MOHAMED LACHGAR

Nom des memebres:

Elgbouri saad & Temsamani omar & Oukattou soufiane

Gestion de depenses

miscroservices

Rapport Projet : Architecture des composants d’entreprise

1. Introduction

* Aperçu du projet :

Notre projet c’est Wallet app qui a plusieurs fonctionnalités, c’est pour la gestion des dépenses :

Mise en avant des Fonctionnalités

Optimisez la gestion de vos revenus, contrôlez le budget familial, suivez vos dépenses en déplacement, et comprenez l'emplacement de l'ensemble de vos fonds grâce à des fonctionnalités à la fois exceptionnelles et simples d'utilisation.

Gestion Budgétaire Sans Effort

Enregistrez vos transactions quotidiennes en quelques secondes seulement. Organisez-les au sein de catégories claires et visuelles telles que Dépenses : Alimentation, Shopping ou Revenus : Salaire, Cadeau.

Analyse des Dépenses

Un rapport détaillé offrant une vue claire de vos habitudes de dépenses. Comprenez l'origine et la destination de vos fonds à travers des graphiques clairs et facilement interprétables.

Suivi Financier Simplifié

Enregistrez vos transactions quotidiennes en quelques secondes. Catégorisez-les de manière claire et visuelle, comme Dépense : Alimentation, Shopping, ou Revenu : Salaire, Cadeau.

Sécurité et Confidentialité Assurées

La sécurité de vos données financières est notre priorité absolue. Bénéficiez de fonctionnalités de sécurité de pointe garantissant la confidentialité et la protection de vos informations.

Personnalisation Avancée

Créez des comptes adaptés à vos besoins, associez des catégories personnalisées, et définissez des objectifs d'épargne alignés sur vos aspirations.

Planification d'Objectifs Financiers

Atteignez vos objectifs financiers sans effort. Que vous économisiez pour des vacances de rêve ou un achat majeur, notre application vous accompagne dans la planification et le suivi de vos progrès.

* Importance de l'architecture microservices :

Dans les architectures de microservices, les applications sont développées de façon à permettre l'exécution indépendante de chaque fonction clé. Ainsi, les équipes de développement peuvent créer et modifier de nouveaux composants pour s'adapter aux besoins métiers changeants sans perturber l'ensemble de l'application.

Les microservices stimulent vos équipes et vos routines grâce à un développement distribué. Vous pouvez aussi développer plusieurs microservices simultanément. Ainsi, davantage de développeurs peuvent travailler en même temps, sur la même application, ce qui réduit la durée du développement.

Haute évolutivité :

À mesure que la demande pour certains services augmente, vous pouvez étendre les déploiements sur plusieurs serveurs et infrastructures pour répondre à vos besoins.

Accessibilité :

Vu que l'application est décomposée en plusieurs éléments, les développeurs peuvent plus facilement comprendre, mettre à jour et améliorer chacun de ces éléments. Résultat : les cycles de développement sont plus courts, surtout s'ils sont associés à des méthodes de développement agiles telles que le DevOps.

Résilience :

Lorsqu'ils sont développés correctement, ces services indépendants n'ont aucun impact les uns sur les autres. Cela signifie que, lorsqu'un élément tombe en panne, l'ensemble de l'application ne cesse pas de fonctionner comme c'est le cas avec le modèle monolithique.

2. Architecture Microservices

* Architecture

A diagram of a software application

Description automatically generated

Dans notre application « wallet », nous avons mis en place une architecture robuste reposant sur trois microservices distincts, à savoir les services utilisateurs, transactions et catégories. Chacun de ces microservices a été conçu pour remplir des fonctions spécifiques, garantissant ainsi une structure modulaire et évolutive.

* Description des services

Le service "Utilisateurs" se concentre sur la gestion et l'authentification des utilisateurs. Il gère les informations relatives aux comptes des utilisateurs, y compris l'inscription, la connexion et la gestion des profils.

Le service "Transactions" est dédié à la gestion des opérations financières. Il enregistre et stocke les données relatives aux transactions quotidiennes des utilisateurs, garantissant un suivi précis des mouvements d'argent au sein de l'application.

