**BACK END**

\*\*Chapitre : Configuration de la Base de Données\*\*

Dans ce chapitre, je vais discuter de la configuration de la base de données pour notre projet et des défis que j'ai rencontrés tout au long du processus. L'objectif principal était de concevoir une base de données fonctionnelle et efficace pour stocker les données de notre application.

\*\*Importation des Tables depuis MySQL Workbench\*\*

Au départ, j'ai utilisé MySQL Workbench pour concevoir le modèle conceptuel de la base de données, y compris les relations entre les tables. Cependant, lors de l'exportation vers MySQL PHPMyAdmin, j'ai rencontré des problèmes liés aux cardinalités entre les tables. Malgré mes efforts pour configurer les relations correctement dans MySQL Workbench, les contraintes de clé étrangère n'ont pas été prises en compte lors de l'exportation.

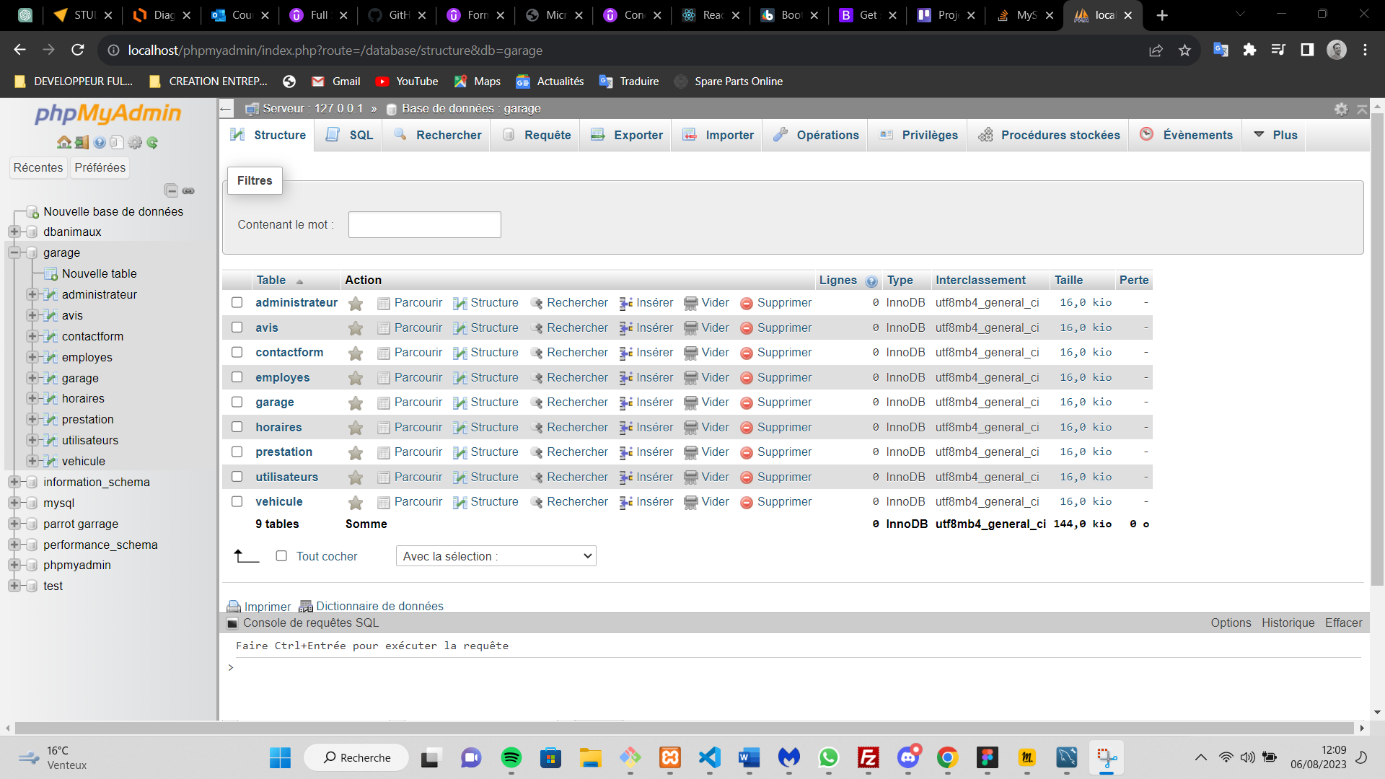
\*\*Ajout Manuel des Contraintes de Clé Étrangère\*\*

Pour résoudre ce problème, j'ai dû ajouter manuellement les contraintes de clé étrangère pour chaque table à l'aide de commandes SQL. Cela a permis d'établir les relations souhaitées entre les tables. Voici un exemple de commande que j'ai utilisée pour ajouter une contrainte de clé étrangère :

