utilisateur doit se reconnecter, renforçant ainsi la sécurité.

Protection contre les attaques CSRF (Cross-Site Request Forgery)

L'application intègre des mécanismes de protection contre les attaques CSRF, empêchant toute tentative malveillante d'effectuer des actions non autorisées au nom de l'utilisateur, en s'assurant que les requêtes sont légitimes et sécurisées.

3.6 Gestion des privilèges utilisateur et accès aux paramètres

La gestion des privilèges utilisateur et des paramètres est essentielle pour garantir un contrôle granulaire sur l'accès aux informations et aux fonctionnalités de l'application. En permettant une hiérarchisation des rôles, ainsi qu'une personnalisation des paramètres, l'application assure à chaque utilisateur un niveau d'accès adapté à ses besoins, tout en garantissant la sécurité et la confidentialité des données. De plus, un suivi complet des activités permet de renforcer la transparence et la responsabilité des utilisateurs. Voici les principales fonctionnalités liées à la gestion des privilèges utilisateur et à l'accès aux paramètres :

Hiérarchisation des rôles et permissions

L'application offre différents niveaux d'accès, tels que utilisateurs standard, administrateurs, et super-utilisateurs. Chaque rôle bénéficie de permissions adaptées à son niveau de responsabilité, garantissant un contrôle précis des actions possibles au sein de l'application.

Personnalisation des paramètres de l'application

Les utilisateurs peuvent personnaliser divers aspects de leur expérience, tels que la langue de l'application, les préférences de notification (notifications push, e-mail, SMS), ainsi que les méthodes d'authentification (par exemple, choix entre mot de passe ou authentification multi-facteurs).

Gestion des comptes liés

Les utilisateurs peuvent ajouter ou supprimer des comptes liés à leur profil, comme ceux d'un conjoint, enfant, ou autre membre de la famille, permettant une gestion commune des finances au sein d'un même compte.

Journal d'activité

Un journal d'activité permet de suivre toutes les actions effectuées sur le compte, incluant les connexions récentes et les actions réalisées (modifications de paramètres, transferts d'argent, etc.), offrant ainsi un historique transparent et sécuritaire de l'utilisation de l'application.

4 Architecture de l'application

4.1 Structure des Composants

L'application est constituée de plusieurs composants clés. Voici les composants principaux de l'application, leur rôle et les technologies associées :

4.1.1 Frontend (Angular)

Rôle Fournir une interface utilisateur réactive et dynamique.

Technologie Angular (un framework JavaScript pour la construction d'applications web à page unique).

Interaction Consomme les API REST du backend et fournit une interface riche pour l'utilisateur.

4.1.2 Backend API (Spring Boot)

Rôle Fournir des endpoints RESTful pour le frontend. Gère la logique métier, les traitements des données, et les requêtes HTTP.

Technologie Spring Boot (un framework Java basé sur Spring pour créer des applications de backend facilement configurables).

Interaction Expose des API REST qui sont consommées par le frontend. Interagit avec la base de données pour gérer les données utilisateurs et autres ressources.

4.1.3 Base de Données (MongoDB)

Rôle Stocker les informations essentielles comme les utilisateurs, les cartes, les transactions, etc.

Technologie MongoDB (une base de données NoSQL orientée documents, idéale pour les applications évolutives avec des données non structurées).

Interaction Le backend interroge et met à jour la base de données MongoDB selon les actions de l'utilisateur ou les processus métier.

4.1.4 Authentification et Autorisation (JWT)

Rôle Gérer l'authentification des utilisateurs et leur autorisation à accéder à certaines ressources.

Technologie JWT (JSON Web Token, utilisé pour sécuriser les échanges entre le frontend et le backend).

Interaction Lors de la connexion de l'utilisateur, un token JWT est généré, et ce token est utilisé pour sécuriser les appels API.

4.1.5 CI/CD (Jenkins, Docker)

Rôle Automatiser les processus de développement, tests, et déploiement.

Technologie Jenkins (outil d'intégration continue et de déploiement continu), Docker (pour conteneuriser les applications et faciliter leur déploiement).

Interaction Le code est automatiquement testé et déployé à chaque mise à jour via Jenkins, et Docker garantit que l'application fonctionne dans un environnement isolé et cohérent.

4.1.6 Déploiement Cloud (AWS)

Rôle Héberger l'application et la base de données dans le cloud.

Technologie Amazon Web Services (AWS) fournit une infrastructure cloud pour déployer des applications évolutives.

Interaction L'application backend et la base de données MongoDB sont déployées sur AWS, offrant ainsi une scalabilité, une sécurité et une disponibilité accrues.

