Rapport : Application Gestion de Finance et Dashboard

Fait par : Manal HAGGOUD et Hala TOB

January 17, 2025

Introduction

Ce rapport technique décrit l'application "Gestion de Finance et Dashboard" développée dans le cadre d'un projet de gestion des finances personnelles avec une interface web permettant de visualiser les transactions et d'obtenir des informations détaillées à l'aide d'un tableau de bord. Ce projet est développé en Java, utilisant Spring Boot pour le backend et Thymeleaf pour la gestion des vues.

L'application permet à l'utilisateur de saisir des transactions financières, d'afficher un tableau de bord avec des graphiques et des statistiques, et de gérer les différentes données financières.

Architecture du Projet

L'architecture du projet est basée sur une structure MVC (Model-View-Controller), ce qui permet une séparation claire entre les données, la logique métier et l'interface utilisateur.

Structure des Dossiers

La structure du projet est la suivante :

```
src
 main
    java
       com.example.bigdata
          controller
             HomeController.java
             DashboardController.java
             TransactionsController.java
          entity
             Transaction.java
          repository
             TransactionRepository.java
          service
             DashboardService.java
             TransactionService.java
    resources
       templates
          home.html
          dashboard.html
```

```
    transactions.html
application.properties
```

Description des Composants

- **HomeController.java** : Gère les requêtes liées à la page d'accueil.
- **DashboardController.java** : Traite les données relatives au tableau de bord et les affiche sous forme de graphiques.
- **TransactionsController.java** : Permet à l'utilisateur d'ajouter, modifier et afficher les transactions financières.
- **Transaction.java** : Représente une transaction financière dans l'application.
- **TransactionRepository.java** : Fournit les méthodes nécessaires pour interagir avec la base de données et gérer les transactions.
- **DashboardService.java** : Contient la logique métier pour la gestion des statistiques et des graphiques sur le tableau de bord.
- **TransactionService.java** : Contient la logique métier pour la gestion des transactions financières.

Détail des Classes et Méthodes

HomeController.java

Cette classe est responsable de la gestion des requêtes liées à la page d'accueil de l'application. Elle contient des méthodes qui retournent des vues simples et permettent à l'utilisateur de naviguer facilement dans l'application.

Exemple de code de la classe `HomeController.java` :

```
@Controller
public class HomeController {

    @GetMapping("/")
    public String home() {
        return "home";  // Retourne la vue 'home.html'
    }
}
```

DashboardController.java

Le `DashboardController` est utilisé pour récupérer les données nécessaires au tableau de bord et afficher les statistiques sous forme de graphiques. Ce contrôleur communique avec le service `DashboardService` pour obtenir les informations nécessaires à l'affichage des graphiques.

Exemple de code de la classe `DashboardController.java` :

```
@Controller
public class DashboardController {

    @Autowired
    private DashboardService dashboardService;

    @GetMapping("/dashboard")
    public String dashboard(Model model) {
        List<Statistic> stats = dashboardService.
            getDashboardStatistics();
        model.addAttribute("statistics", stats);
        return "dashboard";  // Retourne la vue 'dashboard.
            html'
    }
}
```

Transaction.java

La classe `Transaction` représente une transaction financière. Elle contient des attributs comme la date, le montant, la catégorie, et la description de la transaction.

Exemple de code de la classe `Transaction.java` :

```
@Entity
public class Transaction {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Double amount;
    private String category;
    private String description;
    private LocalDate date;

    // Getters and Setters
}
```

TransactionRepository.java

Le `TransactionRepository` est une interface qui étend `JpaRepository`, permettant ainsi d'interagir avec la base de données pour ajouter, supprimer, et récupérer les transactions.

Exemple de code de la classe `TransactionRepository.java` :

```
public interface TransactionRepository extends JpaRepository<
   Transaction, Long> {
    List<Transaction> findByCategory(String category);
}
```

DashboardService.java

La classe `DashboardService` contient la logique métier pour calculer les statistiques financières à afficher dans le tableau de bord. Elle peut calculer des sommes, des moyennes, ou tout autre type de statistique nécessaire pour l'application.

Exemple de code de la classe `DashboardService.java` :

```
@Service
public class DashboardService {

    @Autowired
```

```
    private TransactionRepository transactionRepository;

    public List<Statistic> getDashboardStatistics() {
        List<Transaction> transactions =
            transactionRepository.findAll();
        // Logique pour calculer les statistiques
        return statistics;
    }
}
```

Interface Utilisateur

L'interface utilisateur de l'application est composée de plusieurs vues gérées par Thymeleaf. Les principales pages sont la page d'accueil, le tableau de bord et la gestion des transactions.

home.html

Cette vue affiche un accueil simple avec des liens vers les autres pages.

Exemple de code de `home.html` :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Home</title>
</head>
<body>
    <h1>Bienvenue dans l'application de gestion de finance</
       h1>
    <a href="/dashboard">Voir le tableau de bord</a>
    <a href="/transactions">G rer les transactions</a>
</body>
</html>
```

dashboard.html

Le tableau de bord affiche des graphiques et des statistiques relatives aux finances de l'utilisateur.

Exemple de code de `dashboard.html` :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Dashboard</title>
</head>
<body>
    <h1>Tableau de bord financier</h1>
    <div>
        <h2>Statistiques</h2>
        <ul>
            <li th:each="stat␣:␣${statistics}">
                <p th:text="${stat.name}">Statistique</p>
                <p th:text="${stat.value}">Valeur</p>
            </li>
        </ul>
    </div>
```

```
</body>
</html>
```

Conclusion

Ce projet fournit une solution simple mais complète pour la gestion des finances personnelles, avec une interface web permettant de visualiser les transactions et d'afficher des statistiques à l'aide d'un tableau de bord interactif. Les technologies utilisées, telles que Spring Boot, Thymeleaf et JPA, permettent de construire une application robuste et scalable.

L'extension de cette application pour inclure plus de fonctionnalités, telles que la gestion des utilisateurs, l'ajout d'analyses financières avancées, ou l'intégration d'APIs externes, pourrait en faire un outil encore plus puissant pour la gestion des finances personnelles.