1. ALTER TABLE table\_enfant
2. ADD CONSTRAINT fk\_nom\_contrainte
3. FOREIGN KEY (colonne\_etrangere) REFERENCES table\_parente(colonne\_primaire);

```

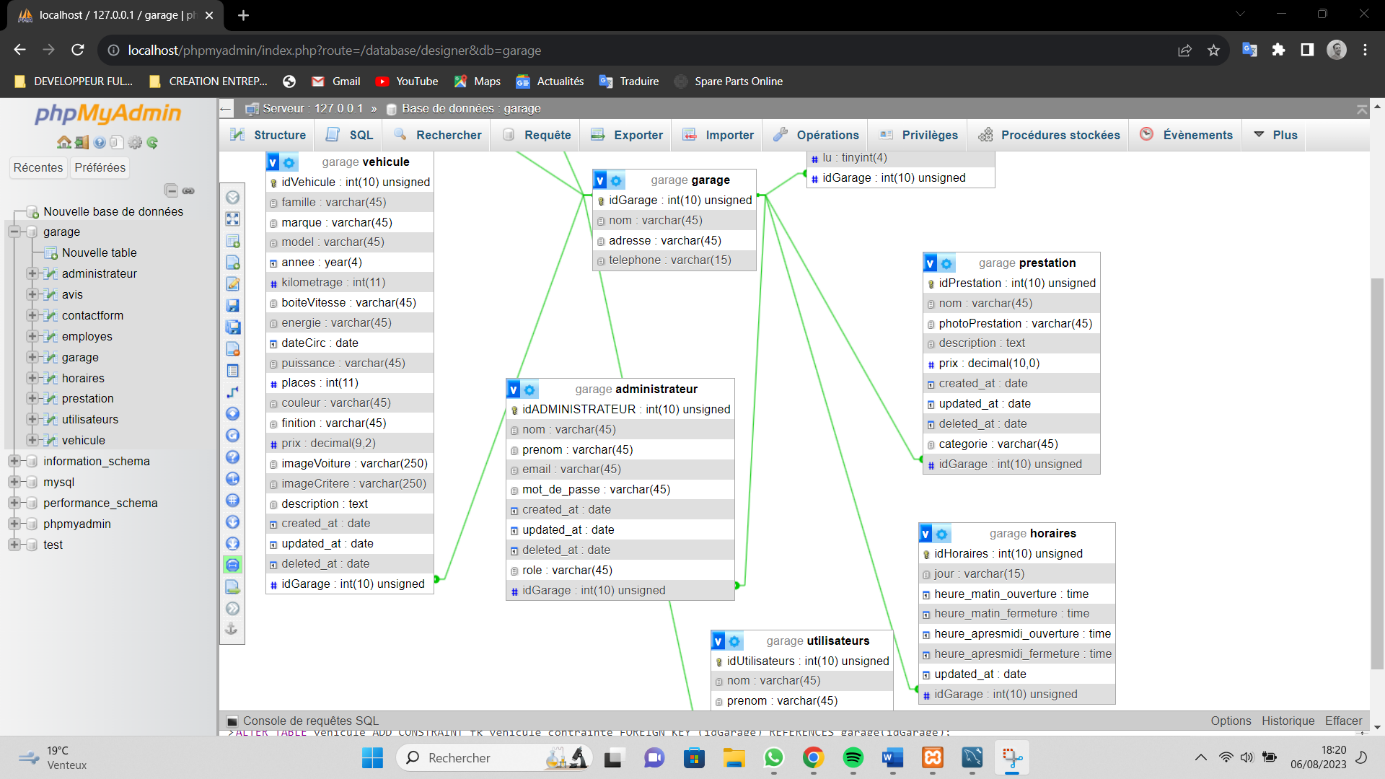
Cette solution a permis de garantir l'intégrité des données dans notre base de données.



Suite à la résolution initiale de mon problème, j'ai vérifié le modèle conceptuel dans PHPMyAdmin et j'ai constaté que les cardinalités n'avaient toujours pas été prises en compte. Face à cette situation, j'ai dû prendre une décision pour avancer dans le projet. J'ai choisi de passer par le panneau de commande SQL pour ajouter manuellement les contraintes de clé étrangère, table par table.

Bien que cette solution ait permis de créer les relations entre les tables conformément à nos besoins, je reste conscient que cela peut être considéré comme une approche moins automatisée et plus sujette aux erreurs. Cependant, dans le contexte de notre projet et compte tenu du temps limité, c'était la solution la plus pratique pour avancer.

Pour l'avenir, je vais continuer à travailler sur l'optimisation de notre base de données et des contraintes de clé étrangère, et je réévaluerai la possibilité de les réactiver une fois que je serai certain que toutes les données sont conformes aux contraintes. Cette approche garantira l'intégrité de nos données à long terme tout en nous permettant de poursuivre notre développement sans interruption.

En fin de compte, cette expérience m'a montré l'importance de la flexibilité et de l'adaptabilité dans le processus de développement, ainsi que la nécessité de prendre des décisions pragmatiques pour faire progresser un projet, même en cas de difficultés imprévues.

**\*\*Création de Données Temporaires dans PHPMyAdmin\*\***

**Après avoir résolu les problèmes liés aux contraintes de clé étrangère, j'ai entrepris de saisir de fausses données temporaires dans les tables de la base de données. L'objectif était de créer un environnement de test pour vérifier le fonctionnement de notre application.**

**Cependant, j'ai de nouveau rencontré des erreurs liées aux clés primaires lors de l'insertion de ces données. Après une analyse plus approfondie, j'ai identifié la source du problème. J'avais mal interprété les cardinalités des relations entre les tables. J'avais créé manuellement les clés étrangères alors que si j'avais choisi les bonnes cardinalités, les clés auraient été créées automatiquement dans chaque table. Une fois que j'ai ajusté les cardinalités correctement, j'ai pu exporter les données sans problème.**

**\*\*Ajout de Données dans la Table Horaire\*\***

**Lorsque j'ai commencé à ajouter des données dans la table "horaire", j'ai réalisé que j'avais initialement considéré uniquement les jours d'ouverture, négligeant les jours de fermeture. J'ai rapidement rectifié cette omission en ajoutant une colonne supplémentaire appelée "ferme" à la table. Cette colonne prendra la valeur 0 si l'établissement est ouvert et 1 si c'est fermé, me permettant ainsi de gérer correctement les jours de fermeture.**

**Mise en Place de l'Environnement de Travail**

**Pour optimiser mon environnement de développement, j'ai réalisé plusieurs étapes clés :**

**- \*\*Création du fichier index.php :\*\* Ce fichier joue un rôle central dans notre modèle MVC (Modèle-Vue-Contrôleur). Toutes les pages que je créerai ultérieurement seront redirigées vers index.php grâce à la méthode GET. Cela permettra une gestion centralisée de nos requêtes et de nos vues.**

**- \*\*Création du fichier .htaccess :\*\* J'ai également créé un fichier .htaccess pour configurer le serveur Apache de XAMPP. Ce fichier est essentiel pour mettre en place un système de réécriture des URLs, ce qui rendra les URLs de notre application plus conviviales et compréhensibles pour les utilisateurs.**

**Dans l'ensemble, ces étapes m'ont permis de mettre en place un environnement de développement robuste et de progresser dans la création de notre application. Je continue à travailler sur l'ajout de données factices dans nos tables pour tester et affiner le fonctionnement de l'application.-création d’une branche que je vais nommé routage avec Git.**

**### 2. Bloc "catch" : Gestion des Exceptions**

**Dans le cadre du développement de notre application, j'ai utilisé des blocs "try" et "catch" pour gérer les exceptions. Voici comment cela fonctionne :**

**Dans le bloc "try", je plaçais le code qui était susceptible de générer une exception, tel que des opérations de base de données ou des appels à des services externes. L'objectif était de capturer toute erreur ou exception potentiellement générée par ces opérations.**

1. **Bloc "try"** : Dans ce bloc, je vais placer le code qui pourrait potentiellement générer une exception. Si une exception est lancée pendant l'exécution de ce bloc, le code s'arrête immédiatement, et le contrôle passe au bloc **catch**.
2. **Bloc "catch"** : Dans ce bloc, je definirais comment gérer l'exception qui a été lancée dans le bloc "try"..
3. **Je souhaite qu’il y est 2 informations après le / pour pouvoir mieux organiser mon système de routage.**

**Le bloc "catch" est essentiel car c'est là que j'ai défini comment gérer l'exception qui a été lancée dans le bloc "try". Je pouvais personnaliser la gestion de chaque type d'exception en fonction de nos besoins spécifiques. Cela incluait l'affichage de messages d'erreur clairs pour les utilisateurs ou la journalisation des erreurs pour un débogage ultérieur.**

**### 3. Mise en Place du Système de Routage**

**Une autre étape cruciale de mon stage a été la mise en place d'un système de routage pour organiser les URL de notre application. Mon objectif était de distinguer clairement la partie "front" de la partie "back" de l'application, tout en permettant une gestion flexible des pages.**

**Je souhaitais que l'URL contienne deux informations après le "/", ce qui permettait une meilleure organisation du système de routage. Par exemple, notre URL ressemblait à ceci : `http://localhost/garageback/back/nom\_de\_la\_page`.**

**Pour mettre en place ce système de routage, j'ai suivi les étapes suivantes :**

**- \*\*Création du fichier index.php :\*\* J'ai créé un fichier index.php qui joue un rôle central dans notre modèle MVC (Modèle-Vue-Contrôleur). Toutes les pages que j'ai développées ultérieurement ont été redirigées vers index.php grâce à la méthode GET.**

**- \*\*Création du fichier .htaccess :\*\* Pour rendre notre application conviviale et améliorer l'expérience utilisateur, j'ai mis en place un fichier .htaccess. Ce fichier a permis de configurer le serveur Apache de XAMPP pour la réécriture des URLs, rendant ainsi les URLs plus compréhensibles et lisibles.**

**Ce système de routage a grandement contribué à l'efficacité de notre application en permettant une gestion plus claire des pages front-end et back-end, tout en offrant une expérience utilisateur améliorée.**

<?php

