### Rapport de Projet : Système de Transfert d'Argent

#### 1. Introduction

Le projet a pour objectif de concevoir et de développer une application web sécurisée pour le transfert de fonds. L'application permet aux utilisateurs de gérer leurs transactions, de consulter l'historique de leurs opérations et de gérer leurs comptes. Elle s'appuie sur une architecture moderne combinant un **backend en Spring Boot** et un **frontend en Angular**, garantissant une interface utilisateur réactive et une architecture robuste.

#### 2. Objectifs du Projet

* **Gestion des comptes et authentification sécurisée** : Permettre aux utilisateurs de s'inscrire et de se connecter via un système d'authentification robuste (JWT, Spring Security).
* **Transferts de fonds** : Faciliter les transferts d'argent sécurisés entre comptes utilisateurs avec calcul automatique des frais.
* **Opérations gérées par des agents** : Mettre en place des fonctionnalités de **dépôt** et de **retrait** d'argent, exclusivement gérées par un agent administrateur pour une sécurité et une traçabilité accrues.
* **Historique des transactions** : Fournir un historique complet et des reçus électroniques pour toutes les opérations.
* **Interface d'administration** : Développer un tableau de bord pour les administrateurs et les agents afin de superviser et de gérer l'ensemble des utilisateurs, des comptes et des transactions.

#### 3. Architecture du Système

Le système est conçu selon une **architecture en couches**, où chaque composant a une responsabilité bien définie, facilitant la maintenance et l'évolutivité. La communication entre le frontend et le backend se fait via une **API REST**.

#### 3.1. Architecture du Backend (Spring Boot)

L'arborescence des fichiers et l'organisation en packages reflètent clairement la structure de l'application :

* **controllers** : Reçoivent les requêtes HTTP et les dirigent vers les services appropriés.
* **services** : Contiennent la logique métier. Des services spécialisés, comme **AdminService**, **TransactionService**, **TransfertService** et **UserService**, gèrent des domaines spécifiques.
* **repositories** : Assurent l'interaction avec la base de données (MySQL) via JPA.
* **models** : Représentent les entités de la base de données.
* **dto** : Objets de transfert de données pour une communication sécurisée et efficace avec le frontend.

#### 3.2. Sécurité et Services Clés

L'architecture logicielle est soutenue par des services dédiés, chacun jouant un rôle crucial dans le fonctionnement et la sécurité du système :

* **SecurityConfig** : Le pilier de la sécurité. Cette classe configure Spring Security pour utiliser une architecture **stateless** basée sur les tokens JWT. Elle définit également les règles d'accès aux URLs, garantissant que seuls les utilisateurs authentifiés et/ou autorisés peuvent accéder aux ressources spécifiques. Elle inclut une configuration robuste pour le hachage des mots de passe (BCryptPasswordEncoder) et la gestion des requêtes CORS.
* **UserService** : Gère toutes les actions de l'utilisateur sur son profil. Il permet de consulter et de mettre à jour les informations personnelles, ainsi que de changer le mot de passe de manière sécurisée en vérifiant le mot de passe actuel et les validations. L'utilisation des DTOs et les validations d'unicité (email, téléphone) renforcent la sécurité et la qualité des données.
* **CompteService** : Se concentre sur la gestion des comptes utilisateurs. Il est notamment capable de récupérer le compte de l'utilisateur actuellement connecté de manière sécurisée en utilisant le SecurityContextHolder de Spring.
* **TransfertService** : C'est le cœur de l'application pour les transferts entre utilisateurs. La méthode **effectuerTransfert** gère de manière atomique la vérification du solde, le calcul des frais (1%), le débit du compte source et le crédit du compte destinataire. L'annotation @Transactional est essentielle pour garantir l'intégrité financière.
* **AdminService** : Gère les opérations sensibles, permettant de modifier les rôles, d'activer/désactiver des comptes et d'exécuter des **dépôts et des retraits** sous la supervision d'un agent administrateur.
* **TransactionService** : Axé sur l'utilisateur final, ce service gère les fonctionnalités d'historique de transactions. Il permet à un utilisateur de consulter ses transactions (envoyées, reçues, complètes), de générer des **reçus PDF**, et d'obtenir des statistiques sur ses propres transactions.

#### 4. Fonctionnalités Implémentées

* **Interface utilisateur sécurisée** : Les utilisateurs peuvent s'inscrire et se connecter, avec un système de gestion des profils.
* **Transferts de fonds** : Un utilisateur peut envoyer de l'argent à un autre, avec une vérification de son solde et le calcul automatique des frais.
* **Opérations de dépôt et de retrait** : Ces opérations sont exclusivement gérées par des agents administrateurs via une interface d'administration dédiée. Elles sont enregistrées et traçables pour chaque agent admin.
* **Historique et reçus** : Chaque transaction génère un reçu électronique qui peut être téléchargé en format PDF, consultable dans l'historique complet ou filtré.
* **Tableau de bord de gestion** : Les administrateurs ont accès à un tableau de bord leur permettant de superviser les transactions et les utilisateurs, d'obtenir des statistiques globales et de gérer les profils.

#### 5. Avantages et Difficultés

* **Avantages** :
  + **Sécurité et traçabilité** : Le contrôle des dépôts et retraits par un agent admin , ainsi que les vérifications intégrées dans le TransfertService, réduisent les risques de transactions non autorisées et améliorent l'audit.
  + **Architecture modulaire** : La séparation claire des responsabilités entre les contrôleurs, les services et les dépôts rend le projet facile à maintenir et à faire évoluer.
  + **Expérience utilisateur** : L'interface réactive et les fonctionnalités claires (historique, reçus) offrent une expérience intuitive.
* **Difficultés Rencontrées** :
  + Mise en place d'un système de **gestion des rôles** complexe avec Spring Security.
  + Gestion de la **synchronisation** entre le frontend et le backend, notamment pour les tokens JWT et la gestion des fichiers PDF.
  + Sécurisation des opérations et des données pour empêcher les accès non autorisés.

#### 6. Conclusion et Perspectives

Le projet "Système de Transfert d'Argent" aboutit à une application complète et sécurisée, répondant à tous les objectifs initiaux. L'architecture technique est robuste et offre une base solide pour le développement futur de fonctionnalités supplémentaires. **Perspectives d'amélioration** : L'application est prête à être déployée et peut évoluer pour intégrer des fonctionnalités avancées comme les notifications en temps réel ou le support multi-devises.