DOSSIER DE PROJET

Vigneron Tristan

Développeur Web et Web mobile
Studi

Site Web Garage automobile V.Parrot

Remerciement :

En finalisant ce dossier projet, je tiens à exprimer ma profonde gratitude envers les individus et les groupes qui ont joué un rôle fondamental dans mon cheminement. Le soutien et l'encouragement que j'ai reçus de leur part ont été inestimables et ont largement contribué à mon développement personnel et professionnel.

Tout d'abord, je souhaite exprimer ma reconnaissance envers Studi, l'institution qui a rendu possible ma formation. Leur engagement envers l'éducation et l'apprentissage m'a ouvert les portes vers de nouvelles opportunités. Leur expertise et leur dévouement ont modelé mon expérience d'apprentissage de manière significative, et je suis profondément reconnaissant(e) pour leur soutien constant.

Un remerciement tout particulier est adressé à mes chers amis, des développeurs chevronnés. Leur passion pour le domaine et leur enthousiasme à partager leurs connaissances m'ont inspiré à explorer le monde de la programmation et du développement. Sans leur influence positive et leurs conseils précieux, je n'aurais jamais envisagé plonger dans ce domaine captivant. Leur amitié et leur mentorat ont été une source inestimable de motivation.

Enfin, je tiens à exprimer ma gratitude envers ma famille. Leur soutien inconditionnel et leur amour m'ont toujours procuré la confiance nécessaire pour poursuivre mes objectifs. Leur présence a allégé mes préoccupations financières et m'a permis de me concentrer pleinement sur mes études et mon projet. Leur compréhension et leurs encouragements ont constitué les piliers de mon parcours, et je leur en suis profondément reconnaissant(e).

Sommaire

Introduction.

1. Compétences du référentiel abordées dans le projet.

a. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité.

- Maquettage d'une application.
- Réalisation d'une interface web adaptable.
- Développement d'une interface dynamique.

Le site Web doit être fonctionnel à la fois sur un ordinateur ainsi que sur un téléphone ou une tablette. C'est la raison pour laquelle le site devait être conçu de manière responsive, assurant ainsi une expérience utilisateur cohérente sur les divers supports. Pour répondre à ce besoin, j'ai choisi d'utiliser React JS, un framework reconnu pour son approche de développement orientée composants, pour sa capacité à faciliter la création d'interfaces réactives et avec un écosystème robuste et une documentation exhaustive.

b. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité.

- Création d'une base de données.
- Développement des composants d'accès aux données.

La partie front-end fait appel à une API spécialement conçue pour extraire les données d'une base de données relationnelle que j'ai développée. Dans cette démarche, j'ai privilégié l'utilisation du langage PHP en raison de sa vaste et active communauté de développeurs. Cette communauté offre non seulement un soutien robuste, mais aussi un éventail de ressources en ligne qui ont grandement facilité le processus de développement. De plus, le choix de PHP s'est avéré judicieux en raison de son support naturel pour la gestion des bases de données, ce qui a contribué à garantir la performance et la fiabilité de l'ensemble du système.

2. Résumé du projet.

Dans le cadre de ma formation en tant que Développeur Flutter Gradué, j'ai entrepris la réalisation d'une Épreuve de Certification Finale (ECF). L'objectif central de cette épreuve était de traverser chaque étape clé de la conception d'une application web complète.

Mon projet visait à créer un site web destiné à un garage automobile, dans lequel les utilisateurs auraient la possibilité d'accéder à divers services tels que la réparation, le dépannage, ainsi qu'une vitrine présentant les voitures d'occasion disponibles au garage. Les fonctionnalités comprenaient également la capacité de contacter le garage et de fournir des avis.

En qualité de développeur unique de ce projet, j'ai pu renforcer mes connaissances préalables tout en en acquérant de nouvelles, grâce à la consultation de documentations et de vidéos concernant les différentes technologies pertinentes à mon projet. Cette approche en solitaire m'a offert l'opportunité d'une consolidation des acquis et d'une acquisition de compétences élargies.

3. Environnement Humain et Technique.

a. Environnement humain.

J'ai travaillé en solitaire sur ce projet. Malgré la charge de travail plus élevée et l'isolement dû au fait de travailler seul, l'autonomie et le contrôle total sur la direction et les décisions du projet, ainsi que la flexibilité au niveau des horaires de travail et des méthodes de gestion, m'ont permis de suivre ma propre vision et d'explorer des idées sans les contraintes habituelles d'une structure hiérarchique.

Je peux adapter mes priorités en fonction des besoins du projet et de mon rythme personnel, renforçant ainsi mon sentiment de responsabilité et d'engagement envers mon travail.

b. Environnement Technique.

Je travaille sur un ordinateur de bureau équipé de deux écrans, ce qui me permet d'accéder plus aisément à mon code tout en le comparant à la documentation, aux divers forums et aux vidéos lorsque je rencontre des problèmes.

Mon choix s'est porté sur Visual Studio Code en raison de son interface ergonomique et de sa vaste gamme d'extensions. Pour assurer le suivi de l'évolution de mon code et la possibilité de revenir en arrière en cas de besoin, j'utilise GitHub.

Conception du projet.

1. Définitions des objectifs et du concept.

a. Réflexion.

Au premier stade, je me suis réuni pour réfléchir aux objectifs de mon site web. Quel était le but ultime que je souhaitais atteindre ? Quel problème mon site devait-il résoudre, ou quel besoin devait-il satisfaire ? J'ai également exploré différentes idées de concepts pour le

design et la fonctionnalité du site, en prenant en compte les tendances actuelles et des attentes en tant qu'utilisateur.

b. Recherche et Analyse.

Avec une vision générale en tête, j'ai entrepris des recherches approfondies pour mieux comprendre le marché, la concurrence éventuelle et les préférences des utilisateurs. Cette étape m'a aidé à affiner mon concept initial, à identifier les lacunes potentielles et à découvrir des opportunités que je pourrais exploiter.

c. Cahier des charges.

Plutôt que de chercher à remplir le cahier des charges en une seule étape, j'ai adopté une approche progressive. J'ai commencé par rédiger les grandes lignes, en incluant les principales fonctionnalités, la structure du site et les objectifs de conception. À mesure que j'avançais dans le processus de développement, j'ai affiné et détaillé davantage chaque section du cahier des charges.

d. Conclusion.

En somme, bien que mon projet de site web ait débuté avec un cahier des charges vide, il est devenu le résultat d'un processus méticuleux et itératif. Chaque étape, de la définition des objectifs à l'élaboration du cahier des charges en passant par la recherche et les consultations, a contribué à transformer l'idée en une réalité bien conçue. Cette approche progressive m'a permis d'assurer que chaque détail a été pris en compte et que le site web serait prêt à offrir une expérience exceptionnelle à ses utilisateurs.

2. Spécification technique.

Voici les différentes technologies que j’ai utilisé pour ce projet :

a. Environnement de travaille :

- Visual Studio code : Un éditeur de code source léger et puissant qui offre de nombreuses fonctionnalités pour faciliter le développement, telles que la coloration syntaxique, l'autocomplétion et l'intégration de débogage.
- Git et Github : Git est un système de contrôle de version qui permet de suivre et de gérer les changements dans le code source. GitHub est une plateforme en ligne qui facilite l'hébergement et la collaboration autour des projets Git.
- Postman : Une application qui simplifie le test et le débogage des API en permettant aux développeurs de créer et d'envoyer des requêtes HTTP personnalisées.
- npm (Node Package Manager) : Un gestionnaire de paquets JavaScript qui permet d'installer, de gérer et de partager des packages et des dépendances pour les projets.
- Figma : Une plateforme de conception d'interfaces utilisateur basée sur le cloud, qui permet aux designers de créer, collaborer et partager des maquettes, des prototypes et des ressources de conception.

b. Partie front-end :

- React JS : Une bibliothèque JavaScript populaire pour la construction d'interfaces utilisateur interactives et dynamiques, en utilisant des composants réutilisables pour organiser le code de manière modulaire.
- React Router Dom : Une extension de React qui permet la gestion de la navigation entre différentes vues (ou pages) d'une application web à interface utilisateur unique.
- Axios : Une bibliothèque JavaScript utilisée pour effectuer des requêtes HTTP depuis le navigateur, facilitant ainsi la communication avec le back-end.
- Material-ui et Bootstrap : Des bibliothèques de composants et de styles préconçus pour faciliter la conception d'interfaces esthétiques et réactives.

c. Partie back end :

- Php : Un langage de script côté serveur largement utilisé pour développer la logique métier d'une application web, gérer les requêtes et les réponses du serveur.
- Composer : Un gestionnaire de dépendances pour PHP qui facilite l'installation et la gestion des bibliothèques et des packages nécessaires pour le développement.
- JWT-Firebase :Une bibliothèque qui facilite la gestion des JSON Web Tokens (JWT) en utilisant Firebase pour l'authentification et l'autorisation sécurisées.

d. Base de données :

- Mysql : Un système de gestion de base de données relationnelles populaire, utilisé pour stocker de manière structurée les données essentielles de l'application.

