République du CAMEROUN
****
Paix – Travail – Patrie
******
Faculté des Sciences
***
Département d'informatique

Republic of CAMEROUN
****
Peace – Work – Fatherland
******
Faculty of Sciences
***
Computer Science Department

UE : INF 462

TP 2: Microservices Fundamentals_ Build API

GROUPE 5

MOUSSA ABAKAR ABBAZENE	**23V2834**
ABDELAZIZ MAHAMAT LOUKY	**18T2916**
SEMDI HONORE	**14L1992**
MESSI DZOU SYLVESTRE CEDRIC	**23V2853**

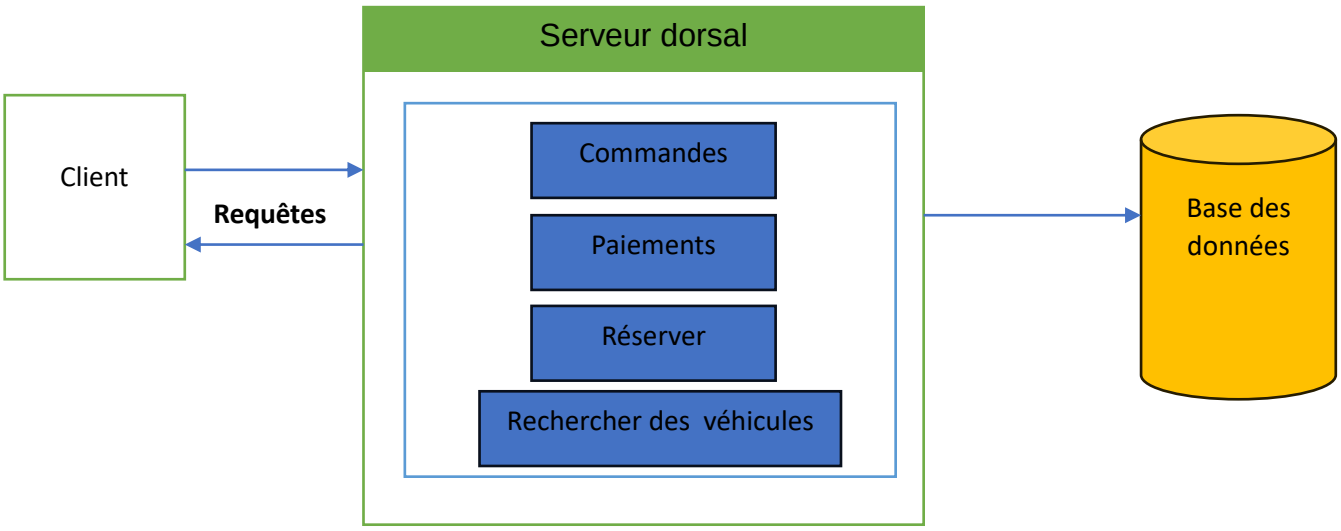
Enseignant : Dr. KIMBI XAVIERA

Sous la supervision de : M. ATEMENGUE REGIS

Année académique : 2023-2024

Taches :

