***RAPPORT CARBUTRACK***

***Une image contenant logo, symbole, Graphique, Police

Description générée automatiquement***

***Réalisé par :***

*NASIR Houssam*

*DANNOUN Zakariae*

*EDDIFAY Ismail*

Introduction génerale :

L'essor continu de l'industrie automobile, conjugué à une prise de conscience croissante de l'impact environnemental, souligne l'importance de la gestion efficace de la consommation de carburant dans le cadre de la mobilité moderne. Dans cette perspective, notre projet vise à mettre en place un système complet de gestion de la consommation de carburant et de la flotte de véhicules, offrant des fonctionnalités avancées pour les utilisateurs et permettant une prise de décision éclairée.

Notre application propose une approche intégrée pour suivre la consommation de carburant de chaque véhicule, en mettant l'accent sur la relation entre l'utilisateur, le type de carburant utilisé, la distance parcourue, et les fluctuations du marché du carburant. Pour ce faire, nous avons développé des API robustes qui calculent la consommation de carburant par kilomètre et estiment le coût du carburant en fonction des prix du marché actuels.

Cette plateforme offre une interface conviviale permettant aux utilisateurs de saisir et de consulter les données relatives à leurs véhicules, de visualiser des statistiques détaillées sur la consommation de carburant et de prendre des décisions éclairées en matière de gestion de flotte. Les informations générées par notre application permettent aux utilisateurs de mieux comprendre la performance de leurs véhicules, d'optimiser les coûts liés au carburant et de contribuer à une utilisation plus durable des ressources.

Au cours de ce rapport, nous explorerons en détail les fonctionnalités clés de notre application, les algorithmes sous-jacents de calcul de la consommation de carburant, ainsi que les API permettant d'obtenir des informations actualisées sur les prix du carburant. Nous examinerons également les perspectives d'extension de notre application, les défis rencontrés pendant le développement, et les avantages qu'elle offre en termes de gestion de flotte et de responsabilité environnementale. En somme, notre projet représente une solution complète et innovante pour la gestion intelligente de la consommation de carburant et de la flotte automobile.

Introduction :

## Aperçu du projet :

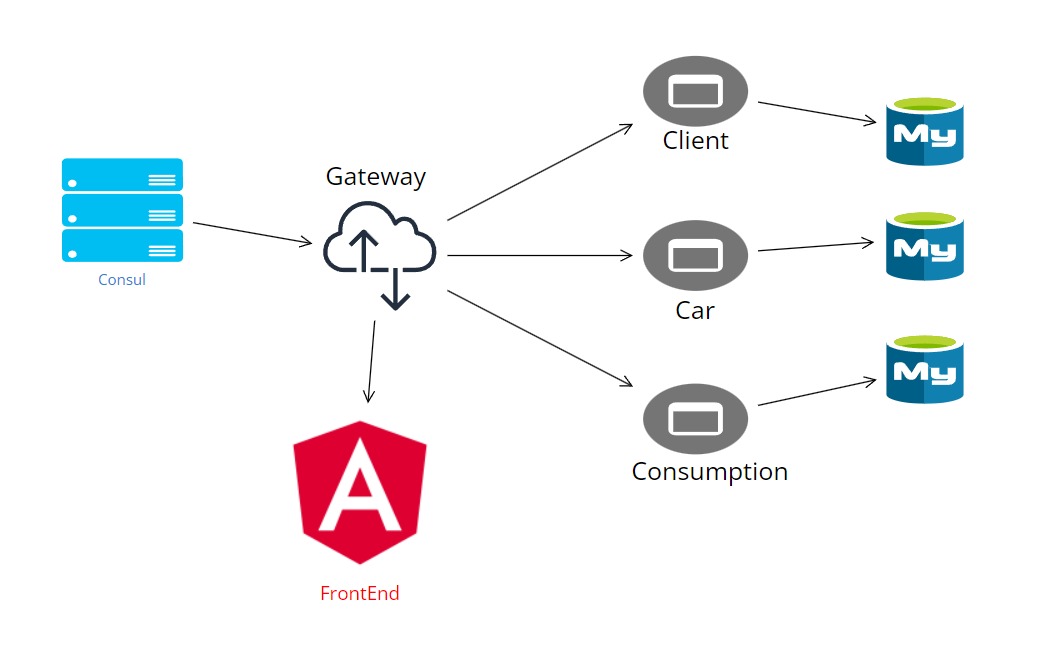
Notre projet de gestion de la consommation de carburant et de la flotte de véhicules se positionne comme une solution holistique pour répondre aux besoins complexes des utilisateurs engagés dans la gestion efficace de leur parc automobile. L'application offre une interface intuitive permettant aux utilisateurs de saisir et de suivre les détails de chaque véhicule, tout en bénéficiant d'outils puissants pour analyser et optimiser la consommation de carburant.