// 1. Création du fichier index.php dans lequel on définit une constante URL.

define("URL", str\_replace("index.php","",(isset($\_SERVER['HTTPS']) ? "https" : "http").

"://$\_SERVER[HTTP\_HOST]$\_SERVER[PHP\_SELF]"));

try{

    if(empty($\_GET['page'])){

        throw new Exception("La page n'existe pas");//Si l'URL est vide ou faussée, on lève une exception et on affiche une page d erreur.

    } else {

        $url = explode("/",filter\_var($\_GET['page'],FILTER\_SANITIZE\_URL));//On récupère l'URL et on la filtre pour pouvoir la mieux sécuriser.

        if(empty($url[0]) || empty($url[1])) throw new Exception ("La page n'existe pas");//On va vérifier que l'URL contient bien 2 paramètres en ajoutant une auytre exeption.Si à l'indice 0 ou 1 de l'url après le / n'existe pas aprés la page front ou back, ds ce cas la je vais lever une erreur.

        switch($url[0]){

            case "front" : //On vérifie la valeur de l'url 0, si elle est égale à front, on va vérifier la valeur de l'url 1.

                switch($url[1]){//On vérifie la valeur de l'url 1, si elle est égale à accueil, on affiche la page accueil.

                    case "accueil": echo "page accueil";

                    break;

                    case "prestations": echo " page prestations".$url[2]." demandées";//On affiche la page prestation et on ajoute à l'indice 2 la prestation demandée avec un Id.

                    break;

                    case "voituresFiltre": echo "page voituresFiltre";

                    break;

                    case "voitureFiche": echo "page voitureFiche".$url[2]." demandées";//On affiche la page voitureFiche et on ajoute à l'indice 2 la voiture selectionnée avec un Id.

                    break;

                    case "contact": echo "page contact";

                    break;

                    case "avis": echo "page avis";

                    break;

                    default : throw new Exception ("La page n'existe pas");

                }

            break;

            case "back" : echo "page back end demandée";

            break;

            default : throw new Exception ("La page n'existe pas");//On léve encore une exeption que s'il n'y a pas front ou back mais autre chose de marqué ds l'url, on affichera un message d'erreur.

        }

    }

} catch (Exception $e){

    $msg = $e->getMessage();

    echo $msg;

}

**### 4. Tests du Système de Routage**

**Après avoir mis en place le système de routage pour distinguer clairement les parties "front" et "back" de notre application, j'ai entrepris de tester ce système pour m'assurer de son bon fonctionnement. Les tests étaient essentiels pour garantir que les pages étaient correctement acheminées vers les contrôleurs appropriés.**

**Pour effectuer ces tests, j'ai utilisé la fonction `echo` pour afficher des informations sur l'URL et vérifier comment le système de routage réagissait. Cela m'a permis de diagnostiquer rapidement les éventuels problèmes de routage et de les corriger en conséquence.**

**# 5. Personnalisation des URLs pour les Pages "voituresFiche" et "prestations"**

**Pour les pages "voituresFiche" et "prestations", j'ai mis en place une personnalisation supplémentaire de l'URL pour une expérience utilisateur améliorée. Mon objectif était de permettre l'accès direct aux fiches de voitures et aux informations sur les prestations en utilisant l'ID unique de chaque objet.**

**e**

**Concrètement, j'ai ajouté l'ID unique à l'indice 2 de l'URL. Par exemple, l'URL ressemblait à ceci : `http://localhost/garageback/back/voituresFiche/123` ou `http://localhost/garageback/back/prestations/456`.**

**Cette personnalisation des URLs permettait aux utilisateurs d'accéder rapidement aux informations spécifiques qu'ils recherchaient, améliorant ainsi l'expérience de navigation et la convivialité de notre application.**

**### 6. Le Rôle du Routeur**

**Un élément central de mon travail lors de mon stage a été la mise en place d'un routeur. Le routeur est un composant essentiel dans une application web, car il permet de faire l'association entre les demandes de l'utilisateur et la logique du site. Cela signifie qu'il décide comment chaque demande HTTP doit être gérée et quel contrôleur doit être appelé pour traiter la demande.**

**Le routeur permet d'acheminer les utilisateurs vers les bonnes pages en fonction de l'URL demandée. Il joue un rôle crucial dans la navigation et l'expérience utilisateur.**

**### 7. La Structure MVC (Modèle-Vue-Contrôleur)**

**Lors de mon stage, j'ai adopté une architecture MVC (Modèle-Vue-Contrôleur) pour organiser notre code de manière efficace et modulaire. Cette structure a été fondamentale dans la séparation des responsabilités et la gestion de l'application. Voici comment elle fonctionne :**

**- \*\*Le Contrôleur :\*\* Le contrôleur est le cœur de l'application. Il gère la partie logique et pilote la demande du client. C'est lui qui décide quels modèles doivent être sollicités pour récupérer les données nécessaires, puis il transmet ces données à la vue. Le contrôleur coordonne ce qui doit être mis en place pour renvoyer le résultat attendu.**

**- \*\*Le Modèle :\*\* Le modèle est responsable de la récupération des données demandées par le contrôleur. Il interagit avec la base de données ou d'autres sources de données pour collecter les informations nécessaires.**

**- \*\*La Vue :\*\* La vue est chargée de l'affichage des pages demandées. Elle utilise les données transmises par le contrôleur pour créer des pages web dynamiques.**

**Cette architecture MVC a permis une organisation claire et modulaire du code, facilitant ainsi la maintenance de l'application et le travail collaboratif entre les membres de l'équipe.**

**### 8. Mise en Place du Contrôleur Front-End**

**Pour commencer à organiser notre système de routage, j'ai créé un contrôleur du côté front-end. Ce contrôleur a regroupé toutes les routes nécessaires pour gérer les demandes des utilisateurs. Par exemple, une URL comme `http://localhost/garageback/front/accueil` a été associée à une fonction spécifique dans le contrôleur.**

**En utilisant des fonctions et des tests `echo`, j'ai pu vérifier si le système de routage fonctionnait correctement. Cette étape a été cruciale pour s'assurer que les utilisateurs étaient redirigés vers les bonnes pages en fonction de leurs demandes.**

**Bien sûr, poursuivons avec ces informations pour votre rapport de stage :**

**---**

**### 9. Gestion des Données avec le Modèle et l'API**

**L'une des étapes cruciales dans le développement de l'application a été la gestion des données. Pour cela, j'ai mis en place une architecture modèle-vue-contrôleur (MVC) pour séparer clairement les responsabilités. Après avoir créé le contrôleur front-end, la prochaine étape a été de mettre en place le modèle pour gérer les données.