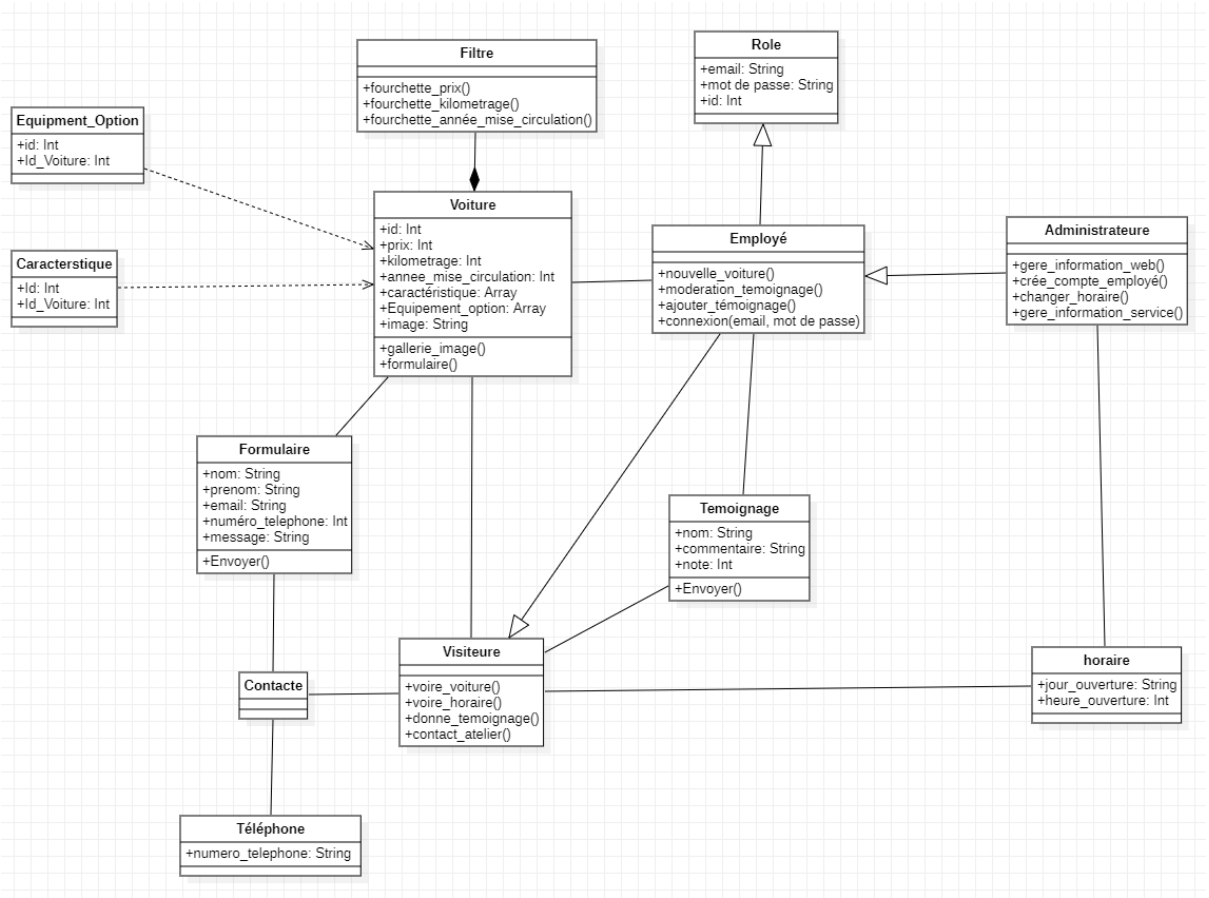
Réalisation du projet.

1. De la Conception à l'Implémentation: Préparation Avant le Code.

a. Élaboration du schéma de la base de données.

En utilisant les informations recueillies lors de mes recherches préliminaires, j'ai commencé par élaborer un schéma de ma base de données. Ce schéma représente les données que j'ai collectées, leurs caractéristiques respectives, leurs fonctions et les relations qui existent entre elles.