## Importance de l’architecture Microservices :

L'architecture microservices a été choisie pour ce projet en raison de ses nombreux avantages. Elle favorise la scalabilité, la flexibilité et la facilité de déploiement. Chaque microservice est indépendant, ce qui permet une évolution et une maintenance aisées. De plus, cette architecture facilite la gestion des différentes fonctionnalités de la plateforme.

Architecture Microservices :

## Architecture Microservices :



Une image contenant texte, capture d’écran, conception

Description générée automatiquement

## Description des services :

**Service Client :**

Description : Le service client est probablement responsable de gérer les informations relatives aux utilisateurs ou clients de votre application. Cela peut inclure la création de profils, la gestion des authentifications, et la sauvegarde des préférences utilisateur.

Responsabilités possibles : Enregistrement des utilisateurs, authentification, gestion des profils, suivi des préférences utilisateur.

**Service de Consommation :**

Description : Le service de consommation semble être impliqué dans le suivi et la gestion de la consommation, probablement de carburant dans votre cas. Il pourrait être responsable du stockage des données relatives à la consommation de carburant pour chaque véhicule.

Responsabilités possibles : Enregistrement des données de consommation, calcul de la consommation par kilomètre, suivi des tendances de consommation, génération de rapports.

**Passerelle (Gateway) :**

Description : La passerelle (gateway) est un composant essentiel dans une architecture basée sur microservices. Elle agit comme un point d'entrée pour les requêtes provenant des clients et dirige ces requêtes vers les services appropriés en utilisant des règles de routage.

Responsabilités possibles : Routage des requêtes vers les services appropriés, gestion des requêtes HTTP, potentiellement la gestion de la sécurité, collecte de métriques ou de journaux.

**Consul :**

Description : Consul est un système de découverte de services et de gestion de la configuration. Il permet aux microservices de s'enregistrer et de découvrir les autres services disponibles dans l'écosystème. Il assure également la gestion des configurations partagées entre les services.

Responsabilités possibles : Découverte de services, enregistrement des services, gestion de la configuration, gestion des clés-valeurs partagées.

## Mécanisme de communication :

Pour la communication entre les services au sein de notre architecture microservices, j'ai opté pour l'utilisation de RestTemplate. Cette solution, intégrée au sein du framework Spring, a simplifié considérablement les échanges HTTP entre les différents composants de notre application. Grâce à RestTemplate, j'ai pu établir des requêtes HTTP de manière transparente, facilitant ainsi la communication entre les services tout en bénéficiant de la modularité offerte par cette bibliothèque. Cette approche a contribué à créer une infrastructure de communication robuste, tout en préservant la simplicité et la flexibilité nécessaires dans un environnement distribué. En utilisant RestTemplate, nous avons réussi à mettre en œuvre une solution efficace et élégante pour répondre à nos besoins de communication inter-services.

Conception des Microservices :

## Approche de conception de chaque service :

**Service Client :**

Modélisation des Données : Définition un modèle de données pour les informations utilisateur, y compris les détails du profil, les informations d'authentification, et les préférences.

Fonctionnalités : Implémentation des fonctionnalités telles que l'enregistrement des utilisateurs, l'authentification, la gestion des profils, et la gestion des préférences utilisateur.

Sécurité : Mettere en place des mécanismes de sécurité robustes.

**Service de Consommation :**

Modélisation des Données : Définition un modèle de données pour les informations de consommation, y compris les détails spécifiques à chaque véhicule, les données de consommation de carburant, et les statistiques associées.

