DOSSIER DE PROJET

LE MEUR Erwan

Développeur Web et Web mobile

Sommaire :

Introduction :

Le projet vise à développer une application web en utilisant le serveur MAMP avec PHP 8.0.1, PDO, MySQL, HTML5, CSS3, Bootstrap, JavaScript, jQuery et MySQL. Le projet couvre les compétences du référentiel suivantes :

1. Compétences du référentiel couvertes par le projet

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

- Réaliser une interface utilisateur web statique et adaptable : L'application utilisera HTML5, CSS3 et Bootstrap pour créer une interface utilisateur web statique et adaptable, assurant une expérience utilisateur optimale sur différents appareils.

- Développer une interface utilisateur web dynamique : JavaScript sera utilisé pour rendre l'interface utilisateur dynamique, offrant une expérience interactive aux utilisateurs.

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

- Créer une base de données : La base de données sera créée en utilisant MySQL. Elle stockera les données nécessaires à l'application.

- Développer les composants d'accès aux données : PHP 8.0.1 sous PDO sera utilisé pour développer les composants d'accès aux données, assurant une communication sécurisée avec la base de données.

- Développer la partie back-end d'une application web ou web mobile : Le back-end sera développé en utilisant PHP 8.0.1 et sera responsable de gérer les demandes des utilisateurs, d'accéder à la base de données et de fournir des réponses dynamiques.

Le projet combinera ces compétences pour créer une application web fonctionnelle, sécurisée et adaptative, offrant une expérience utilisateur optimale.

2. Résumé du projet

Ce projet a consisté en la création d'un site web responsive, adapté pour une utilisation sur mobile, destiné à un garage proposant divers services. Les utilisateurs peuvent parcourir les différentes offres, effectuer des recherches pour trouver un véhicule d'occasion, laisser des commentaires et entrer en contact avec le garage. Le site comprend également une version destinée aux employés du garage pour la gestion du contenu, qui s'actualise dynamiquement, ainsi qu'une interface d'administration permettant de modifier, ajouter ou supprimer du contenu, ainsi que de créer de nouvelles sessions pour les employés.

Pour que tout cela fonctionne, il a été nécessaire de connecter le site au back-end via un serveur et une base de données. J'ai choisi d'utiliser PHP en raison de ses PDO (Prepared Data Objects) pour gérer la communication avec la base de données. Les PDO offrent une couche d'abstraction qui facilite la gestion de requêtes SQL, améliorant ainsi la sécurité du programme.

Mesures de Sécurité implémentés pour le projet :

Pour sécuriser l'ensemble du système, de la base de données au site web, j'ai mis en place plusieurs mesures de sécurité :

Utilisation de requêtes préparées : Les PDO permettent de créer des requêtes SQL préparées, ce qui signifie que les données utilisateur sont automatiquement échappées, réduisant ainsi les risques d'injection SQL.

Validation et nettoyage des entrées : Toutes les données provenant des utilisateurs ont été validées et nettoyées pour éliminer les caractères potentiellement dangereux ou inappropriés.

Gestion des sessions : Les sessions utilisateur ont été gérées de manière sécurisée pour empêcher l'accès non autorisé aux zones réservées aux employés et aux administrateurs.

Contrôle d'accès : Des contrôles d'accès appropriés ont été mis en place pour garantir que seuls les employés et les administrateurs autorisés peuvent effectuer des modifications sur le site.

Cryptage des mots de passe : Les mots de passe stockés dans la base de données ont été sécurisés en utilisant des méthodes de hachage cryptographique.

Ce projet a permis de créer un site web complet et sécurisé, offrant une expérience conviviale aux utilisateurs tout en facilitant la gestion du contenu pour les employés et les administrateurs du garage, grâce à l'utilisation de PHP avec des PDO et une attention particulière à la sécurité à chaque niveau du système.

Ce projet m'a permis d'acquérir une précieuse expérience en développement, en travaillant de manière autonome. J'ai rencontré de nombreuses erreurs qui m'ont poussé à effectuer de nombreuses recherches. Je n'avais jusqu'à présent jamais travaillé sur le backend, ce qui a représenté un défi considérable (et quelques nuits blanches).

3. Environnement humain et technique

A. Environnement humain

Actuellement en reconversion professionnelle et en formation, je vis dans une petite ville où j'ai rencontré des difficultés à trouver un stage, malgré mes nombreuses démarches. Les entreprises locales étant souvent de petite taille, elles recherchent des profils plus expérimentés, ce qui ne correspond pas à mon parcours actuel. Mon isolement géographique complique également le travail en équipe, mais je compense en restant connecté avec d'autres professionnels du secteur via des échanges en ligne réguliers

Étant donné que je travaille seul depuis chez moi. Mon principal défi réside dans le développement de compétences en autodidacte, mais je reste ouverte aux opportunités de collaboration à distance pour élargir mon réseau professionnel.

B. Environnement technique

Au quotidien, j'utilise principalement **React Native** avec **Expo** pour le développement d'applications web et mobiles, en associant **Node.js** et **Express** pour le back-end, ainsi que **MySQL** pour la gestion des bases de données. Git et GitHub me permettent de gérer la version et la collaboration.

Cependant, dans ce projet, j'utilise **PHP** avec **PDO** pour interagir avec une base de données **MySQL**, et je travaille localement avec **MAMP**, configuré sur le port 3307 pour MySQL. Cela me permet de gérer efficacement les données et d'assurer une bonne structure côté serveur.

La réalisation du projet

1. Le cahier des charges

Le lien vers le dépôt GitHub public, contenant le code source de l'application.

Le lien vers la version en ligne de l'application web.

Le lien vers Trello.

Un fichier readme.md expliquant le processus d'installation en local, y compris la création d'un administrateur pour le back-office de l'application web.

Les fichiers de création et d'alimentation de la base de données de l'application web (migrations, fixtures, ou scripts SQL à la main).

Une documentation technique au format PDF, comprenant des réflexions initiales sur le projet, les technologies choisies, la configuration de l'environnement de travail, ainsi que des diagrammes (diagramme de classe ou Méthode MERISE, diagramme de cas d'utilisation, diagramme de séquence).

Une charte graphique au format PDF comprenant une palette de couleurs et une sélection de polices d'écriture.

Des maquettes pour le site web, comprenant des wireframes et/ou mockups en vue mobile et desktop.

Description des Fonctionnalités

Le site web du Garage devra inclure les fonctionnalités suivantes :

Se Connecter : Les utilisateurs (administrateur et employés) doivent pouvoir se connecter au site via un formulaire de connexion.

Présenter les Services : Le site doit afficher clairement les services de réparation automobile proposés par le garage, avec la possibilité pour l'administrateur de les modifier depuis son espace d'administration.

Définir les Horaires d'Ouverture : Les visiteurs doivent pouvoir consulter les horaires d'ouverture du garage, gérés par l'administrateur depuis son espace dédié.

Exposer les Voitures d'Occasion : Le site doit présenter les véhicules d'occasion disponibles à la vente, avec des photos, descriptions, informations techniques, et la possibilité pour les employés d'ajouter de nouvelles voitures.

Filtrer la Liste des Véhicules d'Occasion : Les visiteurs doivent pouvoir filtrer la liste des voitures en fonction de critères comme le prix, le kilométrage, ou l'année de mise en circulation.

Permettre de Contacter l'Atelier : Les visiteurs doivent pouvoir contacter le garage par téléphone ou via un formulaire de contact en ligne. Les informations de contact doivent être visibles sur chaque annonce de voiture.

