YOUR CAR YOUR WAY

SOMMAIRE

Description générale de l’architecture

L'application "Your Car Your Way" est conçue selon une architecture moderne basée sur les principes de Domain-Driven Design (DDD) et orientée microservices.
Cette approche garantit une modularité, une évolutivité et une indépendance des composants pour répondre aux besoins métier et techniques de manière flexible.

Composants principaux

Couche Client :

- Hébergée sur **Amazon CloudFront** pour une distribution rapide des fichiers statiques (frontend **Angular**).

- Les interactions utilisateur sont sécurisées et routées via un Application Load Balancer (**ALB**).

Couche Backend :

- Implémentée sous forme de microservices conteneurisés déployés sur **Amazon ECS**. Chaque microservice gère un domaine fonctionnel spécifique comme la gestion des utilisateurs, des réservations, ou des paiements.

- Les communications asynchrones entre microservices sont orchestrées via **Amazon EventBridge**.

Couche Données :

- Données sensibles et métier (utilisateurs, véhicules, réservations) stockées dans une base relationnelle **PostgreSQL** sur **Amazon RDS**, avec séparation stricte par domaine pour ne pas fortement coupler les services.

Liste de contrôle de l’architecture terminée

Composants logiciels

- [ ] Le service de gestion des utilisateurs permet la création, modification et suppression des comptes avec les champs obligatoires (nom, email, mot de passe, etc.).
- [ ] L'authentification des utilisateurs est sécurisée avec des tokens JWT.
- [ ] L’interface Angular utilise efficacement les API REST pour la communication authentifiée avec le backend.
- [ ] Les mots de passe des utilisateurs sont chiffrés de manière sécurisée.
- [ ] La gestion des profils utilisateur respecte les règles de suppression (aucune réservation en cours).
- [ ] Les notifications via Amazon EventBridge fonctionnent pour tous les événements métier.
- [ ] Les connexions via WebSocket utilisent un certificat SSL/TLS valide.

La pipeline CI/CD est configuré pour automatiser les étapes suivantes :

- [ ] Compilation et validation du code source.
- [ ] Exécution des tests unitaires et d’intégration pour chaque microservice.
- [ ] Le frontend est développé avec Angular 17.
- [ ] Les microservices sont développés avec SpringBoot et Java 21.
- [ ] Analyse continue de la qualité du code via SonarCloud pour détecter les vulnérabilités et assurer la conformité aux normes de codage.
- [ ] Packaging et déploiement des artefacts Docker sur un registre sécurisé (ECR).

- [ ] Les tests end-to-end (E2E) sont automatisés avec Cypress entre le frontend Angular et les microservices backend.

- [ ] Chaque étape du pipeline (build, test, analyse) bloque le déploiement si des erreurs ou des régressions sont détectées.
- [ ] Les mises à jour des microservices utilisent une approche sans interruption grâce à la scalabilité des conteneurs sur Amazon ECS.

Le microservice "Location" applique correctement les règles métier pour :

- [ ] Filtrage dynamique des véhicules disponibles (dates, catégories, disponibilité).

- [ ] Calcul automatique des frais de transfert inter-agences.

- [ ] Gestion des modifications de réservations (dates, agences, options).

Le microservice "Paiement" gère :

- [ ] Les sessions Stripe pour les paiements sans stocker d'informations sensibles.

- [ ] Les remboursements partiels ou totaux suivant les politiques d'annulation.

- [ ] La notification en temps réel des statuts de paiement via Amazon EventBridge.

Le service de support permet :

- [ ] Une interaction en temps réel via WebSocket.

- [ ] La soumission et le suivi des demandes via email.

- [ ] Les performances des recherches (offres de location) répondent à l'exigence (< 2 secondes par requête).

Les tests unitaires et d’intégration couvrent :

- [ ] Les cas limites pour chaque fonctionnalité (création, modification, suppression).

- [ ] Les scénarios identifiés dans les cas d’utilisation.