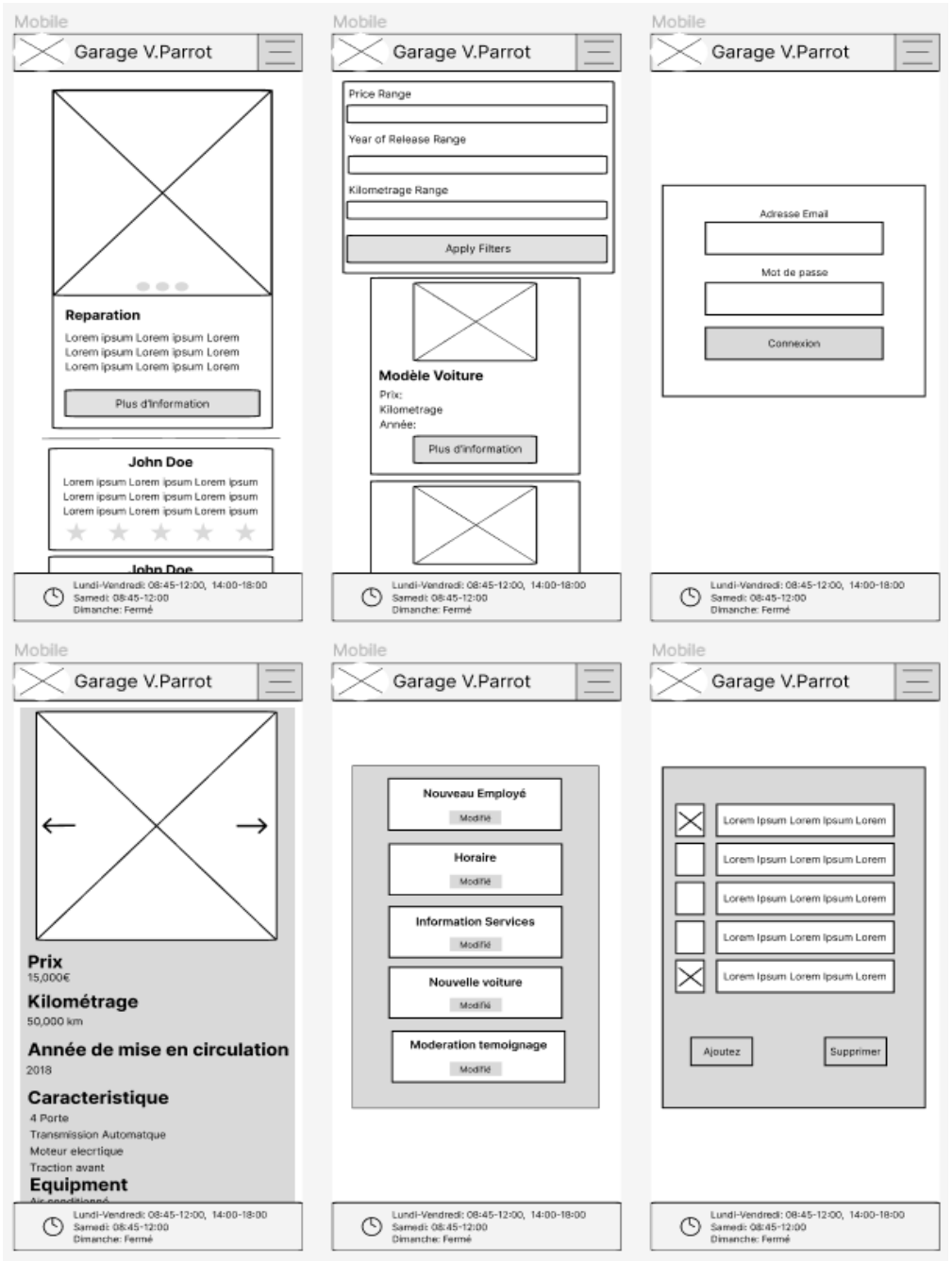
Diagramme de classe de la base de donné

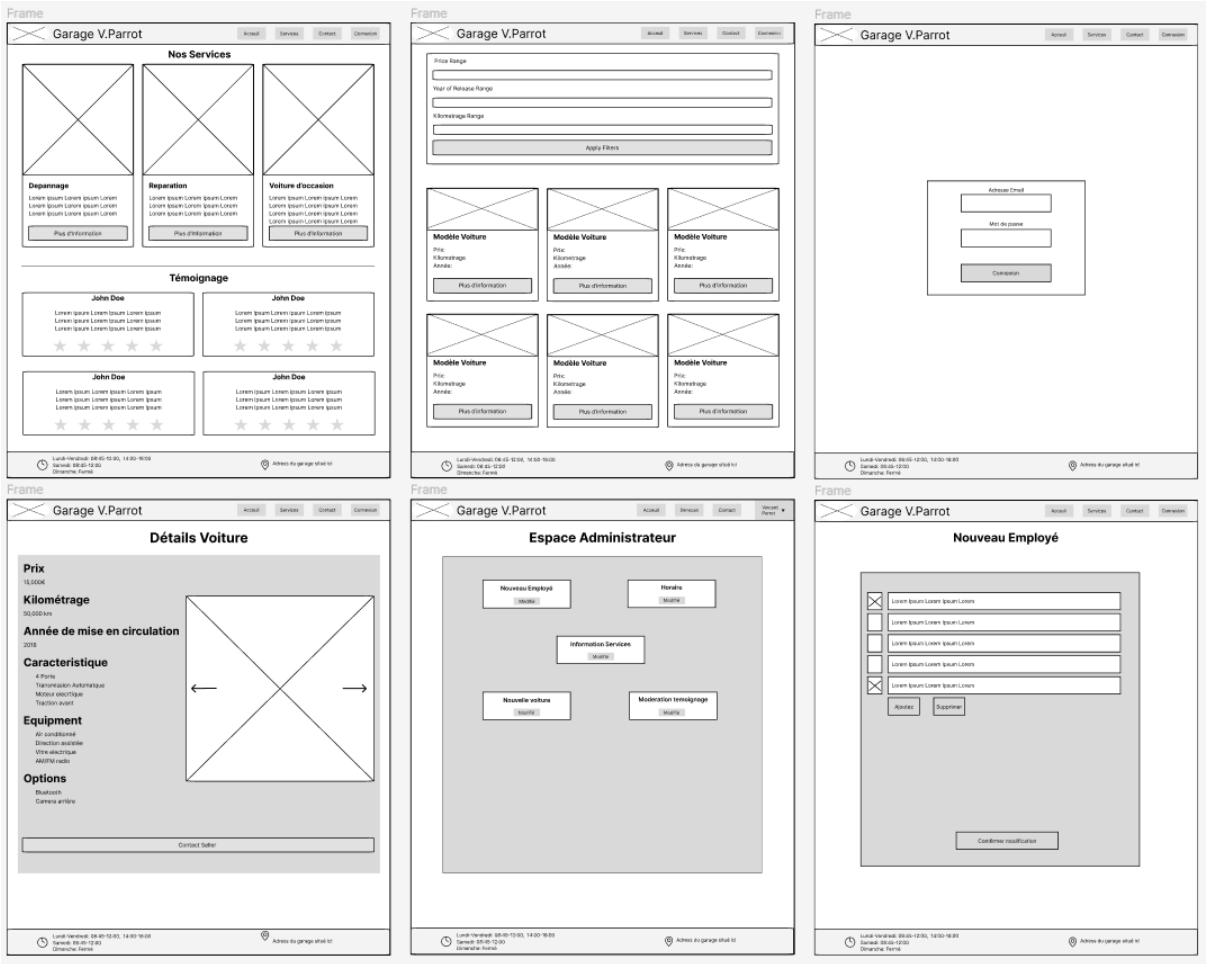


b. Conception du design visuel et de la charte graphique.

Ensuite avant de coder j’ai décidé de faire le Désigne du site à l'aide de Figma ce qui m'a permis de me décider aussi sur ma charte graphique. j’ai commencé par faire le wireframe de la version mobile puis de la version desktop

Wireframe mobile et Desktop





2. L'environnement de développement : Installation des bibliothèques et outils nécessaires.

Ma première étape a été l'installation des différentes bibliothèques principales que j'étais sûre d'utiliser tout au long de la réalisation du projet.

a. Installation de React :

Pour construire l'interface utilisateur réactive de mon application, j'ai choisi d'utiliser React, une bibliothèque JavaScript populaire. Pour l'installer, j'ai utilisé Node Package Manager (npm) qui est livré avec Node.js, notre gestionnaire de paquets.

```
npm install react
```

b. Installation d'Axios :

Afin de gérer facilement les requêtes HTTP vers un serveur, j'ai opté pour Axios, une bibliothèque qui simplifie la communication entre le front-end et le back-end.

```
npm install axios
```

c. Intégration Bootstrap et Intégration de Material-UI :

Pour assurer une expérience utilisateur cohérente sur une variété de périphériques, j'ai intégré le framework CSS réactif Bootstrap. Les grilles flexibles de Bootstrap m'ont permis

de créer facilement une mise en page adaptée aux différentes tailles d'écran. J'ai installé Bootstrap via npm.

```
npm i bootstrap@5.3.1
```

Pour ajouter des composants d'interface utilisateur élégants et personnalisables, j'ai choisi Material-UI. Son installation s'est faite de la même manière via npm.

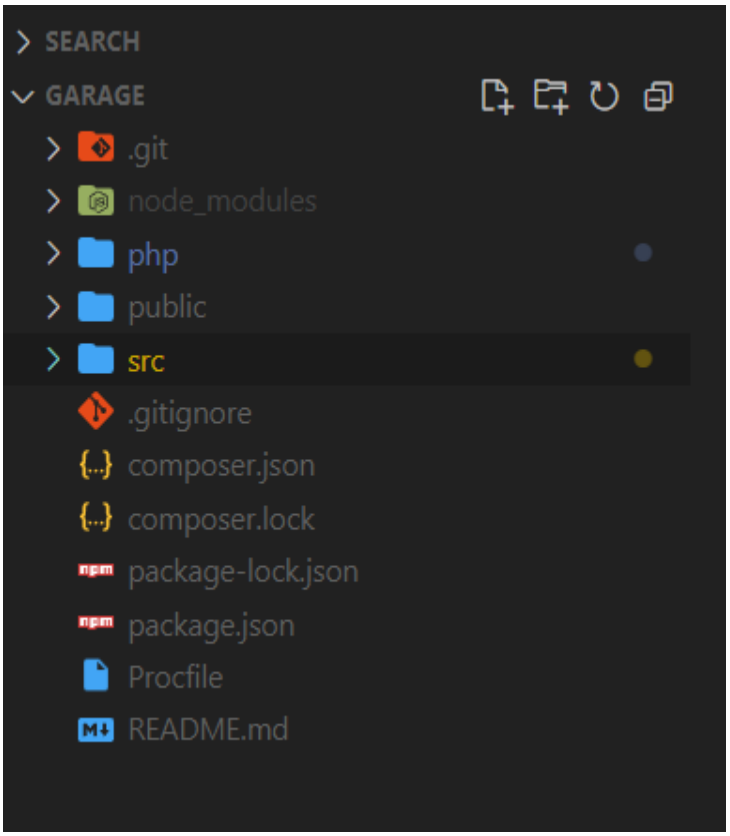
```
npm install @mui/material @emotion/react @emotion/styled
```

d. Mise en place de React Router DOM :

Pour gérer la navigation entre les différentes vues de mon application, j'ai intégré React Router DOM. Son installation a également été réalisée avec npm.

```
npm install react-router-dom
```

e. Arborescence du dossier



Arborescence des dossiers

Une fois les différentes bibliothèques installées j'ai créé un dossier php ou j'allais écrire tout le code de la partie back end.