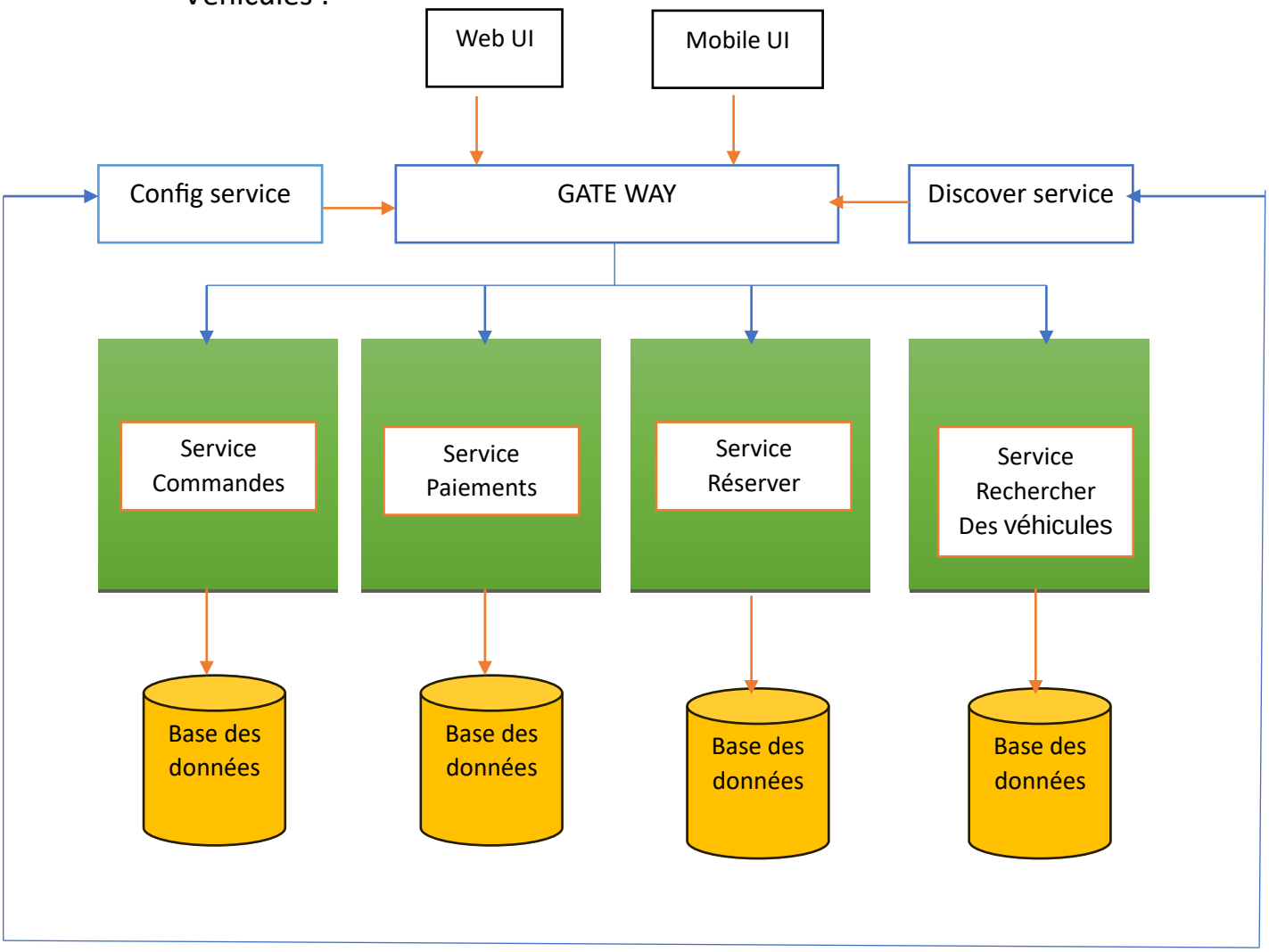
1. Représentons l'ancienne architecture de l'application de véhicule de propelize :



2. Donnons les inconvénients de l'architecture monolithique de propelize :

- **Complexité accrue au fur et à mesure de la croissance de l'application** : Avec le temps, la base de code devient de plus en plus complexe, ce qui rend difficile la compréhension et la maintenance du système.
- **Difficulté à apporter des modifications** : En raison de l'interconnexion étroite des composants, il devient difficile d'apporter des modifications à l'application sans perturber d'autres parties du système.
- **Cycles de construction et de déploiement plus longs** : En raison de la taille de l'application, les cycles de construction et de déploiement sont souvent plus longs, ce qui ralentit le processus de développement.
- **Difficulté d'intégration de nouvelles technologies** : Les monolithes sont étroitement liés à une pile technologique spécifique, ce qui rend difficile l'intégration de nouvelles technologies ou la mise à jour des technologies existantes.
- **Faible fiabilité** : Un bogue ou un problème dans une partie du système peut potentiellement faire tomber l'ensemble de l'application, entraînant des pannes importantes du système.
- **Maintenance difficile** : En raison de la nature complexe et enchevêtrée de l'architecture monolithique, la maintenance de l'application devient difficile à mesure que la base de code grossit et évolue.

3. Proposons une architecture micro service de l'application de location de Véhicules :



4. Choisissons un stack technologique pour l'implémentation du véhicule service et justifiez le choix de votre stack.

Pour l'implémentation du service de gestion des véhicules chez Propelize, je recommande l'utilisation du stack technologique suivant :

- **Spring Boot** : Spring Boot est un framework Java largement utilisé pour le développement de microservices. Il offre une configuration automatique, une prise en charge intégrée des conteneurs et une architecture modulaire qui facilite la création rapide de microservices robustes. Avec ses fonctionnalités telles que Spring Data et Spring MVC, Spring Boot permet de développer facilement des services RESTful et d'intégrer efficacement d'autres composants du système.