**

**J'ai créé un fichier manager dans la partie front du modèle. Ce manager a été chargé de gérer les données en utilisant une API (Interface de Programmation Applicative). L'API a été un élément clé pour interagir avec la base de données et récupérer les informations nécessaires pour l'application.**

**### 10. Utilisation de PDO pour une Liaison Sécurisée avec la Base de Données**

**Pour établir une connexion sécurisée entre notre application et la base de données, j'ai choisi d'utiliser PDO (PHP Data Objects). PDO est une interface de programmation permettant d'interagir avec différentes bases de données de manière sécurisée et portable.**

**L'utilisation de PDO a permis de prévenir les failles de sécurité potentielles telles que les injections SQL. En utilisant des requêtes préparées, j'ai pu garantir que les données entrantes étaient correctement traitées et échappées, réduisant ainsi les risques liés à la sécurité de l'application.**

**La liaison sécurisée entre l'application et la base de données a été un élément essentiel pour garantir l'intégrité des données et la confidentialité des informations stockées.**

**---**

// Création du controlleur du côté front qui va regrouper toutes nos routes

class APIController{

    public function getAccueil(){

        echo "accueil";

    }

    public function getPrestations($idPrestations){//On récupére en paramétre 2 de l'url l'Id

        echo " page prestations".$idPrestations." demandées";

    }

    public function getVoituresfiltre(){

        echo "voiture filtre";

    }

    public function getVoiturefiche($idVoiturefiche){//On récupére en paramétre 2 de l'url l'Id

        echo " page voiturefiltre".$idVoiturefiche." demandées";

    }

    public function getContact(){

        echo "contact";

    }

    public function getAvis(){

        echo "avis";

    }

}

abstract class Model{

    //On va créer une instance à PDO, il y aura qu'une seul instance pour toute la BDD

    private static $pdo;

    private static function setBdd(){//On va se connecter à la BDD

        self::$pdo = new PDO("mysql:host=localhost;dbname=garage;charset=utf8","root","");

        self::$pdo->setAttribute(PDO::ATTR\_ERRMODE,PDO::ERRMODE\_WARNING);//On va afficher les erreurs SQL

    }

}

**### 11. Tests avec de Faux Données et la Page "voituresFiche"**

**Pour effectuer ces tests, j'ai utilisé des données factices que j'avais préalablement créées dans la base de données. Cette approche m'a permis de simuler un environnement réel et de vérifier si tout fonctionnait comme prévu.**

**Lors de ces tests, j'ai spécifié un ID dans l'URL, ce qui a permis d'identifier la voiture spécifique que je souhaitais afficher. Les données de cette voiture ont été récupérées à partir de la base de données et renvoyées sous forme de tableau.**

**À l'avenir, nous prévoyons de formater ces données au format JSON pour une meilleure manipulation et un affichage dynamique sur la page. Cette étape a été cruciale pour garantir que notre application fonctionne correctement et renvoie les informations attendues aux utilisateurs.**

**---**

Array

(

[0] => Array

(

[idVehicule] => 1

[famille] => Berline

[marque] => Peugeot

[model] => 208

[annee] => 2022

[kilometrage] => 23000

[boiteVitesse] => Automatique

[energie] => Essence

[dateCirc] => 2023-08-21

[puissance] => 5

[places] => 5

[couleur] => Rouge

[reference] => 2023-lk-12

[prix] => 12000.00

[imageVoiture] => peugeot.webp

[imageCritere] => critereA.webp

[description] => 1 ligne insérée.

Identifiant de la ligne insérée : 1

Warning: #1265 Data truncated for column 'prix' at row 1

Warning: #1366 Incorrect integer value: '' for column `garage`.`vehicule`.`garage\_idGarage` at row 1

[created\_at] =>

[updated\_at] =>

[deleted\_at] =>

[garage\_idGarage] => 1

)

**### 12. Liaison des Tables "Véhicule" et "Prestation" à la Table "Garage"**

**Au cours de mon stage, j'ai entrepris de relier les tables "Véhicule" et "Prestation" à la table "Garage" en utilisant des clés étrangères. Initialement, j'ai rencontré un problème où il ne semblait pas être capable de localiser la table "Garage" avec son ID. Après une analyse plus approfondie, j'ai découvert que le problème résidait dans les cardinalités que j'avais choisies. Une fois que j'ai ajusté les cardinalités appropriées, le problème de la jointure a été résolu. Cette étape a été essentielle pour créer des relations significatives entre les données dans la base de données.**

**### 13. Mise en Place des Filtres**

**J’ai développées la mise en place de filtres pour trier et afficher les données de manière sélective. J'ai suivi une approche basée sur le modèle MVC en utilisant du code brut.**

**Le modèle MVC impliquait la connexion à la base de données dans le modèle, la création d'un contrôleur qui renvoyait les données au format JSON, et la mise en place de l'API pour gérer les filtres. J'ai utilisé Postman pour effectuer des tests d'URL et résoudre les problèmes qui se sont présentés. J'ai pu ainsi développer une fonctionnalité de filtrage robuste pour notre application.**

**### 14. Gestion des Problèmes Techniques**

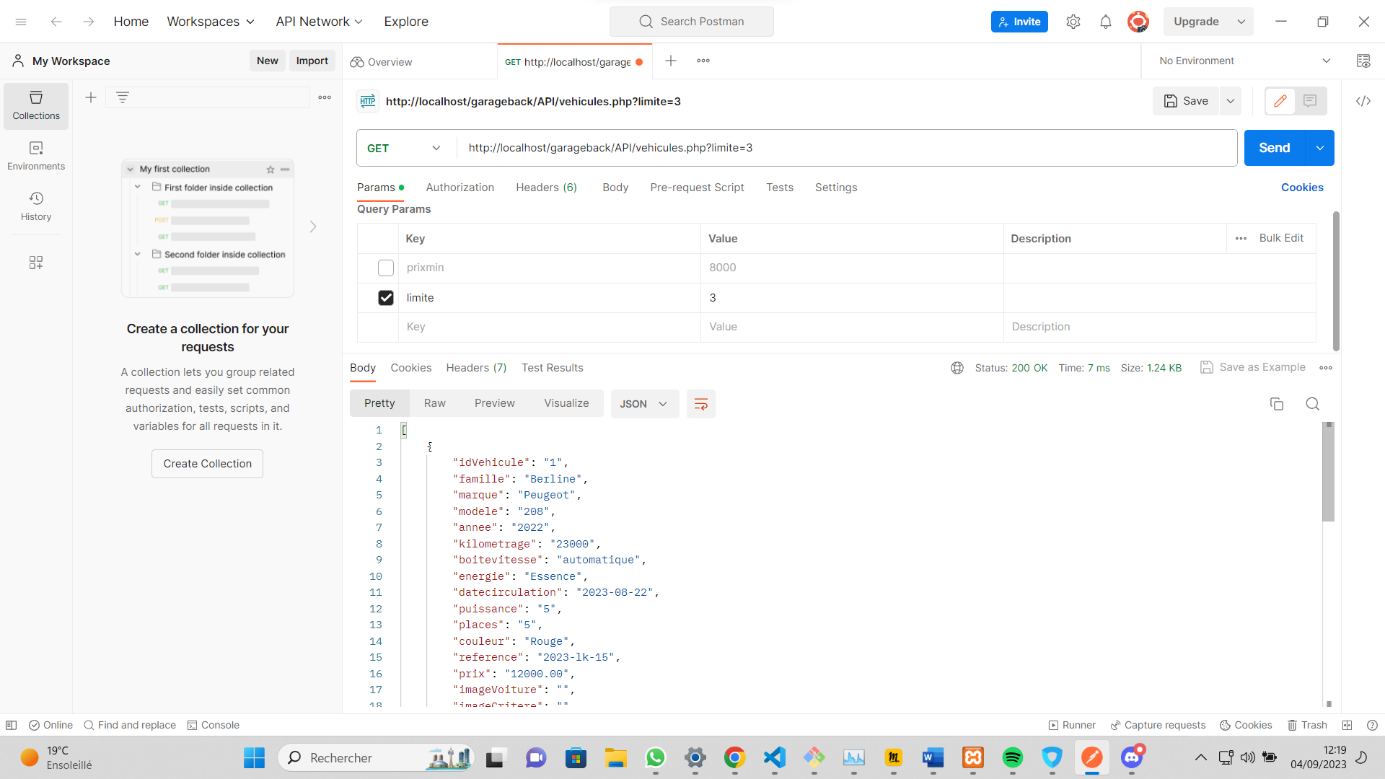
**Pendant mon stage, j'ai été confronté à plusieurs problèmes techniques. L'un d'entre eux concernait MariaDB de XAMPP, où je ne pouvais plus me connecter et où de nombreuses erreurs s'affichaient dans PHPMyAdmin. Après avoir cherché une solution, j'ai remarqué que MySQL ne fonctionnait pas correctement dans le gestionnaire de tâches. J'ai donc pris la décision de réinstaller XAMPP en sauvegardant mes fichiers, ce qui a permis de résoudre ce problème et de rétablir la connexion à la base de données.**

**### 15. Développement du Contrôleur Front-End**

**Avec la gestion des données et des filtres en place, j'ai pu me concentrer sur le développement du contrôleur front-end de l'application. Ce contrôleur a été chargé de manipuler les données des véhicules dans l'application. J'ai créé un fichier séparé du modèle (VehiculeModel) pour accéder aux données et gérer les opérations liées aux véhicules.**

**En utilisant des tests d'URL et en traitant les requêtes HTTP entrantes, j'ai pu mettre en place différentes actions en fonction de la méthode de la requête et des paramètres fournis dans l'URI. J'ai également pris en compte la gestion des erreurs (par exemple, une réponse 404 en cas de problème). Cette phase du projet a permis de préparer le terrain pour le développement du front-end de l'application.**

**---**

**[Continuez à développer votre rapport de stage en incluant d'autres détails sur vos réalisations, les compétences que vous avez acquises, et les défis que vous avez relevés. N'oubliez pas de personnaliser davantage le texte en fonction de votre expérience spécifique.]**

**### 16. Mise en Place du Système de Connexion pour l'Espace Professionnel**

**Une étape cruciale de mon travail a été la mise en place du système de connexion pour l'espace professionnel de l'application. Voici les étapes que j'ai suivies pour créer ce système robuste :**

**- \*\*Création des Champs de Connexion :\*\* Pour gagner du temps, j'ai utilisé des champs Bootstrap préexistants pour créer les champs de connexion. Cela a permis de concevoir rapidement une interface utilisateur conviviale.**

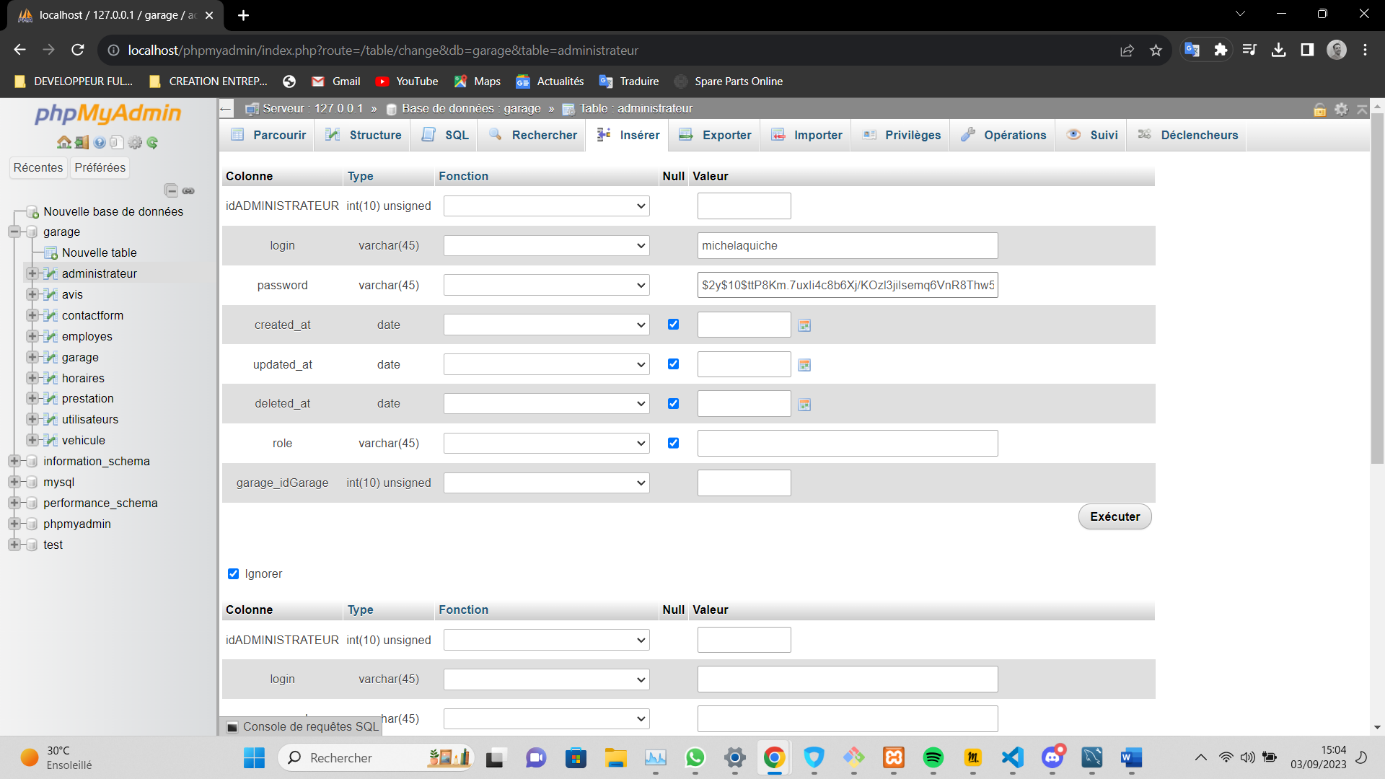