Cette disposition me permet de séparer clairement tout le front dans le dossier src et tout le back end dans le dossier php

Le back-END

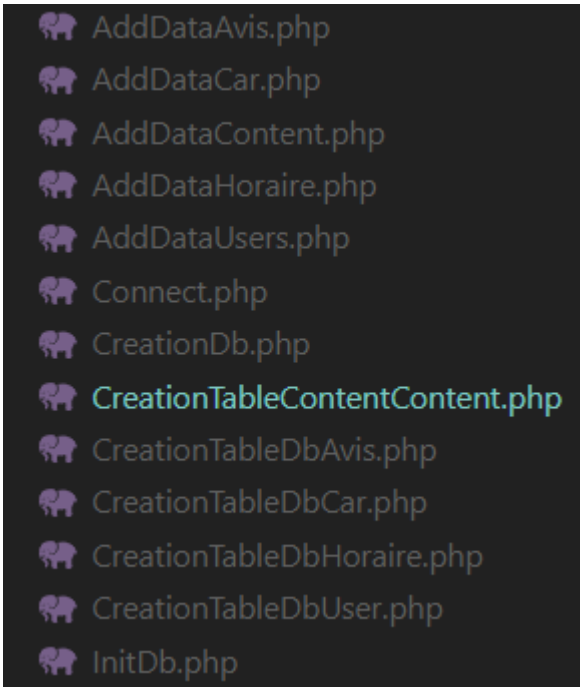
1. Mise en place de la base de donnée

a. Introduction

Dans le cadre de ce projet, l'utilisation d'outils graphiques tels que PhpMyAdmin pour la création de la base de données a été exclue. En conséquence, j'ai mis en place la base de données en développant des classes en PHP
Je me suis concentré sur la mise en place de la logique serveur, qui est le cœur de la dynamique de l'application. C'est ce qui permet aux utilisateurs de s'interagir avec la base de données, d'effectuer des requêtes et d'obtenir des réponses.

b. Utilisation des classes php

Classe base de donnée php



J'ai conçu différentes classes qui correspondent aux besoins spécifiques du site web.

Ces classes encapsulant les fonctionnalités liées à la base de données et fournissent des méthodes pour effectuer des opérations telles que l'insertion, la mise à jour et la récupération de données
.

c. Gestion des erreures

Dans chacune de ces méthodes, j'ai mis en place des blocs "try-catch" pour gérer les erreurs potentielles. En cas d'échec lors de l'exécution d'une opération de base de données, les erreurs sont capturées par le bloc "catch" est affichées pour faciliter le débogage.

d. Utilisation de Pdo

Pour interagir avec la base de données, j'ai choisi d'utiliser l'extension PDO (PHP Data Objects). PDO offre une sécurité accrue en permettant l'utilisation de requêtes préparées, ce qui aide à prévenir les attaques par injection SQL. De plus, PDO améliore les performances grâce à sa gestion efficace des connexions à la base de données.

```php
public function dbConnection()
    {
        $this->conn = null;
        try {
```

```
            $this->conn = new
PDO("mysql:host=".$this->serverName.";",$this->userName,
$this->password);
            $this->conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $e) {
            echo "Connection Failed: " . $e->getMessage();
        }
        return $this->conn;
    }
```

J'ai utilisé MySQL pour stocker toutes les données essentielles de l'application, notamment les informations des utilisateurs, les détails des voitures et les avis des clients. J'ai créé plusieurs tables relationnelles, en m'assurant que chaque table était optimisée pour des requêtes rapides et efficaces.

e. Requête

Grâce à l'utilisation des requêtes PDO, j'ai la possibilité d'employer la syntaxe MySQL pour la création de mes tables. facilitant ainsi l'établissement de liens entre ces tables à l'aide de clés étrangères.

```
$cvSql = "CREATE TABLE IF NOT EXISTS CVVOITURE (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    voiture_id INT NOT NULL,
    caracteristique_id INT NOT NULL,
    FOREIGN KEY (voiture_id) REFERENCES VOITURES(id),
    FOREIGN KEY (caracteristique_id) REFERENCES CARACTERISTIQUE(id)
    )";
```

J'utilise différentes contraintes (constraint) pour garantir l'intégrité des données et maintenir la cohérence de la base de données.

Je vérifie que la table n'est pas déjà existante avec "`NOT EXISTS`" cela permet d'éviter les erreurs dues à la duplication de tables..
J'utilisé "`NOT NULL`" pour indiquer que les colonnes "`voiture_id`" et "`caracteristique_id`" ne peuvent pas contenir de valeurs NULL doit avoir des valeurs valides pour ces colonnes. Cela signifie que chaque ligne dans la table "`CVVOITURE`" doit avoir des valeurs valides pour ces colonnes.

La clause "`PRIMARY KEY`" de mon "`id`" est utilisée pour définir la clé primaire de la table. Ce qui signifie qu'elle doit contenir des valeurs uniques pour chaque ligne et qu'elle est "`NOT NULL`".

J'ai pu facilement obtenir les caractéristiques d'une voiture particulière en utilisant des requêtes SQL qui rejoignent les tables "`CVVOITURE`", "`VOITURES`" et "`CARACTERISTIQUE`"

grâce aux clés étrangères “FOREIGN KEY” pour qu'elles fassent référence “REFERENCES” aux colonnes "id" des tables “VOITURES” et “CARACTERISTIQUE”.

f. Création de la base de donné

Une fois tous les fichiers nécessaires à la base de données créés j’ai créé un script php pour configurer et initialiser la base de données ainsi que ses tables, puis peupler ces tables avec des données.

```php
include_once 'Connect.php';
include_once 'CreationDb.php';
include_once 'CreationTableDbUser.php';
include_once 'CreationTableDbCar.php';
include_once 'CreationTableDbAvis.php';
include_once 'CreationTableDbHoraire.php';
include_once 'CreationTableDbContent.php';
include_once 'AddDataCar.php';
include_once 'AddDataContent.php';
include_once 'AddDataHoraire.php';
include_once 'AddDataAvis.php';
include_once 'AddDataUsers.php';
```

←Inclusion des tables

Le code débute par une série d'instructions d'inclusion qui importent les différents fichiers PHP contenant les définitions de classes requises.

```php
$db_create = new DatabaseCreate();
$db_table_user = new DatabaseTableCreateUser();
$db_table_car = new DatabaseTableCreateCar();
$db_table_avis = new DatabaseTableCreateAvis();
$db_table_horaire = new DatabaseTableCreateHoraire();
$db_table_content = new DatabaseContent();
$add_data_car = new AddDataCar();
$add_data_content = new AddDataContent();
$add_data_horaire = new AddDataHoraire();
$add_data_avis = new AddDataAvis();
$add_data_users = new AddDataUsers();
```

←Initialisation des instances

Ensuite, les instances des différentes classes sont instanciées et initialisées.

```php
try {
    $db_create->creationDb();
    $db_table_user->creationTableUser();
    $db_table_car->creationTableCar();
    $db_table_avis->creationTableAvis();
    $db_table_horaire->creationTableHoraire();
    $db_table_content->creationContent();
    $add_data_content->dataContent();
    $add_data_horaire->dataHoraire();
    $add_data_avis->dataAvis();
    $add_data_users->dataUser();
    $add_data_car->dataCar();
} catch (PDOException $e) {
    echo "Connection Raté : <br>" . $e->getMessage();
    exit;
}
```

←Bloc try-catch

Pour finir on crée la base de données, les différente tables qu’on peuple ensuite de donnée le tout dans un bloc “try-catch” pour pouvoir récupérer les erreurs

2. Développement de l’api CRUD

a. Introduction

Au cœur de mon projet réside le développement d'une API CRUD (Create, Read, Update, Delete). Cette API offre la possibilité d'effectuer les opérations essentielles sur une base de données : création, lecture, mise à jour et suppression de données. Dans le cadre du modèle MVC (Modèle-Vue-Contrôleur), le Modèle et le Contrôleur jouent un rôle central dans la logique back-end développé en Php, tandis que la Vue est représentée par la partie front-end développée en React.

b. Modèle (MVC)

La mise en place de l'API commence par la définition précise de la structure des données et de la logique applicative. Chaque table de la base de données est associée à un script contenant les fonctions nécessaires pour mettre en œuvre les opérations CRUD. Ces fonctions garantissent une manipulation sécurisée des données, en respectant les quatre opérations fondamentales : Create (créer), Read (lire), Update (mettre à jour) et Delete (supprimer).

Chaque étape du processus est soigneusement orchestrée. La connexion à la base de données est initialement établie dans une propriété privée.

```
private $conn;
```

Suivie par la définition des propriétés publiques liées à la table concernée comme par exemple l’id d’une voiture dans la base de données.

```
public $id;
```

Le constructeur assure ensuite la liaison avec la base de données.

```
  public function __construct($db)
    {
        $this->conn = $db;
    }
```

Ensuite j’ai créé les fonctions nécessaire pour l’API par exemple pour ma table `VOITURE` j'ai besoin d’une fonction ajouter(Create).

```
public function createVoiture(){<--->}
```

Je crée la Requête Mysql et je la prépare.

```
$sql = "INSERT INTO VOITURES
                SET
                    prix = :prix,
                    kilometrage = :kilometrage,
                    annee_circulation = :annee_circulation,
                    modele = :modele,
                    nom = :nom,
                    prenom = :prenom,
                    numero = :numero";
        $stmt = $this->conn->prepare($sql);
```

Lors de la création de fonctions CRUD, une attention particulière est accordée à la sécurisation des données. En appliquant des méthodes telles que `htmlspecialchars` et `strip_tags`, prémunissant contre les vulnérabilités d'injection SQL. Ces mesures permettent de préparer les données pour une utilisation sûre dans les requêtes SQL.

```
$this->prix = htmlspecialchars(strip_tags($this->prix));
```

- `strip_tags` : supprime toutes les balises HTML et PHP .
- `htmlspecialchars` : convertit les caractères spéciaux en entités HTML.