- **MySQL** : MySQL est une base de données relationnelle bien établie et largement utilisée. Elle offre une performance fiable, une stabilité et une facilité d'utilisation, ce qui en fait un choix populaire pour les applications d'entreprise. Avec Spring Boot, l'intégration avec MySQL est transparente grâce à Spring Data JPA, ce qui simplifie le développement et la gestion des requêtes SQL.
- **Docker** : Docker est une technologie de conteneurisation qui permet d'encapsuler les microservices et leurs dépendances dans des conteneurs légers et portables. En utilisant Docker, Propelize peut simplifier le déploiement, la gestion et la scalabilité de ses microservices, tout en assurant la cohérence des environnements de développement, de test et de production.
- **Kubernetes** : Kubernetes offre une plateforme d'orchestration de conteneurs puissante pour le déploiement et la gestion des microservices à grande échelle. En combinant Spring Boot avec Kubernetes, Propelize peut bénéficier d'une gestion automatisée des conteneurs, d'une scalabilité horizontale et d'une haute disponibilité, ce qui est essentiel pour les applications d'entreprise à forte charge comme la gestion des véhicules.

En choisissant ce stack technologique, Propelize bénéficiera d'une architecture solide et évolutive pour son service de gestion des véhicules, avec des outils bien établis et une intégration transparente entre les différents composants. De plus, l'utilisation de Spring Boot et MySQL permettra une transition en douceur pour les développeurs Java avec une expertise préexistante dans ces technologies.

5. Construire l'API de gestion des véhicules avec votre technologie :

Ce déjà fait (voir code source).

6. Produisons un document qui décrit le processus de ce problème :

<u>Besoins fonctionnels de Propelize :</u>

- **Gestion des véhicules** : Permettre aux utilisateurs de rechercher, réserver et gérer des véhicules disponibles à la location, ainsi que de visualiser les détails des véhicules.

- **Gestion des utilisateurs** : Permettre aux utilisateurs de créer des comptes, de se connecter, de gérer leurs informations personnelles et leurs réservations.

- **Gestion des réservations** : Permettre aux utilisateurs de faire des réservations de véhicules, de choisir les dates et les heures de location, de visualiser les détails de leurs réservations, et de pouvoir les annuler ou les modifier.

- **Gestion des paiements**: Permettre aux utilisateurs d'effectuer des paiements en ligne sécurisés pour leurs réservations de véhicules et de recevoir des confirmations de paiement.

Besoins non fonctionnels de Propelize :

- **Performance** : Assurer des temps de réponse rapides et une faible latence de l'application, même lorsqu'il y a un grand nombre d'utilisateurs.

- **Sécurité** : Garantir la sécurité des données des utilisateurs, des transactions financières et de l'application dans son ensemble.

- **Disponibilité**: Assurer une disponibilité élevée de l'application avec un minimum de temps d'arrêt planifiés et non planifiés.

- **Scalabilité** : Concevoir l'architecture de l'application pour qu'elle puisse évoluer facilement pour gérer une augmentation du nombre d'utilisateurs et de réservations.

- **Convivialité** : Offrir une expérience utilisateur intuitive et conviviale sur l'application web et mobile pour faciliter la recherche, la réservation et le paiement des véhicules.

- **Interopérabilité:** Assurer l'interopérabilité de l'application avec d'autres systèmes ou services externes, tels que les fournisseurs de paiement et les partenaires de location de véhicules.

Identification de la cible l'application

Dans le contexte de Propelize et de son activité de location de véhicules, l'identification de la cible de l'application peut se faire en définissant les principaux utilisateurs et parties prenantes qui interagiront avec le système. Voici une liste des principales cibles de l'application :

- **Clients individuels** : Les personnes qui utilisent l'application pour rechercher, réserver et gérer la location de véhicules pour leurs besoins personnels, tels que les déplacements quotidiens, les voyages ou les déménagements.
- **Entreprises et professionnels** : Les entreprises ou les professionnels qui utilisent l'application pour la location de véhicules à des fins professionnelles, telles que le transport de marchandises, les services de livraison ou les déplacements professionnels.
- **Administrateurs du système** : Les membres du personnel de Propelize responsables de la gestion et de la maintenance de l'application, y compris la gestion des utilisateurs, des véhicules, des réservations et des transactions.
- **Partenaires de location de véhicules** : Les entreprises ou les agences de location de véhicules qui collaborent avec Propelize en fournissant des véhicules à louer via l'application.