Recueillir les Témoignages des Clients : Le site doit permettre aux visiteurs de laisser des témoignages, avec un commentaire et une note. Les témoignages seront modérés par un employé du garage.

2. Spécifications techniques

Les spécifications techniques incluent les langages et technologies à utiliser, tels que :

Serveur : MAMP, PHP 8.0.1,

Database Serveur : MySQL 5.7.24

Front-end : HTML 5, CSS 3, Bootstrap, JavaScript

Back-end : PHP 8.2 avec PDO, MySQL

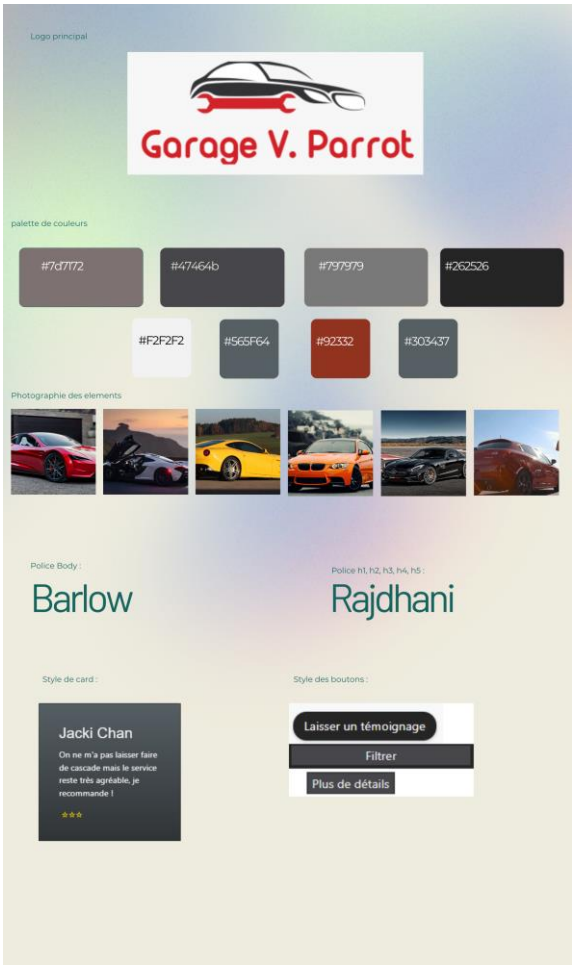
SGBDR *(Système de Gestion de Base de Données Relationnelle)* basé sur le langage SQL *(Structured Query Language)*.

Diagrammes

Les diagrammes de cas d'utilisation, de séquence et de classe (ou Méthode MERISE) doivent être élaborés pour documenter le projet.

Charte Graphique

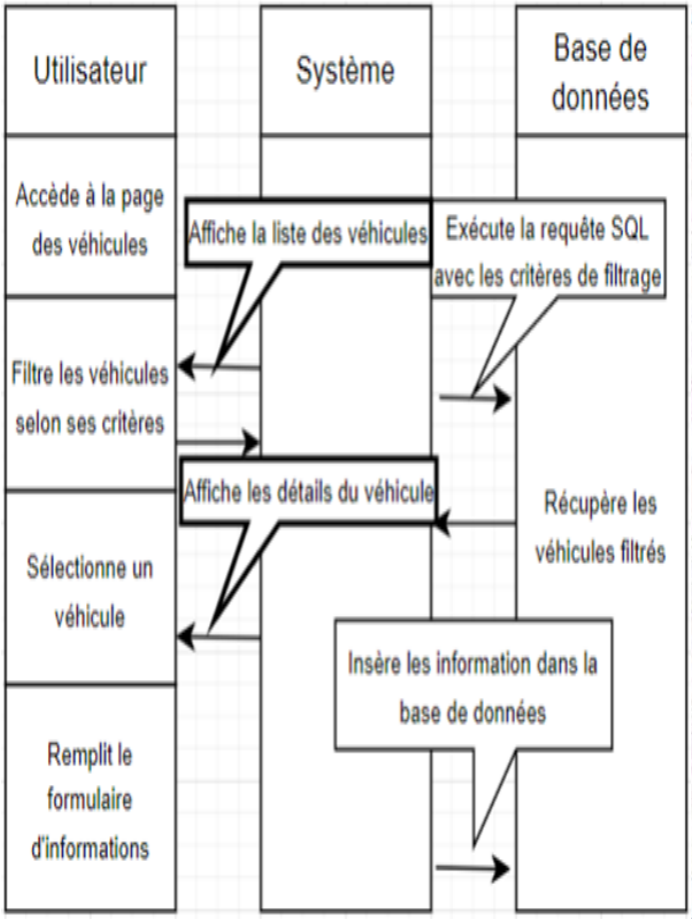
Une charte graphique doit inclure une palette de couleurs, une sélection de polices d'écriture, et des maquettes pour le site web, en vue mobile et desktop.



Ce cahier des charges définit le cadre du projet du Garage et les exigences techniques, fonctionnelles et de livrables attendues pour la réalisation de ce projet.