**- \*\*Utilisation de password\_hash() pour la Sécurité des Mots de Passe :\*\* La sécurité des mots de passe est primordiale. J'ai utilisé la fonction `password\_hash()` pour hacher (crypter) les mots de passe des utilisateurs. Cette fonction génère un hachage sécurisé qui peut être stocké en toute sécurité dans la base de données. J'ai opté pour l'option `PASSWORD\_DEFAULT`, considérée comme la plus sécurisée à ce jour.**

**- \*\*Modification des Tables "Employés" et "Admin" :\*\* J'ai rationalisé les tables "Employés" et "Admin" en supprimant les informations inutiles, notamment l'e-mail. Pour l'instant, j'ai conservé uniquement les champs "Login" et "Mot de Passe". Le login sera généré automatiquement, tandis que le mot de passe sera sécurisé avec `password\_hash()`.**

**- \*\*Tests du Système de Connexion :\*\* J'ai effectué des tests en appuyant sur le bouton "Valider". Le système a généré automatiquement un mot de passe sécurisé, que j'ai pu vérifier en le comparant avec le mot de passe haché stocké dans la base de données. À chaque régénération de la page, un nouveau mot de passe était généré.**

**- \*\*Insertion de Données Factices :\*\* Pour effectuer des tests plus poussés, j'ai inséré de fausses données dans la table, en copiant le mot de passe généré automatiquement et en l'insérant dans le champ "password" de ma table "Employés" et "Admin".**

**La mise en place de ce système de connexion constitue une avancée majeure pour l'application, en garantissant la sécurité des données sensibles des utilisateurs professionnels.**



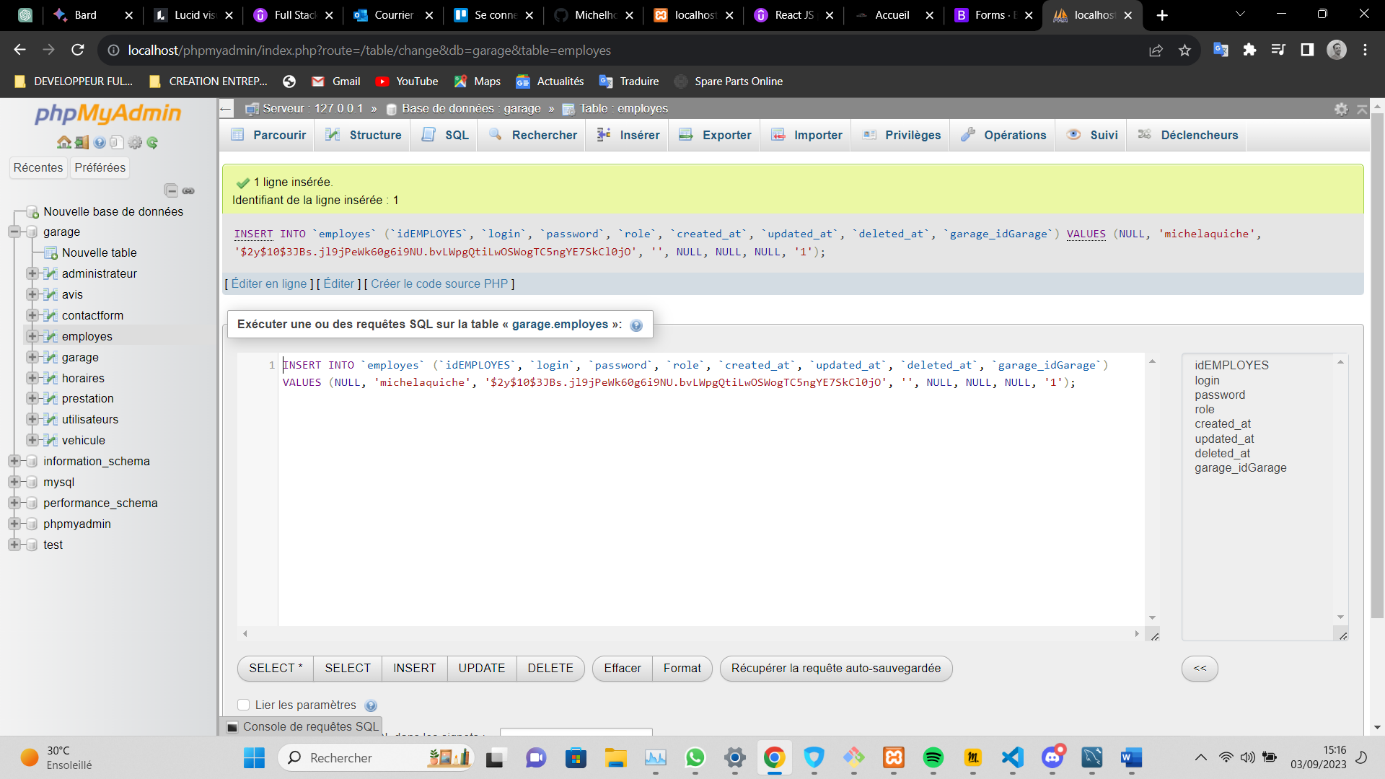
### 17. Validation des Tests et Conclusions

En conclusion de cette phase de développement, j'ai réussi à mettre en place un système de gestion de mot de passe sécurisé pour l'espace professionnel de l'application. Le processus de validation a été concluant, confirmant l'efficacité du hachage de mot de passe.

Pour valider le système, j'ai utilisé la fonction `password\_hash()` pour hacher un mot de passe spécifique, en l'occurrence "adminmichelaquiche". Le test a été un succès, puisque le mot de passe généré automatiquement correspondait au hachage attendu.

Cette réalisation constitue une étape importante dans le développement de l'application, garantissant la sécurité des mots de passe des utilisateurs professionnels et leur confidentialité.

---

****

**Parfait, poursuivons en intégrant ces informations dans votre rapport de stage pour couvrir cette partie du développement :**

**---**

**### 18. Sécurisation et Vérification des Informations de Connexion**

**La sécurité de l'application est une priorité majeure, notamment lorsqu'il s'agit de gérer les informations de connexion. Voici les étapes que j'ai suivies pour renforcer la sécurité de la gestion des comptes professionnels :**

**- \*\*Ajustement de la Longueur des Mots de Passe :\*\* Lors de l'insertion des données dans la table, j'ai rencontré une erreur liée à la longueur du mot de passe haché. Pour remédier à cela, j'ai ajusté la longueur du champ VARCHAR, en veillant à ce qu'il soit suffisamment long pour accueillir le hachage. J'ai également constaté que l'utilisation de caractères spéciaux dans le mot de passe pouvait entraîner des problèmes, j'ai donc préféré n'utiliser que des caractères alphanumériques.**

**- \*\*Mise en Place de Vérifications :\*\* J'ai créé une page dédiée à la sécurité, où j'ai converti en HTML les caractères spéciaux pour éviter certains problèmes de sécurité. J'ai également mis en place un système de vérification pour s'assurer que les champs de connexion sont correctement remplis.**

**- \*\*Gestion des Sessions :\*\* Si les informations de connexion sont correctes (vérifiées en utilisant des identifiants préalablement créés), une session est générée, et l'utilisateur est redirigé vers la page administrative. Cependant, j'ai constaté un problème où la connexion échouait et rafraîchissait la page à chaque tentative.**

**- \*\*Sécurisation de l'URL de la Page Admin :\*\* Pour renforcer la sécurité, j'ai sécurisé l'accès à la page d'administration en créant un fichier dédié à la sécurité. J'ai également prévu de créer un lien de déconnexion qui supprimera la variable de session lorsque l'utilisateur se déconnecte.**

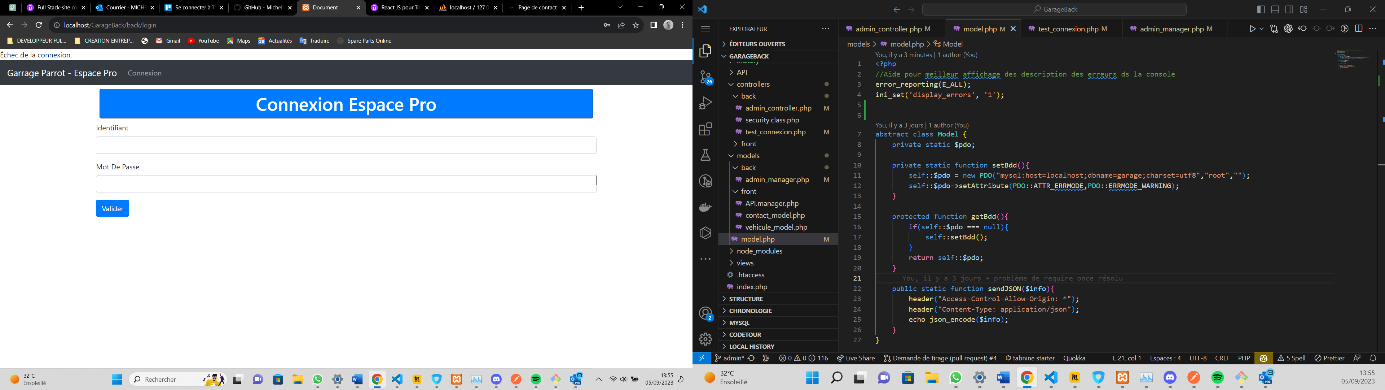
**- \*\*Validation des Informations de Connexion :\*\* J'ai développé une fonction de vérification des informations de connexion pour m'assurer que l'utilisateur a rempli correctement les champs requis. Cela garantit également que lors de la déconnexion, l'accès à la page d'administration est désactivé.**

**- \*\*Mise en Place de la Navbar Admin :\*\* J'ai préparé une navbar pour l'administration, présentant toutes les fonctionnalités autorisées sous forme de menus déroulants. La navbar est visible uniquement lorsque l'administrateur est connecté. Pour ce faire, j'ai intégré du code PHP dans le code HTML existant.**

**- \*\*Résolution des Problèmes de Connexion :\*\* J'ai rencontré des problèmes de connexion, mais en examinant attentivement le code, j'ai identifié une erreur de syntaxe avec les instructions `require\_once`. Après correction, la connexion a fonctionné correctement.**

**Ces étapes ont été essentielles pour garantir que les informations de connexion sont sécurisées et que seuls les utilisateurs autorisés ont accès à la page d'administration.**

**---**

****

**### 19. Tests des données factices dans la table administrateur et Affichage des Données**

**Je me suis rendu compte que ce n’était pas la peine de créer 2 tables différentes admin + employé du côté de l’espace pro.**

**Je vais plutôt gérer tout cela sur une seule table à qui j’attribuerais des rôles différents par la suite.**

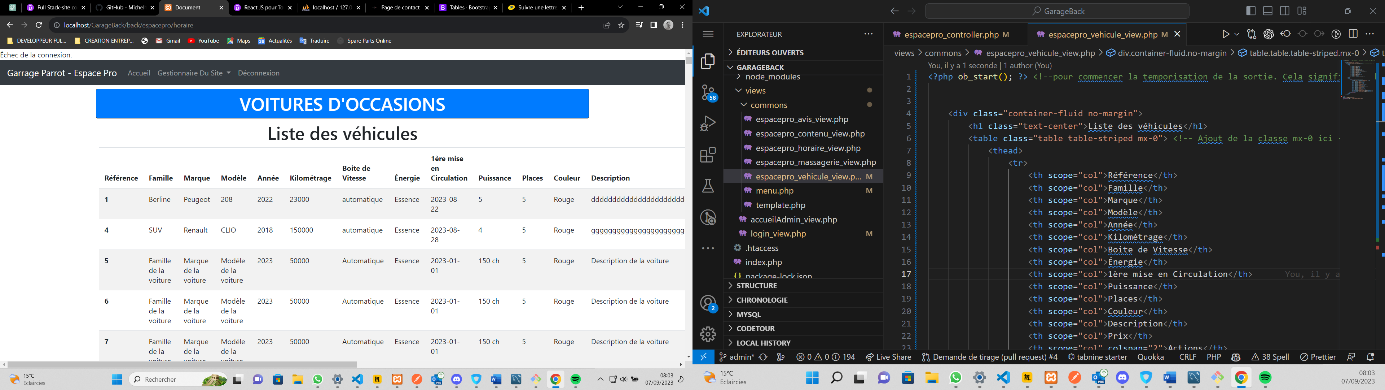