Etablissement des objectifs de cette application

Pour l'application de location de véhicules de Propelize, les objectifs peuvent être définis pour répondre aux besoins des utilisateurs tout en soutenant la croissance et le succès de l'entreprise. Voici quelques objectifs clés pour cette application :

- **Moderniser l'expérience utilisateur** : L'objectif principal est de proposer une application web et mobile moderne qui permettra aux clients de rechercher, réserver et gérer facilement la location de véhicules.

- **Améliorer l'accessibilité et la disponibilité** : L'application doit être facilement accessible à partir de différents appareils, offrant ainsi aux clients la possibilité de trouver rapidement des véhicules disponibles à la location à tout moment et depuis n'importe où.

- **Assurer la sécurité des transactions**: L'objectif est de garantir que les transactions financières effectuées via l'application sont sécurisées, afin de protéger les informations personnelles des utilisateurs et de renforcer la confiance dans le système.

- **Optimiser les processus de réservation :** L'application doit simplifier le processus de réservation en offrant aux clients une interface conviviale pour rechercher des véhicules, choisir les dates de location et effectuer des réservations en quelques clics.

- **Faciliter la gestion des réservations**: Permettre aux utilisateurs de gérer leurs réservations, y compris la modification ou l'annulation des réservations existantes, afin de répondre à leurs besoins changeants.

- **Intégrer des fonctionnalités de notification** : L'application doit fournir des notifications aux utilisateurs pour confirmer leurs réservations, leur rappeler les dates de retour des véhicules et leur fournir des mises à jour pertinentes sur leurs réservations.

- **Favoriser l'évolutivité et la flexibilité** : Concevoir une architecture flexible et évolutive qui permettra à l'application de s'adapter à la croissance de l'entreprise et d'intégrer de nouvelles fonctionnalités à mesure que les besoins évoluent.

Identification des Ressources et des actions

Pour l'application de location de véhicules de Propelize, l'identification des ressources et des actions implique de déterminer les principaux éléments qui seront utilisés dans le système et les actions qui peuvent être effectuées par les utilisateurs. Voici une liste des ressources et des actions clés :

Ressources :

- **Véhicules** : Les différentes voitures et camionnettes disponibles à la location, avec leurs caractéristiques telles que le modèle, la marque, la capacité, etc.

- **Utilisateurs** : Les clients qui utilisent l'application pour rechercher, réserver et gérer la location de véhicules. Cela inclut les informations telles que le nom, l'adresse e-mail, les informations de paiement, etc.

- **Réservations** : Les réservations de véhicules effectuées par les utilisateurs, comprenant des détails tels que les dates de location, les heures, les informations sur le véhicule réservé, etc.

- **Paiements** : Les transactions financières liées aux réservations de véhicules, y compris les détails des paiements effectués par les utilisateurs.

Actions :

- **Recherche de véhicules** : Les utilisateurs peuvent rechercher des véhicules disponibles en spécifiant des critères tels que la date, l'heure et le lieu de prise en charge et de retour.

- **Réservation de véhicules** : Les utilisateurs peuvent réserver un véhicule disponible en sélectionnant les dates de location et en effectuant le paiement.

- **Gestion des réservations** : Les utilisateurs peuvent consulter, modifier ou annuler leurs réservations existantes.

- **Paiement en ligne** : Les utilisateurs peuvent effectuer des paiements en ligne sécurisés pour leurs réservations de véhicules.

Le choix des designs Principles (REST, GraphQL, SOAP)

Nous allons choisir REST comme design principle pour l'architecture de l'application de location de véhicules de Propelize, en raison de :

- **Simplicité et universalité** : REST (Representational State Transfer) est basé sur des principes simples et largement compris, ce qui facilite son adoption par les développeurs et son intégration avec différents frameworks et langages de programmation.

- **Flexibilité** : REST offre une grande flexibilité en termes de manipulation des ressources via les opérations HTTP standard (GET, POST, PUT, DELETE), ce qui permet de concevoir des API facilement extensibles et évolutives.