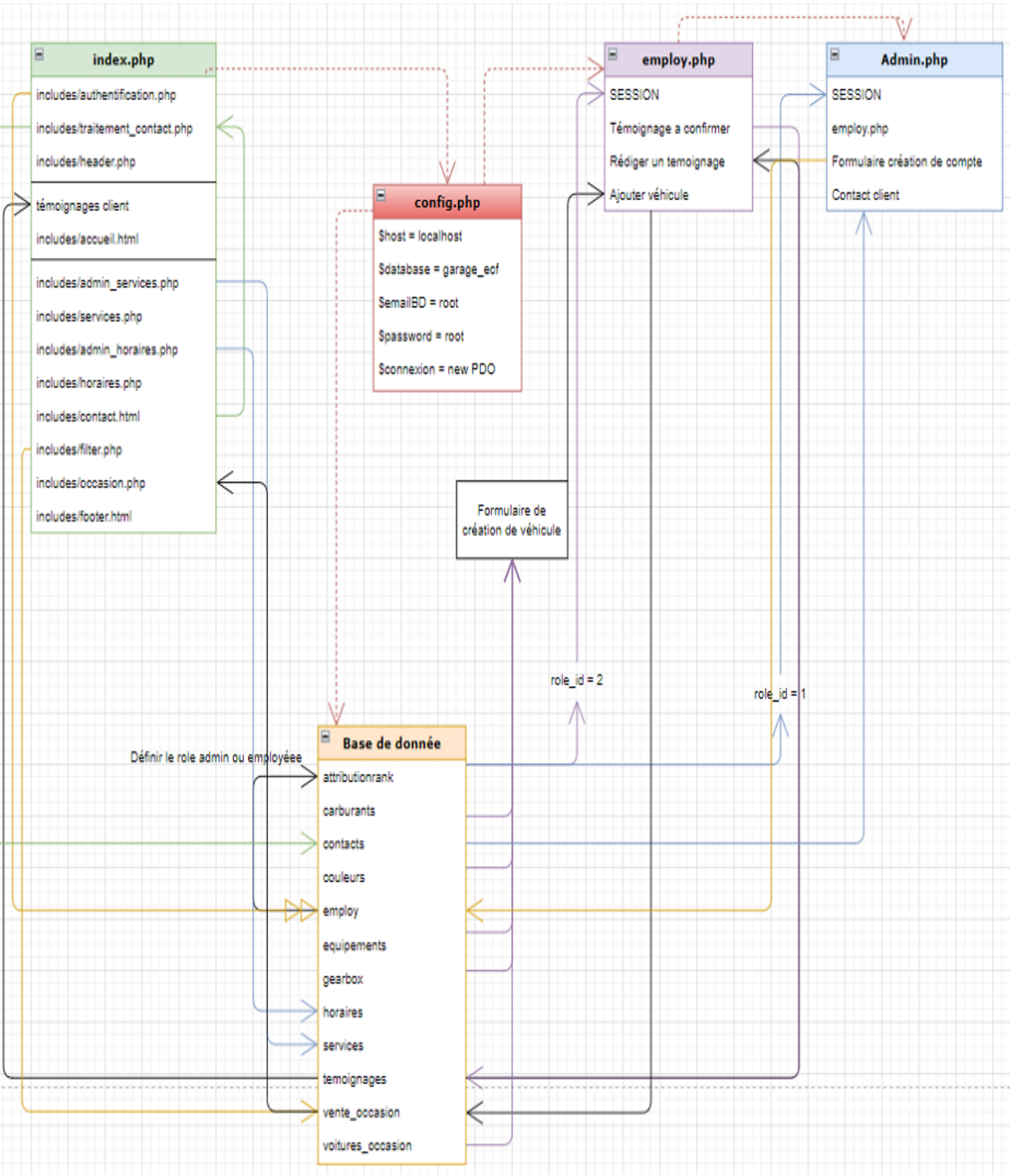
3. Conception de la base de données

J'ai initié le processus en élaborant un système d'interaction *(image de gauche)* entre l'utilisateur, le système, et la base de données concernant la gestion des véhicules., ce qui représente une étape fondamentale dans le développement de mon application. Cela synthétise de manière claire et structurée les besoins essentiels de mon application, aidant ainsi à définir les relations entre les différentes entités et à établir un cadre solide pour la conception de la base de données.



Voici à droite un diagramme de classe basé sur le système de gestion de véhicules présenté dans l'image précédente. Il montre les interactions entre l'utilisateur, le système, et la base de données, avec des méthodes associées à chaque classe.

Pour déterminer précisément les besoins de ma base de données, j'ai élaboré en amont un diagramme MLD (Modèle Logique de Données) qui a connu une évolution significative au cours de l'avancement du projet. Ce diagramme interconnecte chaque fichier créé, décrivant de manière claire et précise les exigences de ma base de données. Le résultat final se présente comme suit



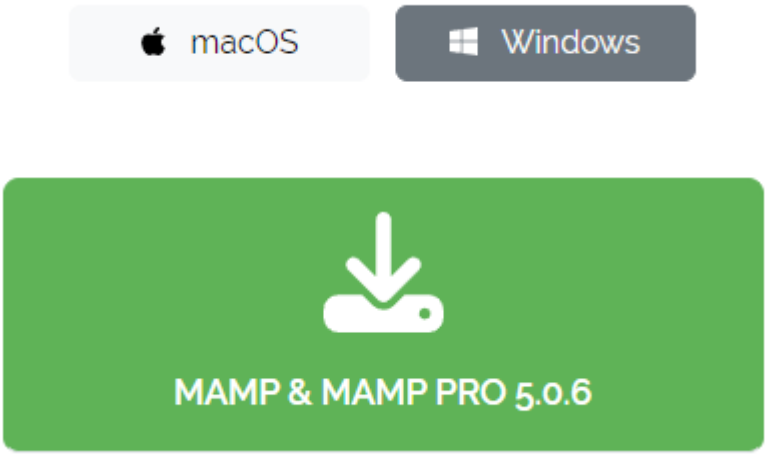
4. Développement de l'API

A. Installation

Pour installer le développement de l'API basée sur les technologies que j'ai mentionnées plus haut (MAMP, PHP 8.0.1, PDO MySQL, HTML5, CSS3, Bootstrap, JavaScrip, jQuery, et MySQL), j'ai suivi certaines étapes :

Étape 1 : Configuration de l'environnement de développement

Installation de MAMP (ou WAMP pour Windows, LAMP pour Linux). Cet ensemble logiciel permet d'exécuter un serveur web Apache, une base de données MySQL et PHP.



Étape 2 : Configuration de la base de données

J'ai ensuite lancer MAMP pour démarrer les services Apache et MySQL.

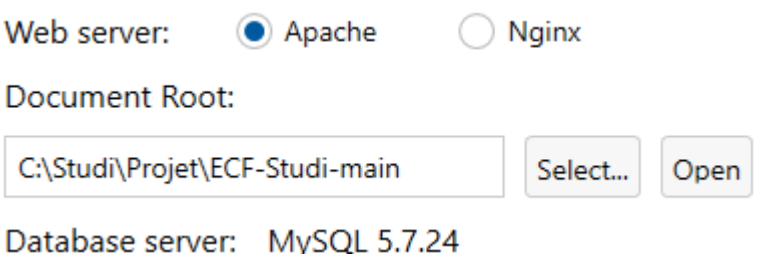
Puis sur phpMyAdmin (généralement via http://localhost/phpmyadmin/) ou MySQL Workbench (un des plus connus), pour créer une nouvelle base de données pour l'application. Importer également le schéma de ma base de données.

```
-- Création de la base de données
CREATE DATABASE IF NOT EXISTS garage_ecf;
USE garage_ecf;
```

Étape 3 : Création du répertoire du projet

Pour la création du répertoire de projet j'ai créé mon dossier à la racine du serveur web dans le dossier "htdocs" pour MAMP.

(Il est également possible de changer le "document root" de MAMP dans ces préférence)



Étape 4 : Configuration de PHP pour le back-end

Pour finaliser la mise en place du back-end, j'ai créé un nouveau fichier PHP dans le répertoire du projet, utilisant la version PHP 8.0.1. Ce fichier sert à établir la connexion entre l'application et la base de

données via PDO (PHP Data Objects) pour MySQL, garantissant une gestion plus sécurisée et performante des requêtes.

Dans ce fichier, j'ai activé l'affichage des erreurs en phase de développement pour faciliter le débogage avec les paramètres `ini_set()` et `error_reporting()`. J'ai ensuite configuré une connexion à la base de données `garage_ecf` en spécifiant le port `3307` (propre à ma configuration MAMP) et l'encodage UTF-8 (`utf8mb4`) pour assurer la compatibilité avec les caractères spéciaux.

J'ai également utilisé plusieurs options PDO essentielles :

- `PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION` pour activer la gestion des exceptions en cas d'erreur,
- `PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC` pour récupérer les résultats sous forme de tableau associatif par défaut,
- `PDO::ATTR_EMULATE_PREPARES => false` pour désactiver l'émulation des requêtes préparées, renforçant ainsi la sécurité.
-

En cas de problème de connexion, un message d'erreur est enregistré dans un fichier log spécifique, `db_error.log`, pour un suivi détaillé, tout en affichant un message d'erreur générique à l'utilisateur pour éviter la fuite d'informations sensibles.

```php
<?php
// Activer la gestion des erreurs en développement
ini_set('display_errors', 1);
error_reporting(E_ALL);

try {
    $connection = new PDO("mysql:host=localhost;port=3307;dbname=garage_ecf;charset=utf8mb4", "root", "root", [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
        PDO::ATTR_EMULATE_PREPARES => false
    ]);
} catch (PDOException $e) {
    error_log("Erreur de connexion : " . $e->getMessage(), 3, __DIR__ . '/logs/db_error.log');
    die("Erreur de connexion à la base de données.");
}
?>
```

Étape 5 : Test de l'API

Pour m'assurer du bon fonctionnement de l'API dans mon projet, j'ai réalisé plusieurs vérifications :

1. **Connexion à la base de données** :

    - J'ai testé la connexion pour m'assurer qu'elle s'établit sans erreur.