En combinant ces deux fonctions, la ligne de code assure que la valeur est prête à être utilisée en toute sécurité dans une requête SQL.
Ensuite je lie les paramètre nommé à une variable dans une requête préparée

```
$stmt->bindParam(":prix", $this->prix);
```

Enfin je vérifie si la requête a bien réussie

```
        if ($stmt->execute()) {
            return true;
        }
        return false;
```

c. Contrôleure (MVC)

La partie Contrôleur joue un rôle essentiel dans la communication entre le back-end (PHP) et le front-end (React). Les fichiers nécessaires pour la connexion à la base de données et les classes contenant les méthodes CRUD sont inclus pour garantir un flux de données fluide et sécurisé entre les deux parties. Ces scripts agissent comme des portes d'entrée(endpoint) dédiées pour l'API depuis l'application React.

Pour autoriser les requêtes cross-origin, des en-têtes CORS sont mis en place avec précision. Ces en-têtes spécifient les autorisations de méthode, d'origine et d'autres paramètres cruciaux pour une interaction sécurisée entre les parties.

```
header("Access-Control-Allow-Origin: http://localhost:3000");
header("Access-Control-Allow-Methods: GET,");
header('Access-Control-Allow-Credentials: true');
header('Content-Type: application/json');
header("Access-Control-Allow-Headers: Content-Type,
Access-Control-Allow-Methods,Access-Control-Allow-Origin,
Access-Control-Allow-Credentials, Authorization, X-Requested-With");
```

En vue de la séparation entre le back-end et le front-end, les fichiers de connexion à la base de données et les classes pertinentes sont inclus. La création d'une instance de connexion à la base de données permet d'établir un lien essentiel entre les données stockées et les requêtes formulées par l'API.

```
include_once '../../Database/Connect.php';
include_once '../../Class/Voiture.php';
```

Je crée un objet et utilise sa méthode pour obtenir une instance de la connexion à la base de données.

```
$database = new DatabaseConnect();
$db = $database->dbConnectionNamed();
```

Je crée ensuite un objet ou plusieure objets en fonction des besoins auquelle je passe la connexion en paramètre

```
$items = new Voiture($db);
```

Les requêtes HTTP, émises par le front-end en React, sont acheminées vers les fonctions CRUD appropriées du contrôleur. En examinant la méthode de la requête, nous identifions le type d'action à exécuter. Par exemple, si la méthode est "GET", nous appelons la fonction appropriée pour récupérer les données.

```
if ($_SERVER['REQUEST_METHOD'] === 'GET') {
    $stmt = $items->getVoiture();
      <--->
}
```

Dans le cas où la méthode n'est pas prise en charge, nous renvoyons une réponse d'erreur (code 405) indiquant que la méthode est non autorisée.

```
else {
    http_response_code(405);
    echo json_encode(array("message" => "Méthode non autorisée"));
}
```

Les données récupérées depuis la base de données sont formatées en JSON, puis renvoyées au front-end. Si des données sont présentes, elles sont incluses dans la réponse. Dans le cas contraire, une réponse d'erreur (code 404) indiquant l'absence de données est envoyée.

```
    $row = $stmt->fetchAll(PDO::FETCH_ASSOC);
    if ($row) {
        echo json_encode($row);
    } else {
        http_response_code(404);
        echo json_encode(array("message" => "Aucune voiture trouvée"));
    }
```

Bien entendu en fonction des données certaines validation des données sont requises. Exemple si je veux rajouter un employé dans la base de donnée une des vérification possible serait de regarder si l'adresse mail est déjà utilisée et si c'est le cas on renvoie une erreur.

```
            if ($check_email_stmt->rowCount()) :
                $returnData = msg(0, 422, 'Cette adresse Mail est déja
utilisé');
                http_response_code(422);
```

Ceci est un exemple mais la validation ne se limite pas à ça et dépend bien entendu de plein de paramètres.

Pour tester mon API et m'assurer que chaque endpoint renvoyait les bonnes données, j'ai utilisé Postman. Cet outil m'a permis de simuler des requêtes et d'examiner les réponses du serveur, m'aidant ainsi à repérer rapidement tout problème ou incohérence.

Le Front-END

1. Création des composants

Le développement du front-end a été organisé autour de la création de divers composants réutilisables pour chaque élément de l'interface utilisateur. Ces composants comprennent un en-tête, un pied de page, une section pour les voitures d'occasion, un formulaire de contact etc . . . Chaque composant est autonome en termes de logique et de style.

Par exemple, mon composant Header est responsable de la création de la barre de navigation en haut du site web. Voici une explication de son fonctionnement :

a. Import des composant et/ou assets

Des modules essentiels et des actifs ont été importés. Par exemple, le composant `React` gère la création et la gestion des composants de l'interface utilisateur, tandis que `Link` permet la navigation sans rechargement de la page. De plus, un hook personnalisé, `useAuth`, est utilisé pour gérer l'authentification.

```
import React from 'react';
import { Link } from 'react-router-dom';
import Logo from '../assests/Image/Logo.png';
import useAuth from '../hooks/useAuth';
```

Ces lignes d'importation sont essentielles pour utiliser les fonctionnalités de React, React Router DOM et les fonctionnalités personnalisées mises en place dans l'application.

b. Intégration du Hook useAuth

J'appelle le hook personnalisé `useAuth()` importé précédemment.

```
    const { auth, logout } = useAuth();
```

J'utilise la déstructuration pour extraire deux éléments de l'objet renvoyé par le hook

- `auth` : est un objet qui contient des informations sur l'authentification de l'utilisateur, telles que le jeton d'accès et le rôle.
- `logout` : est une fonction qui gère le processus de déconnexion de l'utilisateur.

Je défini une fonction `handleLogout` qui sera appelée lorsque l'utilisateur cliquera sur le bouton de déconnexion dans la barre de navigation.

```
  const handleLogout = () => {
        logout();
    };
```

À l'intérieur de cette fonction, on appelle simplement la fonction `logout()` Cela déclenche le processus de déconnexion de l'utilisateur.

Aussi j'utilise des expressions conditionnelles pour afficher certains liens de navigation en fonction de l'état d'authentification et du rôle de l'utilisateur.
Exemple si on a le rôle Admin, Employer ou que l'on est pas connecté

```
{auth.accessToken && auth.role === 'Admin' && (
    <li className="nav-item">
        <Link to="/adminSpace" className="bouton nav-link">
            Espace Admin
        </Link>
    </li>
)}

{auth.accessToken && auth.role === 'Employe' && (
   <li className="nav-item">
       <Link to="/employeSpace" className="bouton nav-link">
           Espace Employe
       </Link>
 </li>
)}

{!auth.accessToken && (
    <li className="nav-item">
        <Link to="/login" className="bouton nav-link">
            Connexion
        </Link>
    </li>
)}
```

c. Liens de navigation

La structure de la barre de navigation utilise les éléments JSX de React. Les éléments `Link` de React Router DOM assurent la navigation entre les vues de l'application. L'image du logo du site est également intégrée ici.

```
export default Header;
<nav className="border-bottom navbar navbar-expand-lg">
<div className="container-fluid">
  <div className="navbar-brand">
      <h1 className="align-items-center text-decoration-none lien">
         {/* Utilisation d'un lien de navigation vers la page
d'accueil* /}
```

```
          <Link to="/" className="lien">
              <img src={Logo} alt="Logo" className="Logo" />
          </Link>
      </h1>
  </div>
  {/* ... (autres parties de la barre de navigation) */}
</div>
</nav>
```

- `<Link to="/">` : Le composant `<Link>` de React Router DOM créer un lien vers la racine de l'application. Le prop `to` spécifie l'URL vers laquelle le lien doit pointer.
- `className="lien"` : Ajoutez une classe CSS "lien" à au lien de navigation pour appliquer des styles spécifiques aux liens de la barre de navigation.
- `<img src={Logo} alt="Logo" className="Logo" />` : J'insère une image de logo à l'intérieur du lien de navigation. La propriété `src` prend la valeur de l'image importée depuis le chemin `Logo` défini précédemment. L'attribut `alt` fournit une description alternative pour l'image.