- **Séparation des préoccupations** : REST favorise la séparation claire entre les clients et les serveurs, ce qui permet de concevoir des systèmes modulaires et évolutifs où les changements du côté client ou serveur peuvent être effectués indépendamment.

- **Interopérabilité**: Les API REST sont basées sur des standards HTTP ou HTTPS, ce qui les rend compatibles avec une large gamme de plateformes et de dispositifs, facilitant ainsi l'interopérabilité entre les différents systèmes.

- **Performance**: En utilisant les méthodes HTTP standard, REST permet une interaction efficace entre les clients et les serveurs, ce qui contribue à des performances optimales de l'application.

- **Évolutivité**: Grâce à sa flexibilité et à sa conception orientée ressources, les architectures REST sont naturellement évolutives, permettant de répondre aux besoins changeants de l'application et de prendre en charge un nombre croissant d'utilisateurs et de fonctionnalités.

La définition des EndPoints

La définition des Endpoints dans l'architecture REST de l'application de location de véhicules de Propelize consiste à identifier les URL (Uniform Resource Locators) qui représentent les ressources manipulées par l'API. Chaque endpoint correspond à une action spécifique sur une ressource et est associé à une méthode HTTP appropriée (GET, POST, PUT, DELETE) pour effectuer cette action. Voici quelques exemples d'endpoints pour notre application :

- **GET /vehicules**: Récupérer la liste de tous les véhicules disponibles à la location.
- **GET /vehicules/{id}** : Récupérer les détails d'un véhicule spécifique en fonction de son identifiant.
- **POST /reservations** : Créer une nouvelle réservation de véhicule.
- **GET /reservations/{id}** : Récupérer les détails d'une réservation spécifique en fonction de son identifiant.
- **PUT /reservations/{id}** : Mettre à jour une réservation existante.
- **DELETE /reservations/{id}** : Supprimer une réservation existante.

Ces endpoints représentent les principales actions que les utilisateurs peuvent effectuer sur les ressources de l'application, telles que les véhicules et les réservations. Chaque endpoint est conçu pour être cohérent, intuitif et conforme aux principes REST, facilitant ainsi l'utilisation et l'intégration de l'API par les clients de l'application.

Définition des formats des Requêtes et des réponses

Dans l'architecture REST de l'application de location de véhicules de Propelize, les formats des requêtes et des réponses sont définis selon les standards HTTP et les conventions RESTful. Voici une définition de ces formats :

Format des requêtes :

- **Méthodes HTTP** : Les requêtes utilisent les méthodes HTTP standard telles que GET, POST, PUT et DELETE pour indiquer l'action à effectuer sur la ressource. Par exemple, GET est utilisé pour récupérer des données, POST pour créer de nouvelles ressources, PUT pour mettre à jour des ressources existantes, et DELETE pour supprimer des ressources.

- **En-têtes de requête**: Les en-têtes de requête contiennent des métadonnées sur la requête, telles que les informations d'authentification, les types de contenu acceptables, les informations sur la langue, etc.

- **Corps de la requête** : Le corps de la requête contient les données envoyées par le client au serveur. Les données peuvent être au format JSON, XML ou d'autres formats, en fonction des besoins de l'application.

Format des réponses :

- **Codes de statut HTTP** : Les réponses HTTP sont accompagnées de codes de statut qui indiquent le résultat de la requête. Par exemple, 200 OK indique que la requête a réussi, 404 Not Found indique que la ressource demandée n'a pas été trouvée, etc.

- **En-têtes de réponse** : Les en-têtes de réponse contiennent des métadonnées sur la réponse, telles que le type de contenu, la longueur du contenu, les informations de cache, etc.

- **Corps de la réponse** : Le corps de la réponse contient les données renvoyées par le serveur au client. Les données peuvent être au format JSON, XML, HTML ou d'autres formats, en fonction du type de contenu demandé par le client et des besoins de l'application.

En respectant ces standards et ces conventions, Propelize peut concevoir une API RESTful cohérente et interopérable pour son application de location de véhicules, permettant une communication efficace entre le client et le serveur.