- J'ai vérifié que les erreurs de connexion sont correctement enregistrées dans le fichier `db_error.log`.

2. **Tests des endpoints** :

   - **Création (POST)** : J'ai vérifié que les nouvelles entrées se créent correctement avec des données valides et que les erreurs sont gérées pour des données invalides.
   - **Récupération (GET)** : J'ai testé la récupération des données pour m'assurer qu'elle fonctionne avec des paramètres valides et que les erreurs sont gérées pour les paramètres invalides.
   - **Mise à jour (PUT/PATCH)** : J'ai vérifié que les mises à jour se déroulent bien avec des données valides et que les erreurs sont gérées pour les identifiants non existants.
   - **Suppression (DELETE)** : J'ai testé la suppression d'entrées et vérifié la gestion des erreurs pour les identifiants non existants.

3. **Sécurité** :

   - J'ai testé l'authentification pour les endpoints protégés et vérifié que l'accès non autorisé renvoie un code d'erreur approprié.
   - J'ai réalisé des tests pour détecter des injections SQL et m'assurer que l'API reste sécurisée.

4. **Performances** :

   - **Tests de performance** : Évaluer la réactivité de l'API sous différentes charges de requêtes pour s'assurer qu'elle peut gérer le trafic prévu sans ralentissement.
   - **Tests de temps de réponse** : Mesurer le temps de réponse pour chaque endpoint afin de s'assurer qu'il est dans les limites acceptables.

Ces tests m'ont permis de valider le bon fonctionnement et la sécurité de l'API dans mon projet.

B. Initialisation du projet

Au début de mon projet, j'ai pris le temps de définir clairement mes objectifs. J'ai réfléchi aux fonctionnalités clés que je voulais intégrer à mon application web, ainsi qu'aux besoins des utilisateurs finaux. Cette phase de réflexion m'a permis d'établir une base solide pour le développement ultérieur.

Pour la structure de mon application, j'ai d'abord créé le fichier `index.php`, qui sert de point d'entrée principal pour l'application. Par la suite, j'ai organisé la structure de mon projet en créant des dossiers pour regrouper logiquement mes fichiers. Cela inclut la création d'un fichier `employ.php` et d'un fichier `admin.php` à la racine du projet.

J'ai également mis en place plusieurs dossiers :

- **css** : pour stocker tous mes fichiers de styles.
- **images** : pour regrouper les ressources graphiques de l'application.
- **includes** : pour les fichiers de configuration et les scripts PHP inclus.
- **js** : pour tous les fichiers JavaScript nécessaires au bon fonctionnement de mon application.

Cette structure m'a permis de gérer plus efficacement mes ressources et de faciliter le développement à mesure que j'ajoutais de nouvelles fonctionnalités.

C. Développement de la base de données

Le développement de ma base de données a été une étape vraiment importante dans mon projet. En me basant sur ce que j'ai appris, j'ai créé une structure solide avec MySQL pour stocker et gérer toutes les infos nécessaires à mon application de gestion de garage.

D'abord, j'ai mis en place la base de données `garage_ecf`, qui contient plusieurs tables qui sont liées entre elles. J'ai fait attention à définir des contraintes d'intégrité pour garantir que les données restent cohérentes et fiables. Par exemple, la table `attributionrank` gère les rôles des utilisateurs, comme Administrateur et Employé, ce qui permet de contrôler l'accès de manière précise.

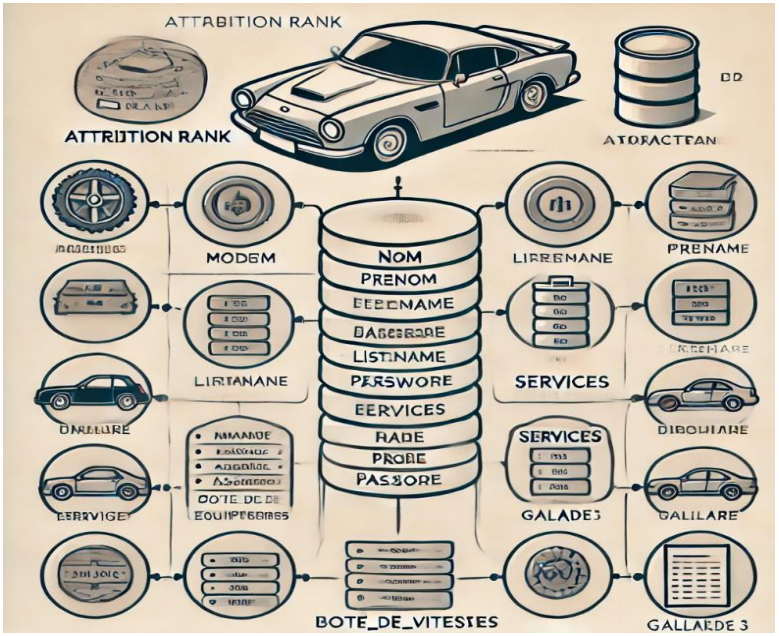
Ensuite, j'ai créé la table `contacts`, qui collecte les informations des clients, avec des champs pour le nom, le prénom, l'email et le message. Ça facilite le suivi des demandes tout en évitant les doublons grâce à des contraintes d'unicité.

La gestion des véhicules d'occasion se fait dans la table `vente_occasion`, où je stocke des infos essentielles comme la marque, le modèle, le prix et l'année de fabrication. Chaque véhicule peut également avoir des images pour mieux les présenter.

J'ai veillé à établir des relations entre les tables, par exemple, en reliant la table `employ` à `attributionrank` grâce à une clé étrangère. Cela garantit que chaque employé est bien associé à son rôle dans l'application.

Enfin, pour rendre les insertions de données plus efficaces, j'ai intégré des optimisations comme `INSERT IGNORE` et `ON DUPLICATE KEY UPDATE`. Cela m'aide à éviter les doublons tout en assurant que chaque entrée est unique.

En résumé, j'ai conçu cette base de données non seulement pour qu'elle fonctionne bien, mais aussi pour qu'elle puisse évoluer avec les besoins de l'application. Cela assure une gestion fluide et efficace des infos essentielles pour le fonctionnement du garage.



D. Développement des routers et controllers

Pour assurer une gestion efficace des requêtes et des réponses de l'application, j'ai conçu les routes et les contrôleurs de mon projet en utilisant PHP avec PDO pour la communication entre le front-end et le back-end.

Le fichier `config.php` joue un rôle clé en tant que point d'entrée vers la base de données. Il configure la connexion à MySQL, gère les erreurs et définit les paramètres de récupération des données, garantissant ainsi une interaction sécurisée avec la base de données.

L'élément central de mon application est le fichier `index.php`, qui intègre divers composants via des instructions `include`. Cela permet de structurer l'application de manière modulaire, facilitant l'ajout ou la modification de fonctionnalités. À travers ce fichier, j'ai pu créer une interface utilisateur dynamique où les témoignages, les services et les horaires sont récupérés en temps réel depuis la base de données.