**Voici comment j'ai procédé pour afficher la table "Véhicule" dans l'espace professionnel :**

**- \*\*Correction des Erreurs de Connexion :\*\* Tout d'abord, j'ai résolu les problèmes de connexion qui empêchaient l'accès à la page. Une fois la connexion réussie, j'ai pu afficher les données sous forme de tableau.**

**- \*\*Tests de Données :\*\* Pour vérifier que tout fonctionnait correctement, j'ai ajouté des données factices à la table et j'ai effectué des tests en utilisant le contrôleur correspondant. Cela m'a permis de m'assurer que les données étaient correctement récupérées depuis la base de données.**

**- \*\*Mise en Forme des Données :\*\* Pour améliorer la présentation des données, j'ai utilisé une table Bootstrap prête à l'emploi. Cela m'a permis de gagner du temps tout en offrant une interface utilisateur plus conviviale. J'ai également préparé des boutons pour les opérations de base du CRUD (Create, Read, Update, Delete).**

**L'affichage des données dans l'espace professionnel est une étape essentielle pour fournir aux utilisateurs un accès rapide et efficace aux informations dont ils ont besoin. Les prochaines étapes consisteront à mettre en place les fonctionnalités de modification et de suppression.**

****

**### 20. Implémentation du Bouton de Suppression**

**Implémentation du bouton de suppression est une étape cruciale pour garantir la sécurité des données et éviter toute suppression accidentelle. Voici comment j'ai mis en place ce mécanisme dans l'espace professionnel :**

**- \*\*Création d'une Nouvelle Route :\*\* Tout d'abord, j'ai créé une nouvelle route pour gérer l'action de suppression. Cette route permettra de transmettre l'identifiant du véhicule à supprimer depuis l'URL.**

**- \*\*Conversion de l'ID en Entier :\*\* Étant donné que l'URL transmet l'identifiant en tant que chaîne de caractères, j'ai dû convertir cette chaîne en entier dans le contrôleur correspondant. Cela a permis d'éviter les erreurs et les problèmes de sécurité potentiels.**

**- \*\*Mise en Place d'Alertes JavaScript :\*\* J'ai préparé des alertes en JavaScript pour confirmer l'action de suppression. Ces alertes sont conçues pour n'apparaître qu'une seule fois par page, et j'ai veillé à supprimer la variable de session associée une fois que le message a été affiché.**

**- \*\*Gestion des Redirections :\*\* Cependant, j'ai rencontré un problème de redirection après la suppression. Bien que le chemin de redirection soit correct, j'ai dû le commenter temporairement pour résoudre le problème.**

**La mise en place de ce mécanisme de suppression sécurisé est essentielle pour protéger les données de l'application et garantir que les opérations de suppression sont effectuées de manière intentionnelle.**

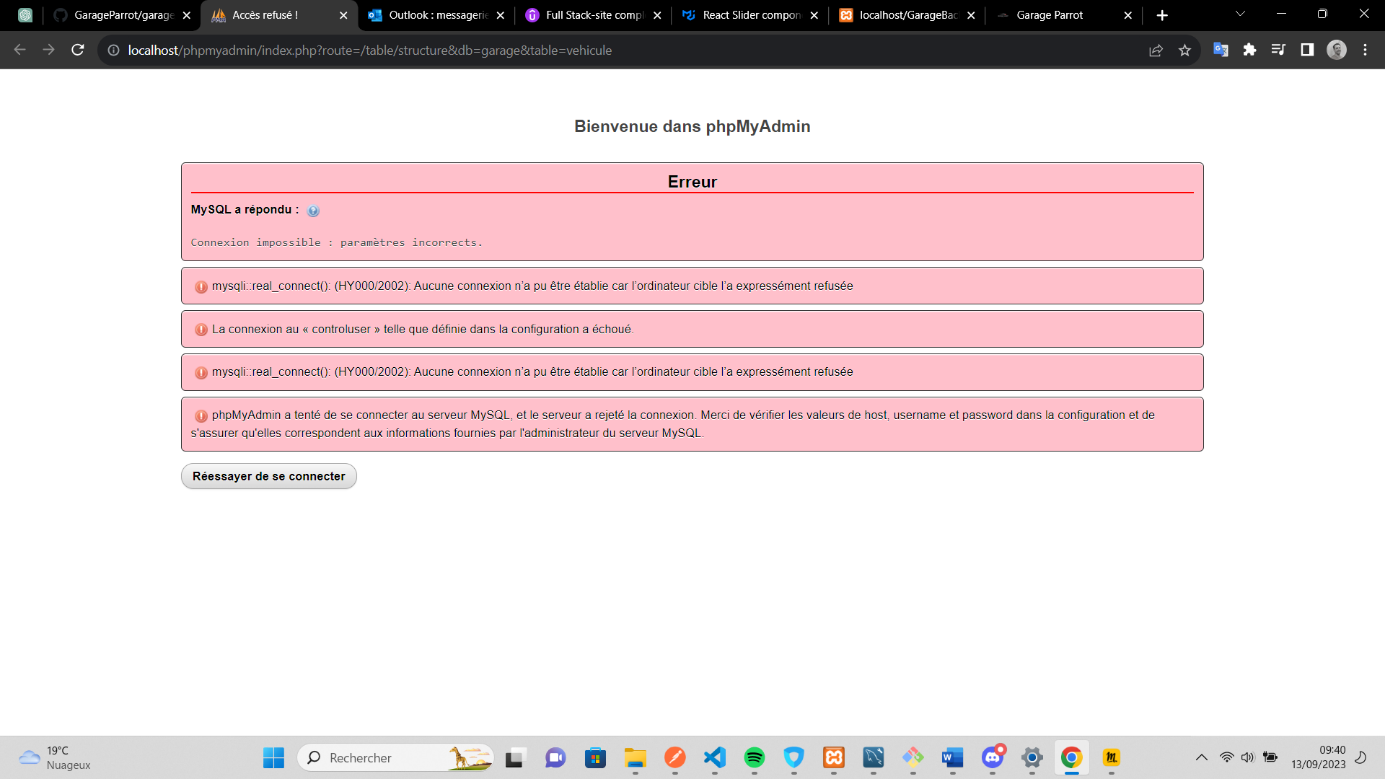
**BOUTON MODIFIER**

**Je vais créer ds le view des inputs pour chaque champs de la table véhicule pour pouvoir modifier l’id d’un véhicule, il y aura donc possibilité de tout changer sauf l’id bien sûr. Je rajouterai un bouton valider par la suite.**

**Après cela, je vais m’occuper du côté serveur lorsque l’utilisateur va valider.**

**Diffuculté rencontré/ :**

**J’ai pour la 2 éme foisplus accée au serveur, sql me refuse carrement l accée et je ne sais pas pourquoi, après les tests que j’ai du faire. J’ai dû réinstallé xampp mais le soucis c’est que j ai pu récupérer ma table sans mes données alors que j ai sauvegardé le dossier sql auparavant. Il doit y avoir une solutions pour garder ces donées donc je vais faire des recherches.**

****

**Je suis complétement dans l’impasse, après avoir réinstallé Xampp, je me retrouve à avoir différents problèmes Je souhaite donc regénérer un autre mot de passe pour pouvoir les insérer dans la Bdd mais impossible de le refaire et m’affiche des messages d’ erreur par rapport à un ancien code du controller qui n’existe même plus . Je vide donc mon cache, redermarre xampp, j isole tout les reste de mon code et rien n’a bougé.**

**En me renseignant, je peux aussi aller voir d autres messages d’erreurs dans un fichier log de Xamps mais en prenant la précaution de désactiver apache pour éviter de se tromper de pouvoir ainsi ciblé le message d erreur voulu ;mais apparement je n’ai que warning qui concerne le certificat ssl qui doit être en https mais comme je travaille en local, ça ne me concerne pas directement**

**\*\*Conclusion :\*\* Dans la conception de l'interface en HTML, j'ai élaboré un tableau destiné à afficher la liste complète des véhicules. À cette fin, j'ai implémenté une boucle "foreach" pour itérer à travers les données du tableau, permettant ainsi l'affichage détaillé de chaque véhicule.**

**De plus, j'ai associé les fonctionnalités de modification et de suppression à des formulaires HTML dédiés. Ces formulaires fournissent une interface utilisateur intuitive pour gérer les opérations de mise à jour et de suppression des véhicules de manière efficace.**

**Pour renforcer la sécurité et éviter toute suppression accidentelle, j'ai intégré une demande de confirmation en JavaScript pour le bouton "Supprimer". Cette mesure préventive assure une expérience utilisateur plus fiable.**

**En outre, j'ai mis en place une condition pour vérifier si un formulaire de modification a été soumis pour un véhicule spécifique. En cas de soumission, un formulaire de modification est généré pour le véhicule concerné, avec des champs pré-remplis (inputs) facilitant ainsi la modification des détails.**

**Cette approche globale garantit une gestion fluide et sécurisée des véhicules au sein de l'application, tout en offrant une expérience utilisateur optimale.**

**admin**

**$2y$10$T2YHXng7Dvx9IYMxG7ZwKePVfMDI3Po0WJE5pzB6/gF**

**GIT**

**Creation des dossiers ds github et copie collé ds git pour pouvoir communiquer entre git et github**

**Git permet de gérer les branches totalement en local et a distant**

**Creation des branch avec git branch=> git checkout + nom de la branche pour aller sur la branche crée**

**Git pull pour récupérer une branche sur le serveur distant**

La commande **git branch** est utilisée pour afficher la liste des branches dans un référentiel Git. Elle vous montre quelles sont les branches existantes, quelle est la branche actuelle et met en évidence la branche sur laquelle vous vous trouvez. Voici comment utiliser **git branch**

**Mettez à jour vos branches locales :** Avant de commencer la fusion, assurez-vous que vos branches locales sont à jour en exécutant les commandes suivantes :

bashCopy code

git checkout <votre\_branche\_de\_destination> git pull origin <votre\_branche\_de\_destination>

git checkout <votre\_branche\_de\_destination>

FUSION BRANCHE

Pour fusionner une branche avec la branche principale (généralement nommée `main` ou `master`), vous pouvez suivre ces étapes. Assurons-nous d'abord que vous êtes sur la branche que vous souhaitez fusionner dans votre dépôt local :

1. \*\*Se positionner sur la branche à fusionner :\*\*

git checkout nom-de-la-branche

2. \*\*Mettre à jour la branche principale (main) :\*\*

Avant de fusionner, assurez-vous d'avoir les dernières modifications de la branche principale (main) en récupérant les dernières modifications du référentiel distant.