Authentification

Dans le contexte de l'application de location de véhicules de Propelize, les considérations sur la sécurité sont essentielles pour protéger les données des utilisateurs, garantir l'intégrité du système et maintenir la confiance des clients. L'une des principales composantes de la sécurité est l'authentification, qui permet de vérifier l'identité des utilisateurs et de contrôler leur accès aux ressources de l'application. Voici quelques considérations sur l'authentification dans cette application :

- **Authentification des utilisateurs** : Les utilisateurs doivent s'authentifier avant de pouvoir accéder à certaines fonctionnalités de l'application, telles que la recherche de véhicules, la réservation et la gestion des réservations. Cela peut être réalisé en demandant aux utilisateurs de fournir des informations d'identification telles que leur nom d'utilisateur et leur mot de passe.

- **Méthodes d'authentification sécurisées** : Il est important d'utiliser des méthodes d'authentification sécurisées telles que OAuth 2.0 ou JWT (JSON Web Tokens) pour garantir la confidentialité des informations d'identification des utilisateurs lors de la transmission et pour éviter les attaques de type "brute force" ou de falsification d'identité.

- **Gestion des sessions** : Une fois qu'un utilisateur est authentifié, une session doit être établie pour maintenir l'état de l'authentification tout au long de son interaction avec l'application. Cela peut être réalisé en utilisant des cookies sécurisés ou des tokens de session.

- **Expiration des sessions** : Les sessions doivent avoir une durée limitée pour des raisons de sécurité. Il est recommandé de définir une durée d'expiration raisonnable pour les sessions afin de limiter les risques d'usurpation de session.

- **Suivi des activités** : Il est important de mettre en place des mécanismes de journalisation et de suivi des activités d'authentification pour détecter les tentatives d'accès non autorisées ou les comportements suspects.

En mettant en œuvre ces considérations sur l'authentification, Propelize peut renforcer la sécurité de son application de location de véhicules et offrir une expérience utilisateur sécurisée et fiable.

Authorization

Pour garantir la sécurité de l'application de location de véhicules de Propelize, il est essentiel de mettre en place des mécanismes d'autorisation solides. L'autorisation détermine les actions spécifiques qu'un utilisateur authentifié est autorisé à effectuer sur les ressources de l'application. Voici quelques considérations sur l'autorisation :

- **Définition des rôles et des permissions** : Identifiez les différents rôles d'utilisateur dans l'application, tels que les utilisateurs normaux, les administrateurs, les gestionnaires, etc. Déterminez les permissions associées à chaque rôle, par exemple, un administrateur peut avoir le droit de créer, modifier ou supprimer des véhicules, tandis qu'un utilisateur normal peut uniquement avoir le droit de consulter les véhicules disponibles.

- **Attribution des rôles aux utilisateurs** : Une fois les rôles définis, attribuez-les aux utilisateurs en fonction de leurs responsabilités et de leurs privilèges dans l'application. Ceci peut être réalisé lors de la création du compte utilisateur ou en fonction des actions effectuées par l'utilisateur.

- **Contrôle d'accès basé sur les rôles (RBAC)** : Mettez en place un contrôle d'accès basé sur les rôles (Role-Based Access Control) pour restreindre l'accès aux ressources de l'application en fonction des rôles attribués aux utilisateurs. Assurez-vous que chaque requête est vérifiée pour les autorisations appropriées avant d'autoriser l'accès à la ressource demandée.

- **Gestion fine des autorisations** : En plus des contrôles d'accès basés sur les rôles, envisagez d'implémenter des autorisations spécifiques au niveau des ressources ou des actions. Par exemple, vous pouvez définir des autorisations granulaires pour les opérations CRUD (Create, Read, Update, Delete) sur les véhicules, permettant un contrôle précis sur les actions autorisées pour chaque utilisateur.

- **Revocation des autorisations** : Prévoyez des mécanismes pour révoquer rapidement les autorisations d'accès en cas de besoin, par exemple, en cas de départ d'un employé ou de changement de rôle d'utilisateur. Assurez-vous que les autorisations sont mises à jour en temps réel pour refléter les changements dans les privilèges des utilisateurs.

- **Audit des autorisations** : Mettez en place des mécanismes de suivi et d'audit pour enregistrer les activités liées à l'autorisation, telles que les tentatives d'accès non autorisées ou les changements dans les autorisations des utilisateurs. Les journaux d'audit peuvent être utilisés pour détecter les violations de sécurité et pour assurer la conformité aux politiques de sécurité de l'entreprise.