En somme, l'adoption d'une approche modulaire avec la création de composants réutilisables pour chaque élément de l'interface utilisateur se révèle être un choix judicieux. Cette méthode favorise la création d'éléments efficaces, personnalisés et cohérents, tout en simplifiant la maintenance et en accélérant le développement.

2. Navigation avec react-router-dom

a. Introduction

Lors du développement d'un site web en utilisant React pour le front-end, il est essentiel de mettre en place un système de navigation efficace pour permettre aux utilisateurs de se déplacer entre différentes pages ou vues de l'application sans avoir à recharger complètement la page. La bibliothèque `react-router-dom` est essentielle pour la navigation. Elle permet de définir des routes pour l'application et d'afficher différents composants en fonction de l'URL.

b. Mise en place et définition des routes

Pour utiliser `react-router-dom`, j'ai commencez par installer la bibliothèque en utilisant npm

```
npm install react-router-dom
```

Une fois installée et importée, j'ai défini les routes de mon site web en utilisant les composants fournis par `react-router-dom`.

```
import { Routes, Route, } from 'react-router-dom';
```

Le composant `Routes` est le conteneur dans lequel je vais définir toutes mes routes.

```
function App() {
  return (
    <div className="App">
      <Routes>
        {/* Définition de mes routes ici */}
      </Routes>
    </div>
  );
}
```

Le composant `Route` est utilisé à l'intérieur du composant `Routes` pour définir chaque route individuellement. Chaque route est associée à un chemin d'URL spécifique et à un composant React que vous souhaitez afficher lorsque l'URL correspond.

```
    <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        {/*Le rest de mes routes*/}
    </Routes>
```

- `path`: Le chemin d'URL associé à la route. Lorsque l'URL du navigateur correspond à ce chemin, le composant `element` de cette route sera rendu
- `element`:: le composant React à afficher lorsque l'URL correspond au chemin de la route.

En utilisant le composant `Route` à l'intérieur d'un autre composant `Route`. Cela permet de créer des chemins d'URL plus complexes et de gérer des logiques de navigation conditionnelle.

```
<Routes>
    <Route path="/" element={<Layout />}>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        {/*Le rest de mes routes*/}
    </Route>
</Routes>
```

3. La vue (MVC)

a. Introduction

Dans l'architecture MVC, la Vue est chargée de gérer l'interface utilisateur. Elle affiche les données et agit comme intermédiaire entre les données et l'utilisateur.

b. Les requêtes http avec axios

L'interaction avec le back-end est cruciale pour obtenir et envoyer des données depuis et vers l'application front-end. C'est ici qu'intervient Axios, une bibliothèque JavaScript populaire pour effectuer des requêtes HTTP.

Pour la communication avec le back-end, la bibliothèque Axios a été choisie. Après son installation, des instances Axios ont été créées avec des configurations personnalisées pour gérer différentes URL et en-têtes.

```
npm install axios
import axios from "axios";
```

Chaque instance a une base URL, des en-têtes de contenu, et peut être configurée pour inclure des informations de cookies et de sessions. Prenons l'exemple de l'instance "`localhost` "

```
const localhost = axios.create({
    baseURL:"http://localhost",
    headers : {
        'Content-Type': 'application/json'
    },
    withCredentials: true
});
```

- Je configure `localhost` pour qu'il utilise l'URL "http://localhost" comme base URL.
- Défini l'en-tête `Content-Type` comme "application/json" pour les requêtes.
- Activé l'option `withCredentials` pour inclure les informations de cookies et de sessions.

Je regroupe ensuite mes instances Axios dans un objet de configuration, ce qui permet de réutiliser ces configurations dans d'autres parties de mon application. J'exporte cet objet pour pouvoir l'importer ailleurs.

```
const config = {
    localTestingUrl: localhost,
    herokuTesting: herokuUrl
}
export default config
```

Une fois ces configurations définies je les importé dans d'autres scripts.

```
import config from '../../api/axios';
```

Je définis les endpoints (URLs) pour vos requêtes.

```
const register_url = "/Api/Car/CarRead.php";
```

Exemple dans un bloc try-catch je peux utiliser les propriétés de l'objet `config` pour accéder aux instances Axios préconfigurées et effectuer des requêtes.

```
const fetchVoiture = async () => {
    try {
      const res = await config.herokuTesting.get(register_url);
```

```
      setVoiture(res.data);
    }
```

Si il y a une erreure je la récupère l'erreure et l'affiche

```
catch (err) {
     console.log(err)
    }
```

c. la Gestion des états (State)

L'affichage des données provenant des requêtes HTTP dans la Vue est une partie essentielle de la présentation des informations au sein de l'interface utilisateur. Les données récupérées du back-end doivent être formatées et affichées de manière claire et conviviale.

Pour une interface utilisateur agréable et réactive, Bootstrap et Material-UI ont été adoptés. Ils offrent des composants préfabriqués qui ont été utilisés pour, par exemple, afficher une liste de voitures et permettre aux utilisateurs de filtrer les résultats selon leurs préférences.

Je récupère les données au chargement du composant.

```
  useEffect(() => {
    fetchVoiture();
  }, []);
```

Je définis des fonctions pour gérer les changements dans les filtres de prix, d'année et de kilométrage et une pour reset ces mêmes valeurs.
Exemple pour le changement de Prix

```
  const priceChange = (event, newValue) => {
    setPriceRangeValue(newValue);
  };
```

Exemple pour le reset d'année

```
  const resetYearRange = () => {
    setYearRangeValue([1960, maxYear]);
  };
```

Je filtre les voitures en fonction des valeurs des filtres de prix, d'année et de kilométrage, en utilisant la méthode `.filter()`

```
  const carsInRange = voiture.filter((car) => {
    const carPrice = parseFloat(car.prix);
    const carYear = parseInt(car.annee_circulation);
    const carMileage = parseInt(car.kilometrage);
    return (
      carPrice >= priceRangeValue[0] &&
      carPrice <= priceRangeValue[1] &&
      carYear >= yearRangeValue[0] &&
      carYear <= yearRangeValue[1] &&
```

```
      carMileage >= kiloRangeValue[0] &&
      carMileage <= kiloRangeValue[1]
    );
  });
```

J'utilise le state pour mettre dynamiquement à jour les composants. Par exemple, dans mon composant de voiture j'utilise le state pour suivre les valeurs des différents filtres.

```
const [priceRangeValue, setPriceRangeValue] = useState([0, 25000]);
const [yearRangeValue, setYearRangeValue] = useState([1960, maxYear]);
const [kiloRangeValue, setKiloRangeValue] = useState([0, 300000]);
```

- `priceRangeValue` stocke la plage de prix sélectionnée par l'utilisateur.
- `yearRangeValue` stocke la plage d'années de fabrication des voitures.
- `kiloRangeValue` stocke la plage de kilométrage des voitures.

Ces états sont associés aux composants de Slider de Material-UI que nous utilisons pour permettre à l'utilisateur de choisir ses critères de filtrage.

```
import { Slider } from "@mui/material";
```

Lorsque l'utilisateur déplace les curseurs des Sliders, les valeurs de ces états sont mises à jour grâce aux fonctions `setPriceRangeValue`, `setYearRangeValue`, et `setKiloRangeValue`. Cela déclenche ensuite une réévaluation du filtrage des voitures, et seules les voitures correspondant aux nouvelles valeurs de filtre sont affichées

Cela garantit une expérience utilisateur interactive, où les résultats sont immédiatement ajustés en fonction des choix de l'utilisateur, rendant la recherche de voitures plus intuitive et efficace.

d. Style et désigne

Ensuite dans la partie JSX j'ai utilisé le composant slider pour permettre à l'utilisateur de choisir ces critères . Ce slider prend différent paramètre comme par exemple la fonction qui s'occupe de la plage de prix

```
            <Slider
              getAriaLabel={() => "Price Range"}
              value={priceRangeValue}
              onChange={priceChange}
              valueLabelDisplay="auto"
              min={minmum}
              max={maximum}
              sx={{ width: "300px", mx: "10px" }}
            />
```

Pour l’affichage des voitures j’ai créé un tableau d’objet unique basé sur l’ID pour supprimer les doublons de voiture.

```
const cars = uniqueCars.map((car) => {
    let carImages = [];
    if (car.voiture_images) {
      carImages = car.voiture_images.split(",");
    }
```

Pour chaque voiture, j'affiche les informations dans un conteneur `div` en utilisant des classes CSS et bootstrap pour le style, le responsive et l'agencement.

```
<div
        className="voit d-flex flex-column align-items-start m-3 px-1
flex-grow-0"
        key={car.id}
> . . . </div>
```