4.2 Schéma de l'architecture

Le schéma suivant illustre la manière dont les différents composants interagissent :

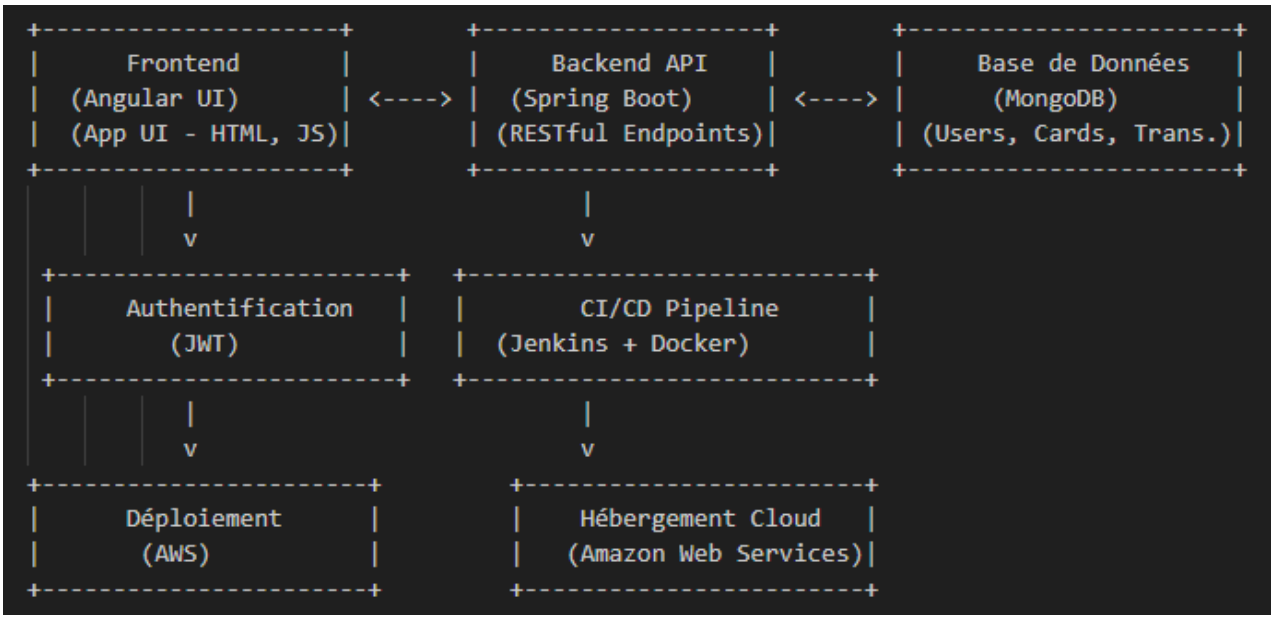


FIGURE 1 – Composants et interactions de l'application

L'architecture est basée sur une architecture client-serveur avec une séparation entre les composants :

1) **Client (Angular)** interface utilisateur qui consomme les API REST fournies par le backend.
2) **Serveur (Spring Boot)** gestion des requêtes, traitement des données, application de la logique métier.
3) **Base de Données (MongoDB)** stockage des données utilisateur, informations des cartes, historiques de transactions, etc.
4) **CI/CD (Jenkins et Docker)** pour le développement, le test et le déploiement continu.
5) **Hébergement (AWS)** déploiement du serveur et de la base de données.

```
BankDash/
├── Backend-Spring/          // Backend Spring Boot + MongoDB
├── Frontend-Angular/        // Frontend Angular
├── Docker/                  // Dockerfiles et Compose
├── Jenkins/                 // Scripts pour CI/CD avec Jenkins
├── AWS/                     // Scripts pour déploiement sur AWS
└── README.md                // Documentation
```

FIGURE 2 – Structure des dossiers de l'application

5 Modèle de Données et Schéma de la Base de Données

Le modèle de données pour cette application repose sur plusieurs collections MongoDB interconnectées, chacune représentant une entité ou un groupe d'entités spécifiques. Chaque collection est conçue pour stocker des informations pertinentes et permettre des requêtes efficaces. Ces

collections sont définies avec des schémas flexibles qui facilitent la gestion des données complexes tout en optimisant les performances de l'application.

5.1 Entités principales de l'application

5.1.1 User

Cette entité représente l'utilisateur de l'application et contient les informations personnelles et de sécurité nécessaires pour l'authentification et l'autorisation. L'entité User contient les attributs suivants :

- **id** Identifiant unique de l'utilisateur.
- **username** Nom d'utilisateur pour la connexion.
- **password** Mot de passe sécurisé (hashé).
- **email** Adresse e-mail de l'utilisateur.
- **role** Rôle de l'utilisateur (e.g., "USER", "ADMIN").
- **profilePicture** URL de la photo de profil de l'utilisateur.
- **createdAt** Date de création du compte.