En mettant en œuvre ces considérations sur l'autorisation, Propelize peut garantir un contrôle d'accès sécurisé et approprié à ses ressources, protégeant ainsi la confidentialité des données et prévenant les accès non autorisés à l'application.

Compliance and data protection

Dans le cadre de l'application de location de véhicules de Propelize, la conformité et la protection des données sont des aspects cruciaux de la sécurité. Voici quelques considérations à prendre en compte :

- **Conformité aux réglementations en matière de protection des données** : Assurez-vous que l'application est conforme aux réglementations locales et internationales en matière de protection des données, telles que le RGPD (Règlement Général sur la Protection des Données) en Europe ou le CCPA (California Consumer Privacy Act) en Californie. Cela implique de garantir que les données des utilisateurs sont collectées, stockées et traitées de manière légale et éthique.

- **Protection des données sensibles** : Identifiez les données sensibles collectées par l'application, telles que les informations personnelles des utilisateurs (nom, adresse, numéro de téléphone), les données de paiement et les données de localisation. Mettez en place des mesures de sécurité appropriées pour protéger ces données contre les accès non autorisés, la divulgation et la manipulation.

- **Chiffrement des données** : Utilisez le chiffrement pour sécuriser les données sensibles pendant le stockage et la transmission. Le chiffrement des données garantit que même si des données sont interceptées, elles restent illisibles sans la clé de déchiffrement appropriée.

- **Gestion des accès et des autorisations** : Mettez en place des contrôles d'accès et des autorisations stricts pour limiter l'accès aux données sensibles aux seuls utilisateurs autorisés. Utilisez des mécanismes d'authentification forte pour vérifier l'identité des utilisateurs avant de leur accorder l'accès à des données sensibles.

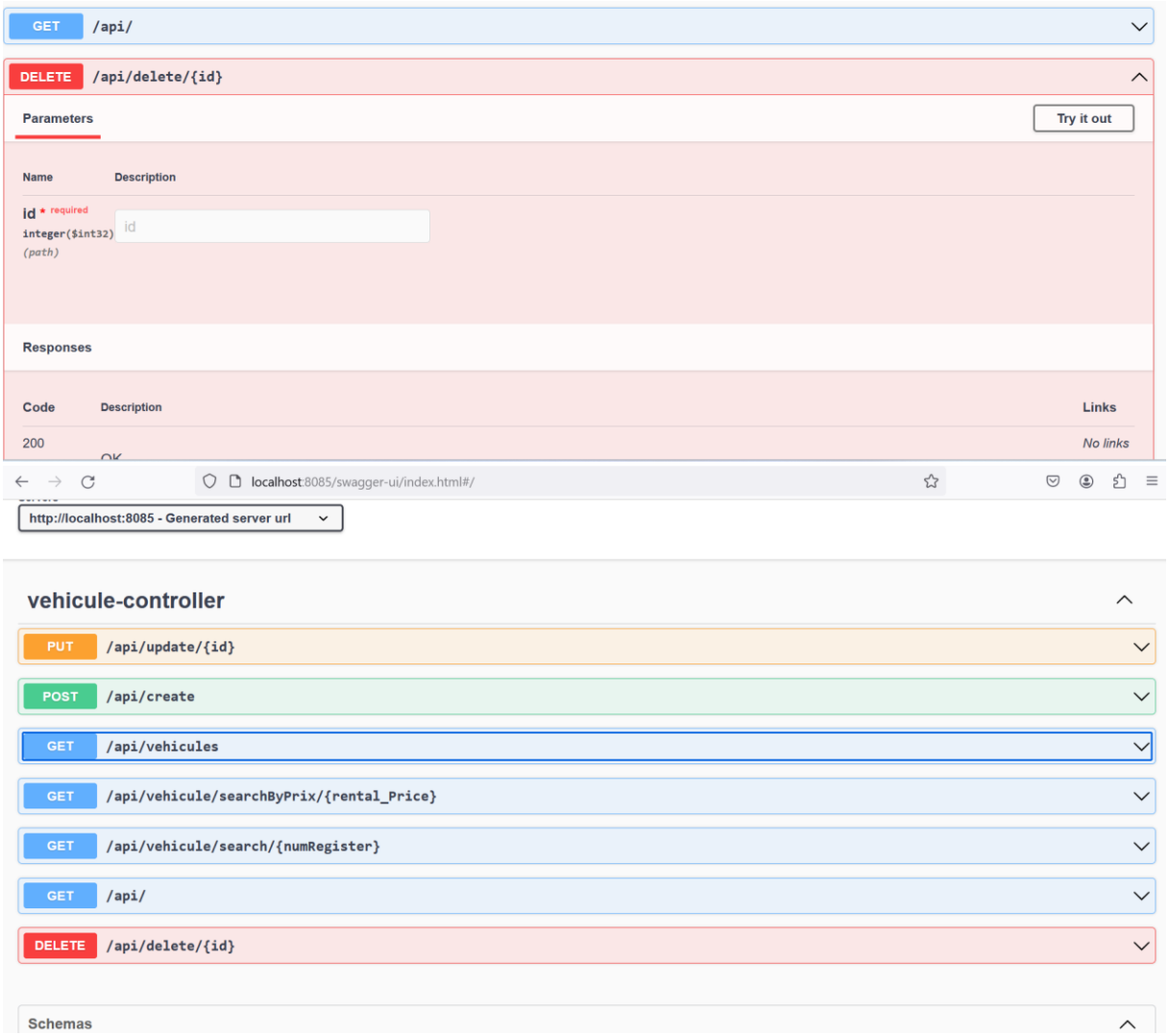
- **Politiques de rétention des données** : Définissez des politiques de rétention des données pour déterminer la durée pendant laquelle les données seront conservées dans l'application. Assurez-vous de supprimer les données obsolètes ou inutiles conformément à ces politiques pour réduire les risques de divulgation accidentelle ou malveillante.