Calcul de la Consommation : Implémentation des algorithmes de calcul de la consommation de carburant par kilomètre parcouru.

Fonctionnalités : Intégration des fonctionnalités d'enregistrement des données de consommation, de suivi des tendances, et de génération de rapports.

**Service Car :**

Modélisation des Données : Définition un modèle de données pour les informations spécifiques aux véhicules, telles que le modèle de voiture, le type de carburant utilisé, et d'autres caractéristiques.

Fonctionnalités : Implémentation des fonctionnalités d'enregistrement des détails du véhicule, de gestion des caractéristiques spécifiques, et d'intégration avec d'autres services, comme le service de consommation.

**Passerelle (Gateway) :**

Gestion des Requêtes : Concevez la passerelle pour router les requêtes entrantes vers les services appropriés en utilisant des règles de routage.

Sécurité : Mettez en place des mécanismes de sécurité au niveau de la passerelle, tels que la gestion des tokens, la validation des requêtes, et la protection contre les attaques potentielles.

Surveillance : Implémentation des mécanismes de suivi et de surveillance pour collecter des métriques et des journaux liés aux requêtes entrantes et sortantes.

Conteneurisation avec Docker :

## Implémentation :

Lors de la dockerisation d'une application, le processus implique la création d'un fichier Dockerfile qui décrit comment construire une image Docker pour chaque service, suivi de la création d'un fichier docker-compose.yml qui orchestre le déploiement et l'interaction entre ces services.

1. Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

   Description générée automatiquementDocker file
2. Docker compose :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

# Avantages :

* Portabilité :

Les conteneurs Docker encapsulent toutes les dépendances et configurations nécessaires, assurant ainsi la portabilité entre différents environnements.

* Gestion Simplifiée des Dépendances :

Docker permet d'isoler les dépendances de chaque microservice, évitant les conflits et simplifiant la gestion des dépendances.

* Facilité de Déploiement :

La conteneurisation simplifie le déploiement en garantissant que chaque microservice fonctionne dans son propre conteneur, indépendamment des autres.

* Mise à l'Échelle Efficace :

La nature légère des conteneurs Docker permet une mise à l'échelle efficace en ajoutant ou en retirant des instances de microservices selon les besoins.

* Répétabilité des Environnements :

Les conteneurs Docker garantissent que chaque environnement, du développement à la production, est cohérent et répétable.

CI/CD avec Jenkins :

## Processus de configuration :

Le processus CI/CD est mis en place à l'aide de Jenkins, automatisant le processus de construction, de test et de déploiement. Cela garantit une intégration continue et une livraison rapide des nouvelles fonctionnalités.

Une image contenant texte, logiciel, Icône d’ordinateur, Page web

Description générée automatiquement

Déploiement automatique :

## Utilisation de Ngrok :

L'intégration de Ngrok facilite le déploiement automatisé et fiable des microservices, garantissant une disponibilité constante de la plateforme. Cette approche devient particulièrement avantageuse lorsqu'elle est conjuguée à l'utilisation de webhooks. Les webhooks, des mécanismes permettant à un système de notifier automatiquement un autre système lorsqu'un événement particulier se produit, bénéficient grandement de l'utilisation de Ngrok.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Integration de Sonarqube :

## Pipline :

Une image contenant texte, capture d’écran, nombre, logiciel

Description générée automatiquement

Une image contenant texte, capture d’écran

Description générée automatiquement

Une image contenant texte, ligne, Tracé, capture d’écran

Description générée automatiquement

Conclusion :

## Résumé des accomplissements :

Le projet a abouti à la création d'une plateforme de gestion de consomation de carburant, basée sur une architecture microservices. L'utilisation de Docker, Jenkins et SonarQube a permis d'assurer une mise en œuvre efficace, une intégration et déploiement continue et une qualité de code élevée.

## Perspective future :

Les perspectives futures du projet incluent plusieurs axes de développement pour améliorer encore la performance et la fonctionnalité de l'application. Une extension envisageable pourrait consister à intégrer des fonctionnalités de maintenance prédictive, permettant ainsi d'anticiper les besoins de service et d'optimiser la durée de vie des véhicules.