# ZiEL Cars

### Documentation Technique de ZINEB&ELISA

Sommaire :

1. **Introduction**
2. **Architecture du Projet**
   * Vue d'ensemble
   * Composants Principaux
3. **Configuration et Déploiement**
   * Prérequis
   * Installation
   * Déploiement avec Docker
4. **Description des Composants**
   * Frontend
   * Backend
   * API
   * Mobile
5. **Modèles de Données**
   * Utilisateurs
   * Voitures
   * Réservations
   * Notifications
6. **Services Externes**
   * SMTP pour les emails
   * MongoDB pour la base de données
7. **API Endpoints**
8. **Sécurité**
   * JWT pour l'authentification
   * Sécurité des cookies
9. **Guide Utilisateur**
   * Utilisation du frontend
   * Utilisation du backend
10. **Dépannage**
11. **Conclusion**
12. **Annexes**

### 1. Introduction

ZIEL est une plateforme de location de voitures, orientée fournisseur, avec un backend pour gérer les flottes de voitures et les réservations, ainsi qu'un frontend et une application mobile pour la location de voitures. Conçu pour fonctionner avec plusieurs fournisseurs, ZiEL permet à chaque fournisseur de gérer sa flotte de voitures et ses réservations.

### 2. Architecture du Projet

#### Vue d'ensemble

L'architecture de ZiEL est basée sur une structure multi-modules, divisée en plusieurs composants principaux :

* **Frontend** : Interface utilisateur pour les clients.
* **Backend** : Interface de gestion pour les fournisseurs et les administrateurs.
* **API** : Fournit les endpoints nécessaires à la communication entre le frontend, le backend et la base de données.
* **Mobile** : Application mobile pour les clients.

#### Composants Principaux

* **API** : Gère les requêtes HTTP et communique avec la base de données MongoDB.
* **Frontend** : Développé avec React, il offre une interface utilisateur pour la recherche et la réservation de voitures.
* **Backend** : Développé avec React et Node.js, il permet aux fournisseurs de gérer leurs voitures et leurs réservations.
* **Mobile** : Développé avec React Native pour les utilisateurs mobiles.

### 3. Configuration et Déploiement

#### Prérequis

* Node.js
* MongoDB
* Docker
* npm ou yarn

#### Installation

Clonez le dépôt :

bash

Copier le code

git clone <repository\_url>

cd bookcars-main

Installez les dépendances pour chaque module (frontend, backend, api, mobile) :

bash

Copier le code

cd <module>

npm install

#### Déploiement avec Docker

Le projet utilise Docker Compose pour simplifier le déploiement. Voici le fichier docker-compose.yml :

yaml

Copier le code

version: "3.8"

services:

api:

build:

context: .

dockerfile: ./api/Dockerfile

env\_file: ./api/.env.docker

restart: always

ports:

- 4002:4002

depends\_on:

- mongo

volumes:

- cdn:/var/www/cdn/bookcars

mongo:

image: mongo:latest

command: mongod --quiet --logpath /dev/null

restart: always

environment:

MONGO\_INITDB\_ROOT\_USERNAME: admin

MONGO\_INITDB\_ROOT\_PASSWORD: PASSWORD

ports:

- 27017:27017

backend:

build:

context: .

dockerfile: ./backend/Dockerfile

depends\_on:

- api

ports:

- 3001:3001

frontend:

build:

context: .

dockerfile: ./frontend/Dockerfile

depends\_on:

- api

ports:

- 80:80

volumes:

- cdn:/var/www/cdn/bookcars

volumes:

cdn:

Déployez les services :

bash

Copier le code

docker-compose up -d

### 4. Description des Composants

#### Frontend

Le frontend est une application React qui permet aux utilisateurs de rechercher et de réserver des voitures. Il interagit avec l'API pour obtenir des informations sur les voitures et gérer les réservations.

**Principales technologies** :

* React
* Redux (pour la gestion de l'état)
* Axios (pour les appels API)
* React Router (pour la navigation)

#### Backend

Le backend est une application React et Node.js qui permet aux fournisseurs de gérer leurs voitures et leurs réservations. Il offre une interface utilisateur pour les tâches administratives.

**Principales technologies** :

* Node.js
* Express
* MongoDB

#### API

L'API, développée avec Node.js et Express, gère les requêtes HTTP, interagit avec la base de données MongoDB, et assure la logique métier.

**Principales technologies** :

* Node.js
* Express
* Mongoose (pour l'interaction avec MongoDB)

#### Mobile

L'application mobile, développée avec React Native, permet aux utilisateurs de rechercher et de réserver des voitures depuis leurs appareils mobiles.