```bash

git pull origin main

3. \*\*Fusionner la branche : sur main : aller sur main et git merge + nom de la branche

Une fois que vous êtes sur la branche à fusionner et que la branche principale est à jour, vous pouvez utiliser la commande `git merge` pour fusionner la branche avec la branche principale.

Cela prendra les modifications de la branche principale et les fusionnera dans votre branche actuelle.

**Revenir en arrière après une fusion incorrecte** :

Si une fusion a été faite incorrectement et que vous souhaitez revenir à l'état

git reset --hard HEAD^ git reset --hard HEAD^

4. \*\*Résoudre les conflits éventuels :\*\*

S'il y a des conflits entre les modifications dans votre branche et celles de la branche principale, résolvez-les en modifiant les fichiers en conflit, en les marquant comme résolus et en ajoutant les modifications.

git add . # Ajouter les fichiers modifiés après la résolution des conflits

git commit -m "Résolution des conflits" # Valider les modifications de résolution

5. \*\*Pousser les modifications fusionnées :\*\*

Une fois que les conflits sont résolus et que les modifications sont validées, vous pouvez pousser les modifications fusionnées vers le référentiel distant.

git push origin nom-de-la-branche # Pousser les modifications de la branche fusionnée

Notez que cela poussera les modifications vers la branche que vous avez fusionnée, pas la branche principale.

**Git log pour voir toutes les opérations effectuées uniquement sur la branche que l’on est.**

**Git branch -a pour voir les branches crée à distance**

**Supprimer une branche en LOCAL après le git push quand on a plus l utilité : aller sur la branch main(git checkout main) => git branch -d + nom de la branche (supprimer en local).**

**Supprimer une branche à distance : git push origin --delete + nom de la branche que l on veut supprimer on peut verifier en faisant git branch -a pour voir si la branche est bien supprimé**

**git add**. Cela signifie que vous signalez à Git que vous souhaitez inclure ces modifications dans votre prochain commit.

