Projet de Développement Web JEE

Introduction

Ce document présente les exigences fonctionnelles et techniques pour la création d'une application Web JEE destinée à la gestion des comptes bancaires. L'objectif est de fournir une solution robuste, sécurisée et facile à maintenir pour la gestion des clients, des comptes et des opérations bancaires.

I. Exigences Fonctionnelles

L'application doit permettre de :

1. Gérer les clients

- **Ajouter un client** : Permettre l'ajout de nouveaux clients dans le système.
- **Consulter tous les clients** : Afficher la liste complète des clients enregistrés.
- **Consulter les clients dont le nom contient un mot-clé** : Rechercher des clients en fonction d'un mot-clé spécifique.

2. Gérer les comptes

- **Ajouter un compte** : Créer un nouveau compte bancaire pour un client existant.
- **Consulter un compte** : Afficher les détails d'un compte spécifique.

3. Gérer les opérations

- **Effectuer un versement** : Ajouter un montant à un compte spécifique.
- **Effectuer un retrait** : Retirer un montant d'un compte spécifique.
- **Effectuer un virement** : Transférer un montant d'un compte à un autre.
- **Consulter les opérations d'un compte page par page** : Afficher les opérations d'un compte de manière paginée.
- **Authentification** : Toutes les opérations nécessitent une authentification préalable.

II. Exigences Techniques

Architecture Technique

L'application sera basée sur une architecture typique d'une application web Java utilisant le framework Spring Boot. Voici une explication détaillée de chaque composant :

1. Conteneur Spring Boot IOC

- **IOC (Inversion of Control)** : C'est le cœur de Spring. Il gère la création et la configuration des objets de l'application, en injectant les dépendances automatiquement.

2. Couches de l'application

- **Couche Web** :
    - **Serveur Tomcat** : Conteneur de servlet qui exécute l'application web.
    - **Controllers** : Gèrent les requêtes HTTP, exécutent la logique métier et renvoient une réponse (souvent une vue).
    - **Vues** : Pages HTML dynamiques générées à partir des données du modèle.
    - **Spring MVC** : Framework MVC de Spring pour la gestion des requêtes et des réponses HTTP.
    - **Spring Security** : Gère l'authentification et l'autorisation des utilisateurs.
    - **HTML5 et Bootstrap** : Technologies utilisées pour créer l'interface utilisateur.
- **Couche Métier** :
    - **Interface Métier** : Définit le contrat des services métier.
    - **Implémentation Métier** : Contient la logique métier réelle.
- **Couche DAO (Data Access Object)** :
    - **Entities** : Classes Java représentant les tables de la base de données.
    - **JPARepository** : Interface de Spring Data JPA pour les opérations CRUD.
    - **Spring Data, JPA, Hibernate, JDBC** : Technologies pour l'accès à la base de données.

III. Flux de l'Application

1. Une requête HTTP arrive sur le serveur Tomcat.
2. Spring MVC intercepte la requête et la dirige vers le contrôleur approprié.
3. Le contrôleur appelle les services métier pour effectuer les traitements nécessaires.
4. Les services métier interagissent avec la couche DAO pour accéder aux données en base de données.
5. Le contrôleur renvoie une vue à Spring MVC, qui la rend au client.
6. Les données sont stockées dans une base de données MySQL.

IV. Couches de l'Application

1. **Couche DAO** :
    - Basée sur Spring Data, JPA, Hibernate et JDBC.
2. **Couche Métier** :
    - Contient la logique métier.
3. **Couche Web** :
    - Basée sur MVC côté Serveur en utilisant Thymeleaf.
4. **Sécurité** :
    - Basée sur Spring Security.

V. Explication des Composants

1. **MySQL** : Système de gestion de base de données relationnelle.
2. **DAO (Data Access Object)** : Couche d'interaction avec la base de données.
3. **Couche Métier** : Logique métier et règles de gestion des données.
4. **MVC (Modèle-Vue-Contrôleur)** :
    - **Modèle** : Représente les données.
    - **Vue** : Présente les données à l'utilisateur.
    - **Contrôleur** : Gère les interactions entre le modèle et la vue.
5. **Thymeleaf** : Moteur de template pour créer des vues web dynamiques.
6. **Spring Security** : Framework de sécurité pour l'authentification et l'autorisation.

Conclusion

Ce cahier des charges doit guider le développement de l'application Web JEE pour la gestion des comptes bancaires. Les exigences fonctionnelles et techniques décrites doivent être respectées pour garantir une application robuste, sécurisée et facile à maintenir.