**Principales technologies** :

* React Native
* Expo

### 5. Modèles de Données

#### Utilisateurs

json

Copier le code

{

"supplier": "ObjectId",

"fullName": "string",

"email": "string",

"phone": "string",

"password": "string",

"birthDate": "Date",

"verified": "boolean",

"verifiedAt": "Date",

"active": "boolean",

"language": "string",

"enableEmailNotifications": "boolean",

"avatar": "string",

"bio": "string",

"location": "string",

"type": "string",

"blacklisted": "boolean",

"payLater": "boolean",

"customerId": "string"

}

#### Voitures

json

Copier le code

{

"name": "string",

"supplier": "ObjectId",

"minimumAge": "number",

"locations": ["ObjectId"],

"price": "number",

"deposit": "number",

"available": "boolean",

"type": "string",

"gearbox": "string",

"aircon": "boolean",

"image": "string",

"seats": "number",

"doors": "number",

"fuelPolicy": "string",

"mileage": "number",

"cancellation": "number",

"amendments": "number",

"theftProtection": "number",

"collisionDamageWaiver": "number",

"fullInsurance": "number",

"additionalDriver": "number"

}

#### Réservations

json

Copier le code

{

"supplier": "ObjectId",

"car": "ObjectId",

"driver": "ObjectId",

"pickupLocation": "ObjectId",

"dropOffLocation": "ObjectId",

"from": "Date",

"to": "Date",

"status": "string",

"cancellation": "boolean",

"amendments": "boolean",

"theftProtection": "boolean",

"collisionDamageWaiver": "boolean",

"fullInsurance": "boolean",

"additionalDriver": "boolean",

"\_additionalDriver": "ObjectId",

"cancelRequest": "boolean",

"price": "number",

"sessionId": "string",

"paymentIntentId": "string",

"customerId": "string",

"expireAt": "Date"

}

#### Notifications

json

Copier le code

{

"user": "ObjectId",

"message": "string",

"booking": "ObjectId",

"isRead": "boolean"

}

### 6. Services Externes

#### SMTP pour les emails

L'application utilise un service SMTP pour envoyer des emails de confirmation et de notification. Les configurations SMTP sont définies dans le fichier .env.

#### MongoDB pour la base de données

MongoDB est utilisé pour stocker toutes les données de l'application, y compris les utilisateurs, les voitures, les réservations et les notifications.

### 7. API Endpoints

Quelques exemples d'endpoints :

* GET /api/cars : Récupérer la liste des voitures disponibles.
* POST /api/bookings : Créer une nouvelle réservation.
* GET /api/users/:id : Récupérer les informations d'un utilisateur.

### 8. Sécurité

#### JWT pour l'authentification

L'API utilise des tokens JWT pour authentifier les utilisateurs. Les tokens sont signés avec une clé secrète définie dans le fichier .env.

#### Sécurité des cookies

Les cookies d'authentification sont sécurisés, HTTPOnly, signés et ont une politique de même site stricte.

### 9. Guide Utilisateur

#### Utilisation du frontend

1. **Recherche de voitures** : Les utilisateurs peuvent rechercher des voitures en fonction de l'emplacement et des dates de prise en charge et de restitution.
2. **Réservation** : Une fois qu'une voiture est sélectionnée, l'utilisateur peut compléter le processus de réservation.

#### Utilisation du backend

1. **Gestion des voitures** : Les fournisseurs peuvent ajouter, modifier ou supprimer des voitures de leur flotte.
2. **Gestion des réservations** : Les fournisseurs peuvent consulter, approuver ou annuler des réservations.

### 10. Dépannage

* **Problèmes de connexion à la base de données** : Vérifiez que MongoDB est en cours d'exécution et que les configurations de connexion sont correctes.
* **Erreurs d'API** : Consultez les logs du serveur pour obtenir des informations détaillées sur les erreurs.

### 11. Conclusion

ZIEL offre une solution complète pour la gestion de la location de voitures, avec des interfaces utilisateur intuitives et une architecture solide. Cette documentation vise à fournir toutes les informations nécessaires pour installer, configurer et maintenir le système.

Some Images :

Une image contenant texte, véhicule, Véhicule terrestre, roue

Description générée automatiquementUne image contenant texte, véhicule, voiture, Véhicule terrestre

Description générée automatiquementUne image contenant texte, capture d’écran, Police, nombre

Description générée automatiquementUne image contenant texte, capture d’écran, Police, document

Description générée automatiquementUne image contenant texte, capture d’écran, Police

Description générée automatiquementUne image contenant texte, capture d’écran, Police, nombre

Description générée automatiquementUne image contenant texte, capture d’écran, Police

Description générée automatiquementUne image contenant texte, capture d’écran, Police, algèbre

Description générée automatiquement