Composants tiers

- [ ] Stripe est correctement configuré pour gérer les paiements en conformité avec PCI-DSS.

L'intégration Stripe assure :

- [ ] Une redirection fluide vers l'interface utilisateur pour la finalisation des paiements.
- [ ] Une gestion automatique des échecs et des annulations.
- [ ] Les connexions via WebSocket utilisent un certificat SSL/TLS valide.
- [ ] L'interface Angular utilise efficacement les API REST pour la communication avec le backend.

Gestion des données

- [ ] Les données personnelles des utilisateurs (noms, emails, mots de passe) sont traitées conformément au RGPD.

- [ ] La conformité au PCI-DSS est assurée pour toutes les transactions financières via l'intégration de Stripe. Aucune donnée de carte bancaire n'est stockée dans le système.

- [ ] Toutes les données sensibles (mots de passe, tokens JWT) sont stockées de manière sécurisée, avec des mécanismes de chiffrement robustes (ex. : AES-256).

- [ ] Les bases de données relationnelles (PostgreSQL sur Amazon RDS) sont configurées pour segmenter les données par domaine métier pour garantir une isolation stricte entre les microservices.

- [ ] Les données utilisateur et métier (réservations, véhicules, paiements) sont validées au niveau applicatif pour assurer leur intégrité et leur cohérence dans la base de données.

- [ ] Les données personnelles des utilisateurs sont supprimées conformément aux politiques de rétention établies et aux demandes de suppression des comptes.

- [ ] Les historiques des transactions financières (sessions) sont conservés conformément aux exigences légales.

- [ ] Les requêtes sur les bases de données sont optimisées pour garantir des temps de réponse rapides (< 2 secondes pour les recherches d'offres).

Infrastructure

- [ ] L’infrastructure est hébergée sur AWS, respectant une architecture cloud-native avec des services managés pour garantir la résilience et la scalabilité
- [ ] Les fichiers statiques du frontend Angular sont hébergés sur Amazon S3 et distribués via Amazon CloudFront
- [ ] Les interactions utilisateur (API ou WebSocket) sont sécurisées et routées via un Application Load Balancer (ALB)
- [ ] Les microservices backend sont conteneurisés et déployés sur Amazon ECS
- [ ] Les données sensibles et métier (utilisateurs, véhicules, réservations) sont stockées dans une base relationnelle PostgreSQL hébergée sur Amazon RDS, avec des principes stricts de séparation des données par domaine
- [ ] La communication entre microservices est orchestrée via Amazon EventBridge, permettant une gestion asynchrone des événements métier
- [ ] Les métriques système (utilisation CPU, erreurs HTTP, etc.) et journaux de log des microservices sont surveillés via Amazon CloudWatch
- [ ] Les connexions WebSocket pour le support en temps réel sont gérées via ALB

Sécurité

- ☐ Les communications entre la Couche client, la Couche backend et les bases de données dans la Couche données sont sécurisées via TLS/SSL
- ☐ Les tokens JWT sont utilisés pour l'authentification et l'autorisation des utilisateurs. Les JWT contiennent des informations essentielles et sont validés par les microservices en utilisant une clé publique stockée dans AWS Parameter Store
- ☐ La sécurisation des connexions WebSocket pour le support en temps réel est assurée via SSL/TLS, en utilisant des connexions wss:// routées par un Application Load Balancer (ALB)
- ☐ Les informations sensibles, y compris les données des utilisateurs, des véhicules et des réservations, sont stockées dans une base relationnelle PostgreSQL sur Amazon RDS
- ☐ Les mots de passes sont chiffrés avec Bcrypt
- ☐ La communication entre les microservices est orchestrée via Amazon EventBridge, garantissant une transmission sécurisée et asynchrone des événements métier
- ☐ Les logs d'accès et d'activité des services sont surveillés via Amazon CloudWatch, permettant de détecter les anomalies