Pour la gestion des données, j'ai mis en place un système de routage basé sur des fichiers séparés qui traitent les requêtes spécifiques. Par exemple, le fichier `ajouter_voiture.php` permet aux employés d'ajouter des véhicules d'occasion à la base de données. Lorsque l'utilisateur soumet le formulaire, les données sont traitées et insérées dans la table `vente_occasion` via une requête préparée, ce qui protège l'application contre les injections SQL.

Chaque route est conçue pour correspondre à une action spécifique, permettant ainsi de gérer les opérations CRUD de manière claire et organisée. Grâce à ce système, la communication entre le front-end et le back-end est fluide, assurant un fonctionnement optimal de l'application tout en maintenant une bonne structure de code.

5. Développement de la partie front-end

A. Mise en place de l'environnement de développement

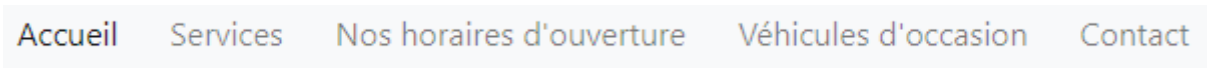
Pour créer l'interface utilisateur du site Garage V. Parrot, j'ai mis en place un environnement de développement complet et optimisé. J'ai utilisé plusieurs technologies pour assurer une expérience utilisateur fluide et agréable :

- **HTML5** pour structurer le contenu de façon claire,
- **CSS3** pour personnaliser le style et créer une mise en page harmonieuse,
- **Bootstrap** (v5.2.3) pour bénéficier d'une grille flexible et de composants réactifs,
- **JavaScript** et **jQuery** pour ajouter des fonctionnalités interactives.

J'ai opté pour l'utilisation de bibliothèques via des **CDNs** afin de faciliter les mises à jour et éviter la gestion de fichiers locaux, ce qui rend le processus plus léger et efficace.

B. La navigation au sein de l'application

Pour la navigation, j'ai veillé à offrir une expérience fluide et intuitive. J'ai intégré des menus, boutons, et liens permettant de passer aisément d'une section à l'autre. J'ai également fait en sorte que la navigation soit cohérente sur toutes les pages, afin que les utilisateurs puissent se repérer et explorer le site sans difficulté.



Les différentes sections incluent une page d'accueil qui présente le garage et ses services, un espace contact pour permettre aux utilisateurs de nous joindre facilement, et une section dédiée aux véhicules d'occasion. Un

carousel a aussi été mis en place pour afficher les témoignages clients, apportant ainsi une touche dynamique et crédible à l'expérience globale.

Pour garantir que le site soit accessible sur tous types de supports, j'ai conçu un **header** qui s'adapte aux écrans plus restreints. L'interface est pensée de façon à ce que les utilisateurs mobiles puissent naviguer aisément, grâce à un menu responsive qui affiche les sections principales du site.

C. Les fonts

Le choix des polices de caractères a été effectué avec soin pour améliorer la lisibilité et l'esthétique de l'application. J'ai sélectionné des polices compatibles avec les standards web et j'ai géré leur intégration de manière à ce qu'elles s'affichent correctement sur différentes plateformes et navigateurs.

D. Les contextes

Contexte général

Ce projet est issu d'un ancien exercice où certaines fonctionnalités m'avaient été imposées, ainsi que le thème d'un site vitrine pour un garage automobile. J'ai choisi de le présenter ici car il offre une base solide pour démontrer mes compétences techniques. Le site permet la gestion des véhicules d'occasion, l'affichage des témoignages, et la mise en avant des services du Garage V. Parrot.

Contexte technique

Le projet a été développé principalement en PHP pour la gestion côté serveur, avec une base de données MySQL pour stocker les informations sur les véhicules, les témoignages et les services. JavaScript a été utilisé côté client pour améliorer l'interactivité du site. Si je devais refaire ce projet aujourd'hui, avec l'expérience acquise, je l'implémenterais probablement avec React pour le front-end et Node.js pour le back-end, mais ici j'ai utilisé du JavaScript pur avec PHP assurant la connexion client-serveur.

Contexte fonctionnel

Le site permet aux utilisateurs de contacter le garage, de consulter les horaires, de laisser des témoignages et de chercher un véhicule via un système de filtre.

- **Clients** : peuvent parcourir les véhicules, consulter les services, et déposer des avis.
- **Employés** : peuvent assister les clients en soumettant des témoignages ou en ajoutant de nouveaux véhicules.
- **Administrateurs** : ont un contrôle total sur le contenu affiché, avec la possibilité de modifier les horaires, les services, les véhicules, et les témoignages.

Contexte organisationnel

Je travaille seul sur ce projet, généralement en soirée après le travail. J'utilise Git pour la gestion de version et MAMP pour tester en local. Aujourd'hui, je suis plus familier avec Node.js, mais ce projet utilise une architecture classique basée sur PHP et JavaScript.

E. Écrans développés

Au cours de cette étape, j'ai développé l'écran principal de l'application, en restant fidèle à la demande initiale de l'exercice, qui était de créer un site vitrine. L'idée était de garder une interface simple et fonctionnelle, sans développer plusieurs écrans, afin de se concentrer sur la présentation du contenu de manière claire et accessible.

Pour ce faire, j'ai utilisé les technologies front-end que j'avais sélectionnées, principalement en PHP et Bootstrap, afin de structurer les différentes sections du site : témoignages, services, horaires, et formulaire de contact. J'ai également intégré un carrousel pour afficher les témoignages clients, ainsi qu'une partie dédiée aux services et horaires, tout en assurant une navigation fluide.

L'ensemble du projet respecte un design adaptatif, permettant une bonne expérience utilisateur, que ce soit sur ordinateur ou appareil mobile. En développant cet écran, j'ai cherché à offrir une interface intuitive qui s'adapte aux besoins des utilisateurs tout en mettant en avant les services proposés par le garage.

Un axe d'amélioration évident pour ce projet réside dans l'organisation des sections de l'interface. Actuellement, tout est présenté sur une seule page, ce qui est efficace pour un site vitrine simple. Cependant, sans la contrainte de créer un site vitrine, j'aurais opté pour une structuration plus avancée, notamment en séparant certaines fonctionnalités dans des écrans dédiés. Par exemple, le formulaire de contact aurait pu être déplacé vers une page distincte, ce qui permettrait de le mettre davantage en valeur et d'offrir une meilleure navigation.

Cette approche aurait non seulement allégé l'écran principal, mais aussi amélioré l'expérience utilisateur en rendant l'interface plus lisible et organisée. Une telle répartition aurait également facilité l'intégration de nouvelles fonctionnalités ou informations à l'avenir, rendant le site plus évolutif.

Fonctionnalité la plus représentative : L’authentification

Dans mon projet de site web en PHP, j'ai choisi de mettre en avant la fonctionnalité d'authentification, car elle représente un aspect clé de la gestion des utilisateurs.

Présentation de la fonctionnalité d'authentification

1. **Point de départ : index.php** Mon projet commence par la page index.php, qui sert de point d'entrée principal pour les utilisateurs. Cette page inclut divers composants comme le header, les témoignages, et les sections horaires et services. C’est également à partir de cette page que l’utilisateur peut accéder à la fonctionnalité d'authentification via le fichier authentification.php.

2. **Le formulaire de connexion (authentification.php)** Sur cette page, je propose un formulaire où l'utilisateur doit entrer son adresse e-mail et son mot de passe. Une fois les informations soumises, elles sont envoyées au serveur pour être vérifiées.