5.1.2 Card

Cette entité représente les cartes bancaires de l'utilisateur affichées dans la section "My Cards". L'entité Card contient les attributs suivants :

- **id** Identifiant unique de la carte.
- **cardNumber** Numéro de la carte (affiché partiellement pour la sécurité).
- **cardHolder** Nom du titulaire de la carte (affiché comme "CARD HOLDER").
- **balance** Solde actuel de la carte.
- **validThru** Date d'expiration de la carte.
- **type** Type de carte (débit, crédit, etc.).
- **createdAt** Date d'ajout de la carte.
- **userId** Référence à l'utilisateur propriétaire de la carte.

5.1.3 Transaction

Cette entité représente les transactions financières de l'utilisateur. Les transactions récentes sont affichées dans la section "Recent Transaction". L'entité Transaction contient les attributs suivants :

- **id** Identifiant unique de la transaction.
- **transactionType** Type de transaction (e.g., "Deposit", "Withdraw").
- **amount** Montant de la transaction.
- **date** Date de la transaction.
- **description** Brève description de la transaction (ex. "Deposit from my Card").
- **source** Source de la transaction (ex. "Paypal").
- **userId** Référence à l'utilisateur qui a effectué ou reçu la transaction.
- **cardId** Référence à la carte associée (si applicable).

5.1.4 Statistic

Cette entité représente les statistiques de dépenses par catégorie pour l'utilisateur, visibles dans le graphique circulaire "Expense Statistics". L'entité Statistic contient les attributs suivants :

- **id** Identifiant unique de la statistique.
- **category** Catégorie de dépense (e.g., "Entertainment", "Bill Expense", "Investment", "Others").

- **percentage** Pourcentage de chaque catégorie de dépense par rapport aux dépenses totales.
- **amount** Montant total des dépenses dans cette catégorie pour une période donnée (facultatif).
- **userId** Référence à l'utilisateur associé.

5.1.5 Transfer

Cette entité représente les transferts d'argent entre utilisateurs, visible dans la section "Quick Transfer". L'entité Transfer contient les attributs suivants :

- **id** Identifiant unique du transfert.
- **amount** Montant transféré.
- **date** Date du transfert.
- **status** Statut du transfert (e.g., "Completed", "Pending").
- **message** Message optionnel pour le transfert.
- **receiverId** Référence à l'utilisateur qui reçoit l'argent.
- **senderId** Référence à l'utilisateur qui envoie l'argent.

5.1.6 BalanceHistory

Cette entité représente l'historique du solde de l'utilisateur, utilisé pour générer le graphique de l'évolution du solde ("Balance History"). L'entité BalanceHistory contient les attributs suivants :

- **id** Identifiant unique de l'historique de solde.
- **date** Date du solde enregistré.
- **balance** Solde à cette date.
- **createdAt** Date de création de l'enregistrement.
- **userId** Référence à l'utilisateur associé.

5.1.7 Notification

Bien que non visible dans la maquette, une entité Notification peut être ajoutée pour gérer les notifications de transaction ou d'autres événements importants. L'entité Notification contient les attributs suivants :

- **id** Identifiant unique de la notification.
- **message** Message de la notification.
- **type** Type de notification (e.g., "Transaction", "Alert").
- **date** Date de la notification.
- **readStatus** Indique si la notification a été lue.
- **userId** Référence à l'utilisateur concerné.

5.2 Relations entre les entités

- **User** a plusieurs Card, Transaction, Transfer (comme expéditeur et destinataire), Statistic, BalanceHistory, et Notification.
- **Card** appartient à un User et peut être associée à plusieurs Transaction.
- **Transaction** est associée à un User et à une Card.
- **Transfer** est associé à un User en tant qu'expéditeur et à un autre User en tant que destinataire.
- **Statistic** est associé à un User avec des informations sur les dépenses.
- **BalanceHistory** est associé à un User pour suivre les variations de son solde.
- **Notification** est associe à un User

6 Création du Backend avec Spring Boot et MongoDB

6.1 Initialiser le projet Spring Boot

6.1.1 Accéder au site Spring Initializr

J'ai accédé sur le site Spring Initializr et j'ai rempli les informations suivantes :

Project Maven
Language Java
Spring Boot Version 3.3.5
Group com.bankdash
Artifact bankdash
Name BankDash
Description Application de gestion bancaire avec un tableau de bord
Packaging Jar
Java Version 17

6.1.2 Ajouter les dépendances nécessaires

J'ai cliqué sur "Add Dependencies" et j'ai ajouté les dépendances suivantes :

- **Spring Web** pour la création de l'API REST.
- **Spring Data MongoDB** pour l'intégration avec MongoDB.
- **Spring Security** pour la sécurité et l'authentification.
- **Lombok** pour générer automatiquement les getters, setters, etc

6.1.3 Générer le projet

J'ai cliqué sur "Generate" pour télécharger le projet. Une fois généré, j'ai décompressé-le et le copié dans le dossier du projet de l'application BankDash. Enfin, j'ai ajouté, commité et poussé les fichiers sur GitHub.