- **Protection contre les violations de données** : Mettez en place des mesures de détection des violations de données pour surveiller les activités suspectes et les tentatives d'accès non autorisées. Prévoyez des procédures de réponse aux incidents pour réagir rapidement en cas de violation de données et pour limiter les dommages potentiels.

En intégrant ces considérations de conformité et de protection des données dans le développement et la gestion de l'application de location de véhicules, Propelize peut assurer la confidentialité, l'intégrité et la disponibilité des données de ses utilisateurs, tout en se conformant aux exigences réglementaires en matière de protection des données.

Présentation de l'application avec Swagger

Notre application est accessible sur le lien http://localhost:8085/swagger-ui/index.htm après le démarrage de l'image



```
  "make": "string",
  "model": "string",
  "year": "string",
  "rentalPrice": 0,
  "numRegister": "string"
}'
```


Request URL

```
http://localhost:8085/api/update/2
```

Server response

Code	Details
200	Response body


```
{
  "id": 2,
  "make": "string",
  "model": "string",
  "year": "string",
  "rentalPrice": 0,
  "numRegister": "string"
}
```


Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Tue,21 May 2024 22:59:10 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links

vehicule-controller

PUT /api/update/{id}

POST /api/create

Parameters | Cancel | Reset

No parameters

Request body required | application/json


```
{
  "make": "Toyota",
  "model": "Model120",
  "year": "2022",
  "rentalPrice": 25000000,
  "numRegister": "A213DSA"
}
```


Request URL

```
http://localhost:8085/api/vehicules
```

Server response

Code	Details
200	Response body


```
[
  {
    "id": 1,
    "make": "Toyota",
    "model": "Model120",
    "year": "2022",
    "rentalPrice": 25000000,
    "numRegister": "A213DSA"
  },
  {
    "id": 2,
    "make": "Toyota",
    "model": "Model120",
    "year": "2022",
    "rentalPrice": 25000000,
    "numRegister": "A213DSA"
  }
]
```


Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Tue,21 May 2024 22:45:33 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

GET /api/vehicule/searchByPrix/{rental_Price}

Parameters Cancel

Name	Description
rental_Price * required number($double) (path)	200000

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8085/api/vehicule/searchByPrix/200000' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8085/api/vehicule/searchByPrix/200000
```

200

Response body


```
[
  {
    "id": 1,
    "make": "Toyota",
    "model": "Model120",
    "year": "2022",
    "rentalPrice": 25000000,
    "numRegister": "A213DSA"
  },
  {
    "id": 2,
    "make": "string",
    "model": "string",
    "year": "string",
    "rentalPrice": 0,
    "numRegister": "string"
  }
]
```


Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Tue,21 May 2024 23:01:29 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	*No links*