3. **Vérification des identifiants** En arrière-plan, une requête SQL est exécutée pour récupérer l'utilisateur correspondant à l'email fourni. Si l'utilisateur existe, je compare le mot de passe saisi à celui stocké dans la base de données en utilisant password_verify, qui me permet de sécuriser les mots de passe via un système de hachage.
4. **Gestion des rôles** Après cette vérification, si les identifiants sont corrects, je démarre une session pour l'utilisateur. Je stocke alors des informations telles que son email, son rôle (administrateur ou employé), ainsi que son statut de connexion. J’utilise l’ID de rôle pour déterminer les droits de chaque utilisateur :
    - Un ID de rôle égal à 1 correspond à un **administrateur**.
    - Un ID de rôle égal à 2 correspond à un **employé**.

5. **Redirection en fonction du rôle**

    - Si l'utilisateur est un **administrateur**, je le redirige vers admin.php, une page qui contient les outils de gestion du site.
    - Si c'est un **employé**, il est redirigé vers employ.php, où il peut accéder aux fonctionnalités qui lui sont destinées.

6. **Gestion des erreurs** Si l’authentification échoue, que ce soit à cause d'un email inexistant ou d'un mot de passe incorrect, l'utilisateur est renvoyé vers index.php. J’ai également prévu la possibilité d'ajouter un message d'erreur pour informer l’utilisateur de la raison de l’échec.

Conclusion

J’ai choisi de mettre en avant cette fonctionnalité d'authentification car elle démontre l'importance de la sécurité et de la gestion des utilisateurs sur un site web. Ce système assure que seuls les utilisateurs ayant les bonnes autorisations peuvent accéder aux parties sensibles du site, tout en gérant efficacement les rôles grâce à PHP et aux bases de données.

En utilisant password_verify pour vérifier les mots de passe et en intégrant des sessions pour stocker les informations de connexion, j’assure une sécurité renforcée pour les utilisateurs et leurs données.

1. Jeu d’essai

1. Tester la navigation et le responsive design

7. **Objectif :** Vérifier que la navigation entre les différents écrans fonctionne correctement et que la mise en page est adaptée aux petites résolutions.
8. **Étapes :**
    - Accéder à la **page d'accueil (index.php)** depuis un navigateur.
    - Naviguer vers les différents écrans (services, horaires, témoignages, etc.).
    - Redimensionner la fenêtre du navigateur pour vérifier l’adaptabilité aux résolutions mobiles.
9. **Résultat attendu :**
    - La navigation doit être fluide et aucun élément de la mise en page ne doit être déformé ou caché lors du changement de taille d'écran.

2. Tester l’authentification réussie

10. **Objectif :** Vérifier que l’authentification fonctionne correctement et redirige les utilisateurs vers les pages appropriées en fonction de leur rôle.
11. **Étapes :**
    - Sur la **page de connexion (authentification.php)**, entrer un **email valide** et le **mot de passe correct** d’un **administrateur**.
    - Valider le formulaire.
12. **Résultat attendu :**
    - L'utilisateur est redirigé vers la page **admin.php** s'il est administrateur, ou vers **employ.php** s'il est employé.

3. Tester l’erreur d’authentification

13. **Objectif :** Vérifier que le système d'authentification renvoie un message d’erreur en cas de mauvaises informations et ne permet pas l'accès aux pages administratives.
14. **Étapes :**
    - Sur la **page de connexion**, entrer un **email valide** mais un **mot de passe incorrect**.
    - Valider le formulaire.
15. **Résultat attendu :**
    - Un message d’erreur s’affiche avec le texte : "Erreur : identifiant ou mot de passe incorrect."
    - L'utilisateur reste sur la page de connexion sans être redirigé.

4. Tester la déconnexion et la gestion de session

16. **Objectif :** Vérifier que la déconnexion fonctionne et que la session est correctement détruite.
17. **Étapes :**
    - Après une connexion réussie, accéder à la **page d’administration** ou à la **page employé**.
    - Tester le bouton ou lien de **déconnexion** pour détruire la session.
    - Essayer d'accéder directement à la page admin ou employé sans se reconnecter.
18. **Résultat attendu :**
    - Après la déconnexion, l’utilisateur doit être renvoyé vers la **page de connexion** s’il essaie d’accéder aux pages réservées.

2. Recherches anglophones sur le JWT

Points Pertinents sur JWT :

19. **Qu'est-ce que JWT ?**
    - JWT est un standard ouvert (RFC 7519) qui définit un moyen compact et autonome pour transmettre des informations entre parties sous forme d'objet JSON. Ces informations peuvent être vérifiées et signées.
20. **Structure d'un JWT :**
    - Un JWT se compose de trois parties :
        - **Header** : Contient des informations sur la façon de traiter le token (algorithme de signature, type de token).
        - **Payload** : Contient les données (claims) à transmettre, comme l'identifiant de l'utilisateur et les rôles.
        - **Signature** : Permet de vérifier que le message n'a pas été modifié.
21. **Authentification et Autorisation :**
    - JWT est souvent utilisé pour l'authentification dans les applications web et mobiles. Une fois qu'un utilisateur se connecte, le serveur génère un JWT qui est renvoyé au client. Le client stocke ce token et l'envoie avec chaque requête subséquente pour accéder aux ressources protégées.
22. **Avantages de JWT :**
    - **Stateless** : JWT ne nécessite pas de stockage côté serveur, ce qui le rend idéal pour les applications scalables.
    - **Inter-opérabilité** : JWT est utilisé dans de nombreux langages de programmation et frameworks, facilitant l'intégration avec différents systèmes.
23. **Sécurité des JWT :**
    - Bien qu'il soit pratique, il est essentiel de bien gérer les clés de signature et de ne pas stocker d'informations sensibles dans le payload, car il peut être décodé par n'importe qui ayant accès au token.
24. **Utilisation de JWT avec des rôles :**
    - Dans ton projet, tu peux utiliser des JWT pour gérer les différents rôles d'utilisateurs (administrateur, employé, vétérinaire). Les claims dans le payload peuvent contenir des informations sur le rôle de l'utilisateur, ce qui simplifie le contrôle d'accès.

Ressources Anglophones :

A JSON Web Token (JWT) is a compact and secure method for transmitting authentication and authorization information between two parties. It consists of three main components: the **header**, the **payload**, and the **signature**. The header typically includes metadata about the token type and the cryptographic algorithm used. The payload contains claims or statements about the user or entity, and the signature is generated to verify that the token has not been altered.

JWTs are particularly useful in web applications for several reasons. They help prevent issues like data tampering and session hijacking, and can also facilitate single sign-on (SSO) capabilities. When a user logs in, the server generates a JWT with the user's information, which the client then includes in subsequent requests to access protected resources. This self-contained nature of JWTs eliminates the need for the server to maintain session states, making applications more scalable (F22 Labs) (SOPHOS).

For a deeper dive into JSON Web Tokens, including their structure, security benefits, and examples of usage, you can refer to the full article by Sophos. The author is Sophos, a company specializing in cybersecurity solutions.

Conclusion

Au cours de ce projet, j'ai appris énormément en passant par différentes phases de tests, d'ajustements, et de recherches. J'ai commencé par maîtriser les bases du développement web avec du HTML et du CSS pour styliser mon site. Par la suite, j'ai intégré PHP afin de gérer la communication avec le backend et permettre une interactivité plus avancée.