Quant au service "Catégories", il offre une structure organisée pour classer les transactions. Il permet aux utilisateurs de catégoriser leurs dépenses et revenus, facilitant ainsi une analyse détaillée des habitudes financières.

* Mécanismes de communication

Pour le mécanisme de communication entre les microservices au sein de notre application, nous avons choisi d'utiliser RestTemplate. RestTemplate est une bibliothèque Java qui facilite les appels RESTful entre différents services. Grâce à RestTemplate, nous avons mis en place une communication transparente et efficace entre les microservices, permettant l'échange sécurisé de données.

RestTemplate offre une abstraction simple pour effectuer des requêtes HTTP, qu'il s'agisse d'envoyer des demandes de lecture de données, de création de nouvelles transactions, ou de mettre à jour des informations utilisateur. En utilisant cette bibliothèque, nous avons simplifié le processus de communication entre les services, tout en bénéficiant de fonctionnalités telles que la gestion automatique des erreurs, la manipulation des en-têtes et la sérialisation/désérialisation des données au format JSON.

3. Conception des Microservices

* Approche de conception pour chaque service

La conception des microservices repose sur une approche modulaire visant à décomposer l'application en composants indépendants, chacun se concentrant sur une fonctionnalité spécifique. Voici l'approche de conception pour chaque service dans notre architecture de portefeuille :

Microservice Utilisateurs :

Gestion des Utilisateurs : Ce service se concentre sur la gestion des comptes utilisateur, y compris l'inscription, la connexion, la mise à jour des profils et la gestion des autorisations.

Authentification et Autorisation : La sécurité est une priorité. Le microservice gère les mécanismes d'authentification, d'autorisation et la protection des données utilisateur.

Microservice Transactions :

Enregistrement des Transactions : Ce service est dédié à l'enregistrement des opérations financières quotidiennes, avec des fonctionnalités telles que l'ajout de nouvelles transactions, la récupération d'historique et la mise à jour des détails.

Analyse Financière : Il offre des fonctionnalités pour analyser les habitudes de dépenses des utilisateurs, générer des rapports et fournir une vue détaillée des transactions.

Microservice Catégories :

Gestion des Catégories : Ce service permet aux utilisateurs de catégoriser leurs dépenses et revenus. Il offre des fonctionnalités pour créer, mettre à jour et supprimer des catégories personnalisées.

Organisation des Données : Il s'occupe de la manière dont les transactions sont classées et affichées dans des catégories, facilitant ainsi l'analyse des habitudes financières.

4. Conteneurisation avec Docker

* Implémentation et avantages

On va afficher les microservices qui sont dans Eureka :

A screenshot of a computer

Description automatically generated

Et Docker :

A screenshot of a computer

Description automatically generated

Donc voila l’implementation de docker pour chaque microservice, et pour ces avantages sont :

**Avantages de Docker :**

Portabilité : Les conteneurs Docker encapsulent une application et ses dépendances, garantissant une portabilité élevée. Un conteneur fonctionnera de la même manière sur n'importe quel environnement compatible Docker, qu'il s'agisse d'un développeur local, d'un serveur de production ou d'un service cloud.

Isolation : Les conteneurs offrent une isolation légère, permettant à plusieurs applications de s'exécuter sur un même hôte sans interférer les unes avec les autres. Chaque conteneur a son propre système de fichiers, processus et espace d'adressage, assurant une séparation efficace.

Facilité de Déploiement : Docker simplifie le processus de déploiement en encapsulant l'application et ses dépendances dans une image. Cela garantit que l'application fonctionnera de la même manière sur tous les environnements, éliminant ainsi les problèmes liés aux variations entre les configurations.

Évolutivité : Grâce à des outils comme Docker Compose ou Kubernetes, il est facile de mettre à l'échelle les applications basées sur Docker. Les conteneurs peuvent être répliqués et orchestrés de manière efficace pour répondre à des besoins de charge variables.

Gestion des Ressources : Docker optimise l'utilisation des ressources en partageant le noyau du système d'exploitation entre les conteneurs. Cela permet d'économiser de l'espace de stockage et de la mémoire, tout en facilitant le déploiement de multiples instances d'une application.