Pour chaque voiture on vérifie s' il y a des images. Si il y des image on affiche la première dans un “image-container " de bootstrap, sinon on affiche un message indiquant qu’il n’y a pas d’image.

```
<div className="image-container align-self-center p-1">
          {carImages.length > 0 ? (
            <img
              src={require(`../../assests/Image/${carImages[0]}`)}
              alt="cars"
              className="align-self-center py-3 img-fluid"
              style={{ width: "300px", height: "200px" }}
            />
          ) : (
            <div>No Image</div>
          )}
```

Le reste des informations est aussi affiché.

```
<div className="ps-2">{car.modele}</div>
<div className="ps-2">Année: {car.annee_circulation}</div>
<div className="ps-2">Kilométrage: {car.kilometrage} Km</div>
<div className="ps-2">Prix: {car.prix} €</div>
<Link className="align-self-center bouton bouton-lien"
to={`/Voiture/${car.id}`}>
Plus d'information
</Link>
```

Fonctionnalité la plus représentative: Autorisation et Authentification

1. Authentification

a. Introduction

L'authentification est un aspect essentiel de la sécurité. L'authentification garantit que seul un utilisateur autorisé peut accéder à des ressources ou des fonctionnalités spécifiques. Dans mon projet, un système d'authentification basé sur les JSON Web Tokens (JWT) a été créé en utilisant la bibliothèque JWT-firebase pour PHP.

Je l'ai installé à l'aide de composer.

```
composer require firebase/php-jwt
```

b. Création (encodage) des JSON Web Tokens (JWT)

J'ai construit mes scripts en suivant les instructions de la bibliothèque.

Le premier script consiste en une classe avec un constructeur initialisant des variables telles que l'heure d'émission, l'heure d'expiration et une clé secrète.

```
public function __construct(){
    date_default_timezone_set('Europe/Paris');
    $this->issuedAt = time();
    $this->expire = $this->issuedAt + 3600;
    $this->jwt_secrect = "clef_secret";
}
```

Une méthode de cette classe permet de générer le token JWT en prenant comme entrées un émetteur ($iss) et les données à y inclure($data).

```
public function jwtEncodeData($iss, $data)
{ .  .  . }
```

Ensuite on crée un token JWT avec les informations appropriées, y compris l'heure d'émission, l'heure d'expiration et les données personnalisées, puis renvoie le token encodé.

```
$this->token = array(
    "iss" => $iss,
    "aud" => $iss,
    "iat" => $this->issuedAt,
    "exp" => $this->expire,
    "data" => $data
);
$this->jwt = JWT::encode($this->token, $this->jwt_secrect, 'HS256');
return $this->jwt;
```

c. Vérification (décodage) des JSON Web Tokens (JWT)

Une autre méthode au sein de cette classe décode le token JWT, utilisant la clé secrète, et vérifie sa validité en consultant l'heure d'expiration.

```
public function jwtDecodeData($jwt_token){
 try {
    $decode = JWT::decode($jwt_token, new Key($this->jwt_secrect ,
'HS256'));
    $currentTimestamp = time();
  if ($currentTimestamp > $decode->exp)
{ . . . }
}
}
```

Si le token est valide, les données qu'il contient sont renvoyées.

```
    echo "<script>localStorage.removeItem('accessToken');</script>";
    throw new ExpiredException('Token has expired');
return [
"data" => $decode->data
];
```

Si le token a expiré, un message d'erreur est généré.

```
catch (ExpiredException $e) {
    return [
        "message" => $e->getMessage()
     ];
} catch (Exception $e) {
     return [
         "message" => $e->getMessage()
];}
```

Pour gérer spécifiquement l'authentification et l'autorisation dans mon application j'ai créé une deuxième qui étend la classe précédente.

```
class Auth extends JwtHandler
{ . . .}
```

Le constructeur hérite de `JwtHandler` et prend en entrée la base de données (`$db`) et les en-têtes de la requête HTTP (`$headers`).

```
    public function __construct($db, $headers)
    {
        parent::__construct();
        $this->db = $db;
        $this->headers = $headers;
```

```
    }
```

Ensuite je crée une méthode de validation du token qui vérifie si un token JWT est présent dans les en-têtes de la requête HTTP.

```
if (array_key_exists('Authorization', $this->headers) &&
preg_match('/Bearer\s(\S+)/', $this->headers['Authorization'],
$matches))
{ . . . }
```

Si un token est trouvé et est valide, elle renvoie un tableau avec le statut de réussite et l'adresse e-mail de l'utilisateur associé au token.

```
$data = $this->jwtDecodeData($matches[1]);
    if (
        isset($data['data']->id) &&
        $email = $this->fetchEmail($data['data']->id)
    ) :
        return [
            "success" => 1,
            "email" => $email
     ];
```

Si le token n'est pas trouvé ou a expiré, elle renvoie un message d'erreur approprié.

```
else :
return [
     "success" => 0,
     "message" => $data['message'],
];
endif;
} else {
return [
     "success" => 0,
     "message" => "Token not found in request"
            ];
        }
```

Autorisation

a. introduction

Après avoir authentifié un utilisateur, il est essentiel de définir les ressources auxquelles il peut accéder. Cette fonction est gérée via une architecture basée sur React.

b. AuthContext et le Provider AuthProvider

Le composant `AuthProvider` enveloppe tout le site web et fournit les données d'authentification aux composants enfants.
Il initialise le contexte d'authentification en fournissant les données d'authentification nécessaires aux composants enfants. Cela garantit que toutes les parties de l'application ont accès aux informations d'authentification lorsque cela est nécessaire.

```
export const AuthProvider = ({ children }) => { . . . }
```

J'ai créé un contexte appelé `AuthContext` pour gérer les informations d'authentification des utilisateurs.
Utilisant la fonction createContext de React, ce contexte permet de partager ces données d'authentification entre différents composants en les passant via les props.

```
const AuthContext = createContext({});
```

Le code tente de récupérer le jeton d'accès (`accessToken`) à partir du stockage local (localStorage).

```
const accessToken = localStorage.getItem('accessToken');
```

Il obtient également le timestamp actuel en secondes à l'aide de

```
const currentTimestamp = Math.floor(Date.now() / 1000);
```

Si un token est disponible et valide, le rôle de l'utilisateur est extrait des données du token avec `jwtDecode`.

```
if (accessToken) {
  try {
    const decodedToken = jwtDecode(accessToken);
    const expToken = decodedToken.exp;
    if (expToken >= currentTimestamp) {
    role = decodedToken.data.role;
}
```

Si le jeton a expiré ou s'il y a une erreur lors du décodage, le code supprime le jeton d'accès du stockage local.

```
else {
    localStorage.removeItem('accessToken');
    }
 } catch (error) {
    console.error("Invalid token:", error);
     localStorage.removeItem('accessToken');
}
```

À l'intérieur de `AuthProvider`, un état local `auth` est initialisé à l'aide de `useState()`.
Cet état contiendra les données d'authentification telles que le jeton d'accès (`accessToken`) et le rôle de l'utilisateur.

```
const [auth, setAuth] = useState(() => { . . .}
```

Enfin, le composant `AuthProvider` rend le contexte d'authentification (`AuthContext.Provider`) en fournissant la valeur `auth` et la fonction `setAuth` comme valeur du contexte.

```
return (
    <AuthContext.Provider value={{ auth, setAuth }}>
      {children}
    </AuthContext.Provider>
  );
```

c. Hook personalisé UseHook

J'ai également développé un hook appelé `useAuth` qui offre un moyen d'accéder facilement aux données d'authentification à partir du contexte `AuthContext`. Cela simplifie l'accès aux informations d'authentification dans les composants React qui en ont besoin.