Pour m'organiser, j'ai d'abord créé des maquettes et des wireframes sur Figma. Ces outils m'ont permis de visualiser l'apparence du site et de structurer son fonctionnement avant de passer à la phase de développement. À ce sujet, la différence entre une maquette et un wireframe est simple : le wireframe est une esquisse de la structure du site, tandis que la maquette intègre les aspects visuels et la charte graphique. Cela m'a aidé à définir un style cohérent et à structurer le site selon mes attentes.

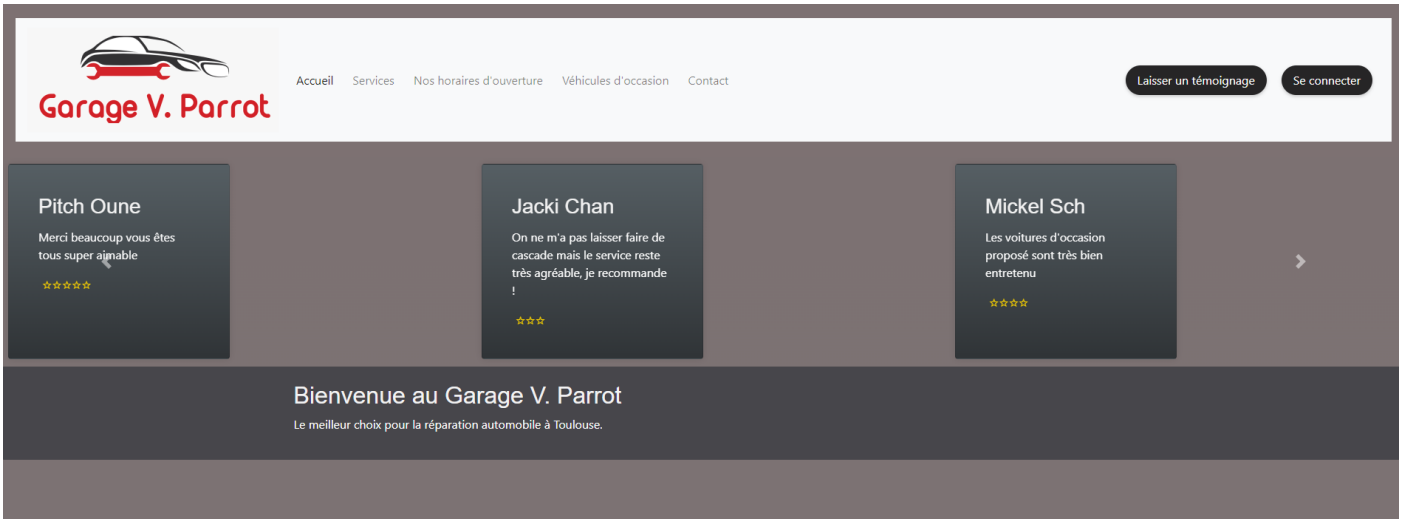
J'ai ensuite organisé mon projet sur VS Code en séparant les différents fichiers selon leur fonction : un dossier pour les fichiers CSS, un autre pour les images, un pour les fichiers PHP « includes », et un pour le JavaScript. Au début, j'avais un fichier index.html, que j'ai plus tard renommé en index.php pour intégrer des fonctionnalités dynamiques. Progressivement, j'ai ajouté des pages comme admin.php et employ.php à la racine du projet.

Le projet est devenu une plateforme web dynamique permettant de présenter des véhicules d'occasion. Les administrateurs et employés peuvent se connecter pour ajouter des véhicules et des témoignages, ainsi que gérer les horaires directement côté client grâce à une interface simple. J'ai également intégré un système de filtre pour permettre aux utilisateurs de rechercher des véhicules selon des critères spécifiques. L'ensemble des informations est géré via une base de données, permettant une gestion souple et flexible des services et des contenus.

Durant le développement, j'ai rencontré divers bugs qui m'ont conduit à explorer des forums comme Stack Overflow pour trouver des solutions, ce qui m'a permis de progresser et d'affiner mes compétences. Grâce à cette expérience, j'ai appris à mieux structurer un projet web complet, de la conception initiale à la mise en œuvre technique.

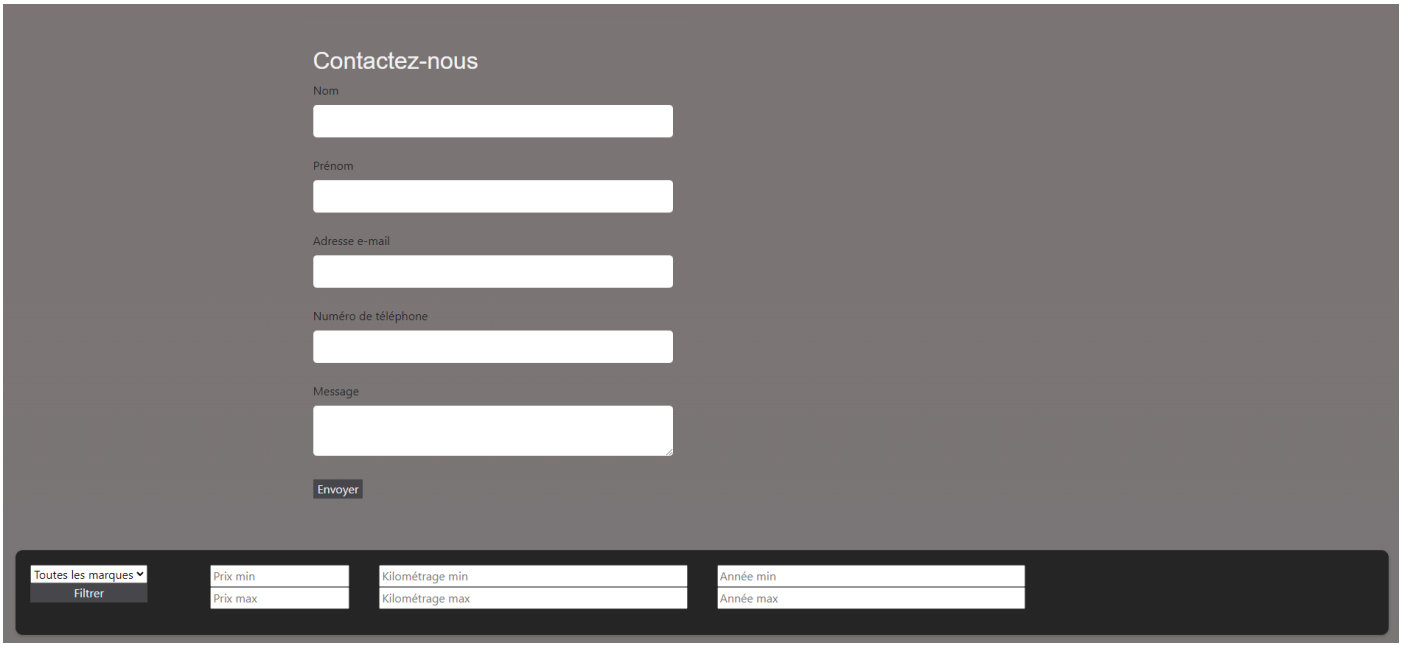
En conclusion, ce projet m'a permis de consolider mes compétences techniques et d'aborder des problématiques concrètes liées au développement d'applications web dynamiques.

