Documentation Complète — Projet SimpleCash

1. Présentation du Projet

1.1 Contexte

Le projet **SimpleCash** est une application bancaire simplifiée développée avec **Spring Boot**, **JPA/Hibernate**, **H2**, et une architecture REST.
Elle permet de gérer :

- les agences bancaires,
- les gérants,
- les conseillers,
- les clients,
- les comptes bancaires,
- les opérations de crédit, débit et virement,
- ainsi qu'un module d'audit.

L'application vise à reproduire un fonctionnement bancaire simplifié avec règles métiers.

1.2 Objectifs du Projet

- Mettre en œuvre une **architecture en couches**
- Utiliser **Spring Boot + JPA**
- Implémenter des **API REST**
- Gérer des **DTO**
- Appliquer des **règles métier**
- Tester via **Postman**

- Structurer un projet professionnel prêt à être maintenu

---

2. Architecture Générale

2.1 Architecture en Couches

Le projet suit une architecture **3-tiers** :

Couche	Rôle
Controller	Expose les endpoints REST
Service	Contient la logique métier
Repository	Accès à la base de données
Entity	Modèle JPA
DTO	Objets d’échange API
Mapper	Conversion Entity ⇄ DTO

2.2 Technologies Utilisées

- Java 17
- Spring Boot
- Spring Web
- Spring Data JPA
- H2 Database
- Hibernate
- Maven
- Postman
- Lombok

---

3. Modèle Métier (Entités)

3.1 Agence

- id (String)
- dateCreation

3.2 Gérant

- id (Long)
- nom
- prenom
- agence

3.3 Conseiller

- id (Long)
- nom
- prenom
- agence

3.4 Client

- id (Long)
- nom
- prenom
- adresse
- codePostal
- ville

- telephone
- typeClient (PARTICULIER / ENTREPRISE)
- conseiller

3.5 Compte

- id (Long)
- numero
- solde
- dateOuverture
- typeCompte
- client

---

4. Règles Métier

Fonction	Règle
Suppression compte	Autorisée uniquement si solde = 0
Débit	Interdit si solde insuffisant
Virement	Comptes distincts uniquement
Montant	Doit toujours être positif
Gérant	Peut ajouter des conseillers à son agence
Audit	Détection de comptes débiteurs anormaux

---

5. Fonctionnalités Implémentées

CRUD Agence
CRUD Gérant

CRUD Conseiller
CRUD Client
CRUD Compte
Crédit / Débit
Virement bancaire
Audit bancaire
DTO & Mapper manuel
Tests Postman
Script `data.sql`

---

6. API REST — Documentation

6.1 Clients

Méthode	URL
GET	/api/clients
POST	/api/clients
GET	/api/clients/{id}
PATCH	/api/clients/{id}
DELETE	/api/clients/{id}

6.2 Comptes

Méthode	URL
POST	/api/comptes/client/{clientId}
GET	/api/comptes/{id}
POST	/api/comptes/credit
POST	/api/comptes/debit
DELETE	/api/comptes/{id}

6.3 Virement

Méthode	URL
POST	/api/virements

6.4 Audit

Méthode	URL
GET	/api/audit/comptes/particuliers-anormaux
GET	/api/audit/comptes/entreprises-anormaux
GET	/api/audit/global

---

7. Données Initiales (`data.sql`)

Au démarrage, l'application crée automatiquement :

- 1 Agence
- 1 Gérant
- 1 Conseiller
- 2 Clients
- 2 Comptes

Cela permet de tester immédiatement toutes les fonctionnalités.

---

8. Tests Réalisés (Postman)

8.1 Tests Clients

- Création client
- Modification client
- Suppression client
- Récupération client

8.2 Tests Comptes

- Création compte
- Crédit compte
- Débit compte

8.3 Tests Virement

- Virement OK
- Solde insuffisant
- Virement sur le même compte

8.4 Tests Audit

- Audit global
- Comptes particuliers anormaux
- Comptes entreprises anormaux

---

9. Sécurité & Validation

- `@Valid`
- `@NotBlank`
- `@NotNull`
- Gestion des erreurs centralisée avec `@RestControllerAdvice`

---

10. Organisation du Projet

```
com.simplecash
 ├ controller
 ├ dto
 ├ entity
```

```
├ mapper
├ repository
├ service
├ service.impl
└ SimplecashApplication.java
```

---

11. Conclusion

Le projet **SimpleCash** respecte les standards d'un projet **Spring Boot professionnel** avec :

- séparation claire des responsabilités,
- utilisation propre des DTO,
- services métiers bien isolés,
- règles bancaires respectées,
- audit fonctionnel,
- base de données initialisée automatiquement,
- tests API validés avec Postman.