J'utilise le hook `useContext` pour accéder au contexte d'authentification que j'ai créé précédemment (`AuthContext`). Cela permet de récupérer les données d'authentification (`auth`) ainsi que la fonction `setAuth` qui permet de mettre à jour ces données.

```
 const { auth, setAuth } = useContext(AuthContext);
```

La fonction `setAuthData` est responsable de la mise à jour des données.Elle prend en argument un objet `data` contenant les informations d'authentification à mettre à jour.
Elle utilise `setAuth` pour mettre à jour les données d'authentification avec les nouvelles informations fournies dans `data` et stocke le jeton d'accès (`accessToken`) dans le stockage local (`localStorage`) du navigateur.

```
const setAuthData = (data) => {
    setAuth(data);
    localStorage.setItem('accessToken', data.accessToken);
  };
```

J'ai créé une fonction `logout` est utilisée pour déconnecter l'utilisateur. Elle réinitialise les données d'authentification en les vidant et supprime le jeton d'accès du stockage local.

```
  const logout = () => {
    setAuth({});
    localStorage.removeItem('accessToken');};
```

A la fin le le hook `useAuth` renvoie un objet qui contient les informations d'authentification (`auth`), la fonction pour mettre à jour ces informations (`setAuthData`), et la fonction de déconnexion (`logout`).

```
  return {
    auth: auth,
    setAuth: setAuthData,
    logout: logout,
  };
```

Ces fonctions d'authentification sont essentielles pour la gestion de l'authentification dans notre application React, et elles interagissent étroitement avec le contexte d'authentification pour maintenir un système d'authentification sécurisé et efficace.

d. le RequireAuth

'ai importé un composant `RequireAuth`

```
import RequireAuth from './components/RequireAuth';
```

qui prend une prop `allowedRoles` contenant les rôles autorisés. Ce composant enveloppe les routes auxquelles vous souhaitez limiter l'accès en fonction des rôles.

```
<Route element={<RequireAuth allowedRoles={[ROLE.Employe, ROLE.Admin]}
/>}>
  {/* Routes accessibles aux Employés et Admins */}
</Route>
```

Le composant `RequireAuth` gère les autorisations et les redirections en fonction de l'état d'authentification et des rôles des utilisateurs.

J'importe `useLocation`, `Navigate` et `Outlet` de `react-router-dom`.

```
import { useLocation, Navigate, Outlet } from "react-router-dom";
```

- `useLocation` permet d'obtenir l'objet de localisation actuel.
- `Navigate` est utilisé pour effectuer des redirections
- `Outlet` est utilisé pour afficher les enfants du composant parent en fonction de la structure imbriquée des routes.
-

Le hook `useAuth` pour obtenir l'état d'authentification de l'utilisateur. Cela inclut l'`accessToken` (jeton d'accès) et le `role` (rôle de l'utilisateur) parmi les propriétés de l'objet `auth`.

Si l'`accessToken` n'est pas présent (ce qui signifie que l'utilisateur n'est pas authentifié), vous rediriger l'utilisateur vers la page de connexion en utilisant

```
if (!auth.accessToken) {
        return <Navigate to="/login" state={{ from: location }} replace
/>;
    }
```

Pour permettre à l'utilisateur de revenir à la page d'origine après la connexion, j'utilise l'objet de localisation actuel (`location`).

```
const location = useLocation();
```

Si l'`accessToken` est présent, on vérifie si le `role` de l'utilisateur est inclus dans le tableau `allowedRoles`. Si le rôle de l'utilisateur n'est pas autorisé à accéder à la page actuelle, il est redirigé vers la page d'accès non autorisé

```
    if (!allowedRoles.includes(auth.role)) {
```

```
        return <Navigate to="/unauthorized" state={{ from: location }}
replace />;
    }
```

Si l'utilisateur est authentifié et a un rôle autorisé, il affiche les enfants du composant parent en utilisant `<Outlet />`. Cela permet de rendre les composants enfants correspondant aux routes imbriquées à l'intérieur du composant `RequireAuth`

```
if (!auth.accessToken) {
    {/* . . . */};
}
if (!allowedRoles.includes(auth.role)) {
    {/* . . . */};
}
return <Outlet />
```

Recherche Anglophone

Les deux sites anglophones qui ont le plus d'impact sont Stackoverflow et Youtube.

J'ai utilisé Stack Overflow pour rechercher des réponses à des questions techniques spécifiques liées au développement front-end et back-end de mon projet. Les discussions et les réponses fournies par la communauté m'ont souvent aidé à résoudre des problèmes de codage, à comprendre des concepts difficiles et à optimiser mon code. C'est grave a lui que j'ai trouvé une solution à un problème de gestion des états de mon hook dans la section front-end de mon application grâce à une discussion sur Stack Overflow.

YouTube a été une source précieuse de tutoriels vidéo en anglais pour approfondir mes connaissances techniques et apprendre de nouvelles compétences. J'ai suivi plusieurs chaînes YouTube de développeurs expérimentés qui ont partagé des didacticiels la création d'une API CRUD ce qui ma permit de comprendre pourquoi le mien ne fonctionnait pas comme il faut.. Ces vidéos ont été particulièrement utiles pour moi en tant que développeur visuel, car elles ont montré des exemples concrets et des démonstrations pratiques des concepts que je devais maîtriser pour mon projet

Aussi tout au long de mon projet j'ai utilisé les documentations des différentes Technologies de mon Projet comme le site de Php, des liens Git des bibliothèques, Bootstrap.

Conclusion

Tout au long de ce projet full stack, j'ai développé une application intégrée de bout en bout, depuis la conception initiale jusqu'à sa mise en œuvre complète. Cette démarche m'a permis d'aborder un ensemble diversifié de technologies et de compétences.

L'utilisation de PDO a garanti des opérations de base de données sécurisées et efficaces, prévenant les injections SQL et offrant une interface homogène pour diverses bases de données.

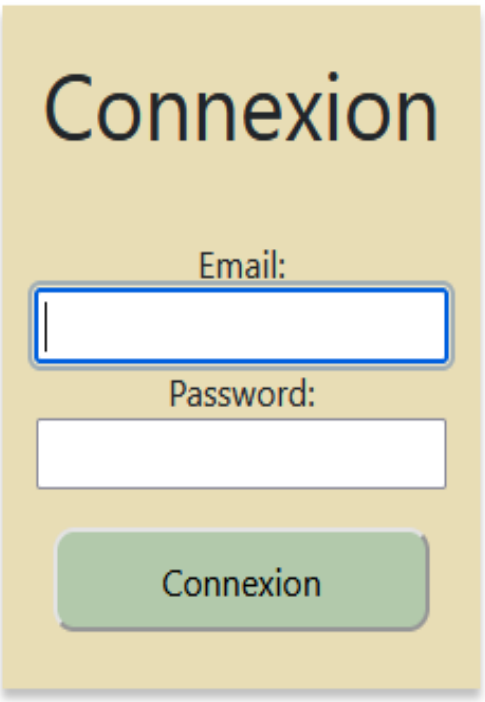
Un des points essentiels de ce projet a été la mise en place d'un système d'authentification et d'autorisation basé sur les JSON Web Tokens (JWT) via la bibliothèque JWT-firebase. Cela a non seulement renforcé la sécurité de notre application, mais aussi assuré que chaque utilisateur accède uniquement aux informations et fonctionnalités qui lui sont destinées.

La création d'une interface utilisateur intuitive grâce à React a été un autre jalon majeur, offrant une interaction simple et engageante pour les utilisateurs finaux.

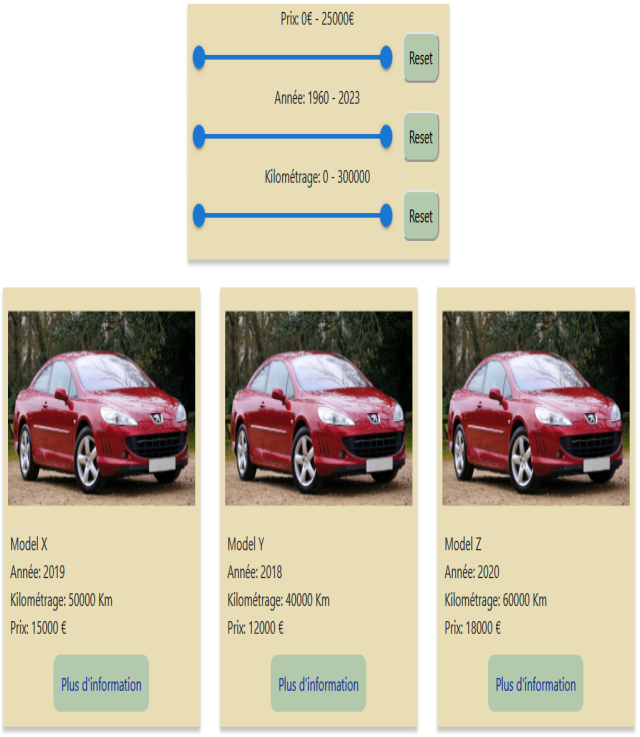
L'achèvement de ce projet a été une expérience incroyablement enrichissante pour moi. Chaque étape, du concept à la mise en œuvre, m'a offert de précieuses leçons. Cela m'a non seulement renforcé en tant que développeur, mais aussi en tant que penseur critique, toujours à la recherche de la meilleure solution pour chaque défi. Je suis fier de ce que j'ai accompli et je suis impatient de mettre en pratique ce que j'ai appris dans mes futurs projets.

Annexe

Ecran de connexion



Filtre de Voiture



Header

Footer