Annexes

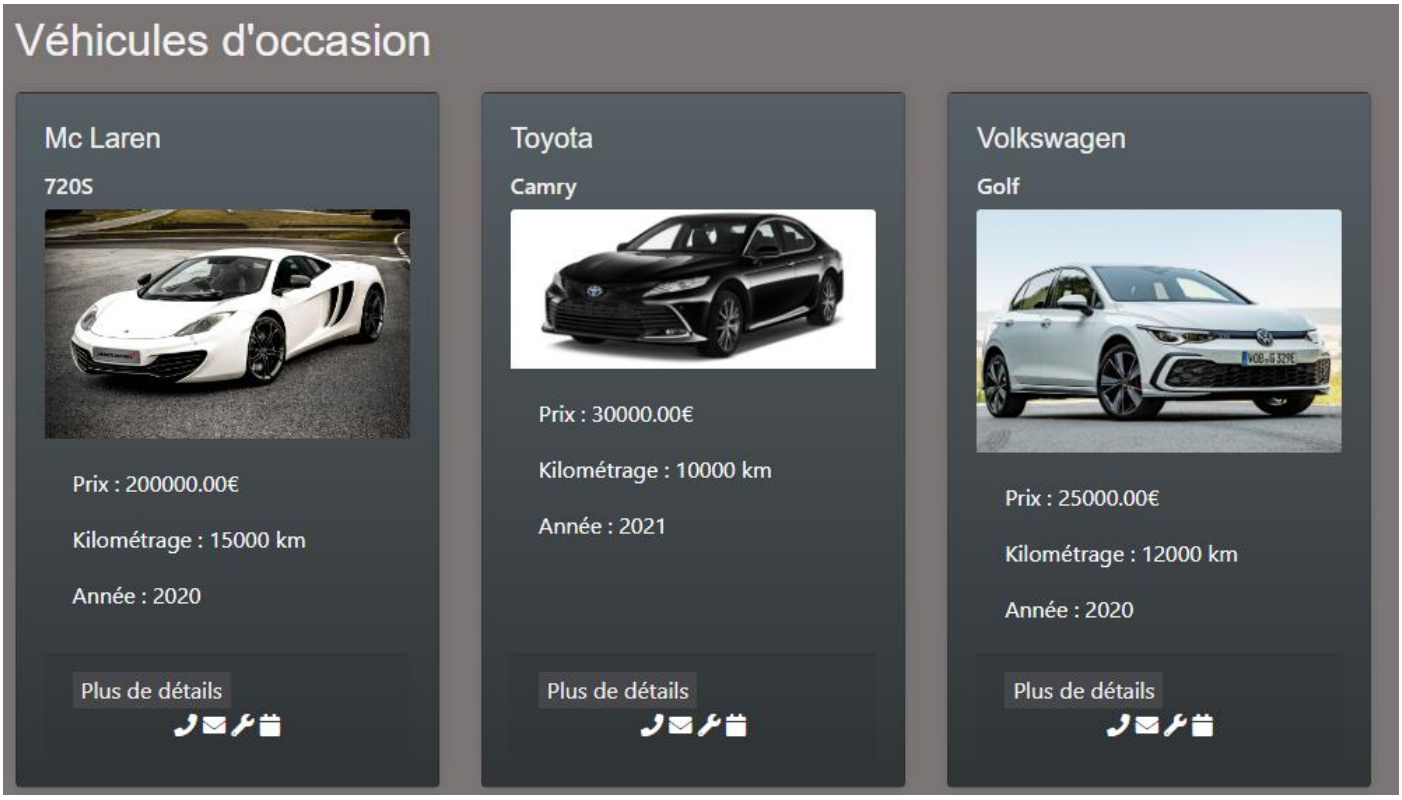


(Voici le haut de ma page d'accueil de mon site web)

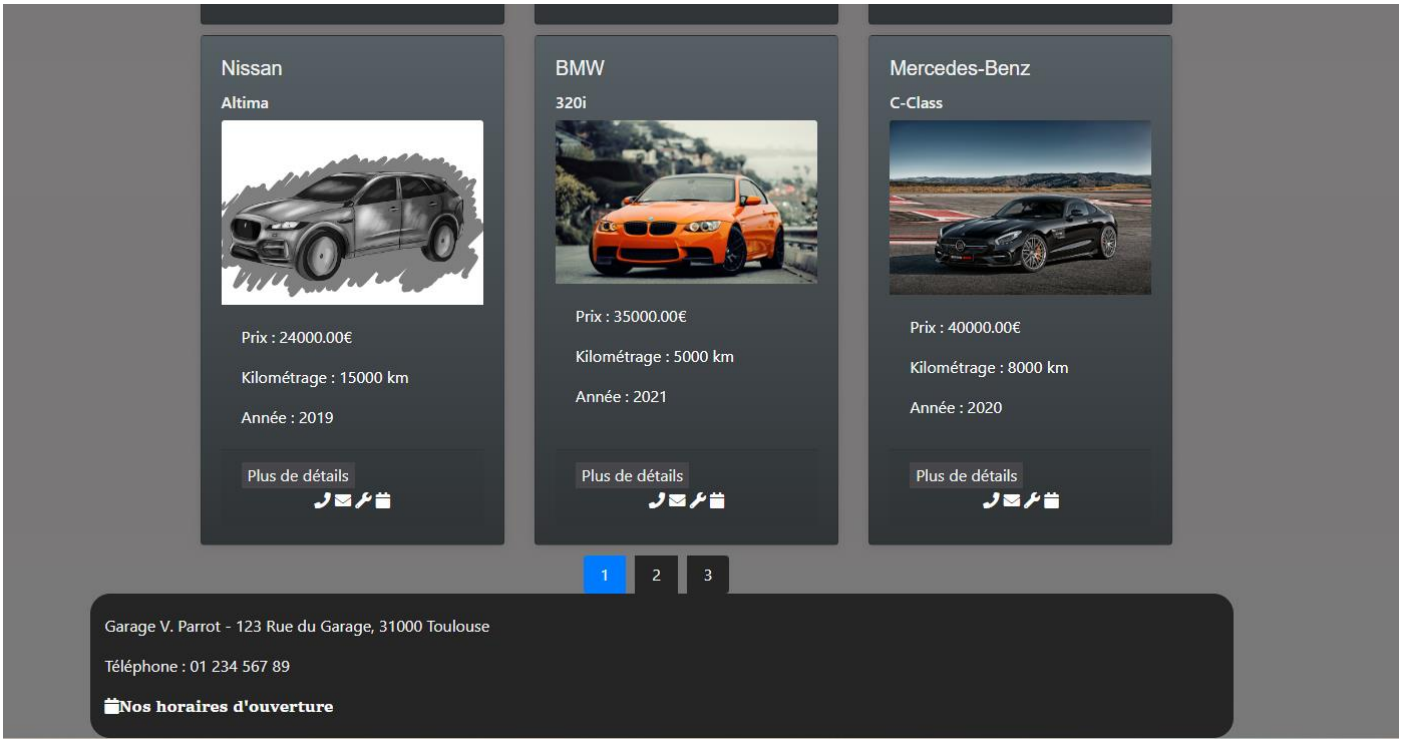
(Un peu plus bas nous avons les informations du garages)



(Nous avons ensuite le formulaire de contact ainsi que le filtre des véhicules)



(Voici les véhicules en ventes dans ce garage)



(Pour finir nous avons une pagination et un pied de page avec les informations du garage)