Git commit -m “explication breve du travail effectué“

Git push pour envoyer le travail sur github

**FRONT END**

Mise en place de l’environnement de travail, REACT,des dossier, difficulté d’importer les composants : problème, il ne reconnait pas les majuscule des dossiers, je suis obligé de les mettre à la main, j ai eu le meme probleme sur un autre projet.

Mise en place de bootstrap VIA les CDN

**COMPOSANTS SF ET SL**

Création de snippnets sur vscode pour composant SL et SF de manière automatique pour faciliter le travail => fichier => preference => extrait utilisateur =>javascript.json =>

"import React & creer un composant stateless": {

        "prefix": "imp-r-sl",// (import r stateless) A definir soit même, on peut mettre le nom que l on veut afin qu'il soit bien repérable pour pouvoir l utiliser au plus vite

        "body": [

            "import React from \"react\"; \n",//\n = sauter une ligne

            "const ${component} = (props) => (",//${component} = permet de nommer plus facilement

            "\t//Ecrire le code ici", //\t = tabulation = permet au code d'être bien rangé

    ");\n",

        "export default ${component};"

        ],

        "description": "Import React & creer un composant stateless"

    },

    "import React & creer un composant statefull": {

        "prefix": "imp-r-sf",// (import r statefull) A definir soit même, on peut mettre le nom que l on veut afin qu'il soit bien repérable pour pouvoir l utiliser au plus vite

        "body": [

            "import React, { Component } from 'react';\n",

                "class  ${component} extends Component{",

                "\trender() {",

                    "\t\treturn (",

                    "\t\t\t//Ecrire le code ici",

                "\t\t);",

                "\t}",

            "}\n",

                "export default ${component};",

        ],

        "description": "Import React & creer un composant statefull"

    }

}

//On va pouvoir maintenant sur n importe quel fichier JS juste en tapant import pour pouvoir importer le code ci dessus et ainsi gagner du temps

// Un composant Stateless, également connu sous le nom de composant sans état ou composant fonctionnel, est un concept dans la

// programmation des interfaces utilisateur (UI) qui se réfère à un type de composant qui ne maintient pas d'état interne.

// Un composant Stateless est principalement défini par ses entrées (props) et génère une sortie (UI) basée uniquement sur ces entrées.

// Voici quelques caractéristiques d'un composant Stateless :

// 1. Absence d'état interne : Un composant Stateless ne conserve pas de données ou d'état interne. Il ne gère pas ses propres données,

//  mais utilise plutôt les données fournies par ses propres props (propriétés) pour générer l'interface utilisateur.

// 2. Basé sur les props : Un composant Stateless reçoit des données sous forme de props. Ces props sont passées au composant en tant

// que paramètres lorsqu'il est utilisé dans un autre composant. Le composant utilise ces props pour générer dynamiquement l'UI en

//  fonction des données reçues.

// 3. Logique minimale : Étant donné que les composants Stateless ne gèrent pas leur propre état, ils contiennent généralement une

// logique minimale. Leur principale responsabilité est de transformer les données fournies par les props en éléments d'interface

// utilisateur.

// 4. Réutilisabilité : Les composants Stateless sont souvent réutilisables car ils ne dépendent pas de leur état interne spécifique.

// Ils peuvent être utilisés dans différents contextes en leur fournissant simplement les données appropriées via les props.

// L'utilisation de composants Stateless peut contribuer à la simplicité et à la modularité du code, car ils se concentrent

// uniquement sur la transformation des données d'entrée en sortie UI, sans se soucier de la gestion de l'état interne.

// Ils sont couramment utilisés dans les frameworks et bibliothèques de développement d'interfaces utilisateur tels que React, Vue.js,

//  Angular, etc.

CREATION DU COMPOSANT NAVBAR AVEC BOOTSTRAP.

Créatio d’un composant stateless nomé navbar.js. J’ai importé toutes le images et inséré le code ds le JSX et en nommant les classes className pour le style.

J’ai crrée un fichier site.js où je mettrais par la suite mon composant navbar.

J’ai présenté tous les services de réparation sous forme de dropdown et je préfére en les entourant des Navlinks pour éviter à chaque fois le rechargement de la page,je vais d’ailleur le faire pour chacune des routes.

CREATION DE LA PAGE D ERREUR 404

Je crée une route type error 404

AVIS CLIENT

Je vais utiliser axios pour les requêtes : npm install axios

COMPOSANT CARD VOITURES

Je vais crée un composant sous forme de carte et je vais les directement connecter à la bdd avec l aide de axios .

Résultat max de 12 vehicules et mis en place d’une pagination

SYSTEME DE RECHERCHE VEHICULE PAR FILTRE

ajout filtre indicateurs en temps réel au-dessus des curseurs pour les filtres d'année et de kilométrage ET PRIX et un dropdown pour selection marque

REACT ROUTEUR

J EVAIS maintenant réaliser le routage des routes avec react router : avant tout aller ds le terminal pour l’installer : npm install react-router-dom.

Quand le composant est installé je l import sur apps.js et je vais indiquer au composant site que je vais englober toutes mes routes.

ACCUEIL

-Mise en place du composant h1

npm install --save font-awesome et ensuite

-Mise en place du composant caroussel pour les prestation du garage sous forme de card avec bouton

FORMULAIRE DE CONTACT

Création de la route formulaire de contact ds site.js avec son composant

Création du composant formulaire que je vais intégrer au composant contact

Import du formulaire bootstra

Terminal : npm install --save formik yup pour imposer des restrictions lors du remplissage des formulaires

J’ai décidé d’ajouter un recapcha de google à le fin du formulaire :

Install recapcha ds terminal : npm install react-google-recaptcha

Création de l’id du capcha

**COMPOSANT DE RECHERCHES FILRES**

J’ai créé des éléments d'interface utilisateur tels que des sliders, des sélecteurs (select), et des cases à cocher (checkbox) .

D’ailleurs pour les sliders, je souhaite que l’utilisateur puisse pouvoir gérer ses choix plus facilement en lui proposant un mini et un max pour chaque slider pour qu’il puisse avoir une meilleure expérience , je le ferais pour le kilométrage, prix et année.

Je vais créer plusieurs composants séparés pour chaque fonctionnalité de recherche le checkbox, le dropdown et le slider (composants fils) où l’état sera géré individuellement que je regrouperais par la suite ds 1 seul fichier vehiculeFilters (composant parent).

**BasicRange :**

**Composant de base :**

Je vais créer un curseur de plage de base en utilisant la bibliothèque Material-UI en installant les dépendance nécéssaire pour le bon fonctionnement du composant, je vais ensuite faire copié collé du code

**npm install @mui/material @emotion/react @emotion/styled.**

**npm install @mui/material @mui/icons-material**

Je vais utiliser le destructuring pour extraire les valeurs des props avec des valeurs par défaut.

-**handleChange** => comme un composant parent ne peut pas communiquer directement avec un composant enfant, on utilisera cette fonction qui sera appelée à chaque fois que la valeur du slider change et recevra 2 parmétres nom (pris, km, année) avec la nouvelle valeur séléctionnée.

-**label**=> etiquette qui sera affichée au dessus du slider lorsque le curseur bougera, cela aidera l’utilisateur à mieux comprendre les valeurs qu’il sélectionne.

-**name**=> nom de chaque slider

-**marks**=> tableau d’objets qui spécifira les marques sur le slider dont chaque objets à une valeur et un label. Exemple pour prix : 5000 - 50 000

-**range**=> plage de valeur possible du slider. Je vais utiliser une plage de base qui sera de 0 à 100 car cela conviendra à de nombreuses situations et donc utilisable pour de bons nombres de situations. Je vais faire en sorte que la plus petite valeur soit toujours en première position et que la plus grande soit en dernière en utilisant la fonction **sort** qui permettra de comparer 2 éléments et de les trier.

En sachant que sort quand il n’a pas d’arguments, il va trier des chaines de caractères et comme prix, ce sont des entiers [0,100], voici la fonction que je vais utiliser : range.sort((a,b) => a – b) ;

C’est une fonction de comparaison qui va comparer 2 éléments du tableau et va déterminer leur ordre relatif. En les comparant 2 par 2, il les placera dans l’ordre croissant en fonction du résultat de la comparaison.

**Le basic select :**

Je vais créer ici un composant dropdown avec des options configurables telle que les marques de voitures.

Lorsque l’utilisateur sélectionne une option, on refait la même pour chaque composant, on va réutiliser la fonction handlechange qui va communiquer avec le parent et qui transmettra.

**Composant parent vehiculeFilters :**

J’ai importé Axios pour qu’il puisse effectuer les requêtes.

Pour le slider ‘année’, j ai crée une fonction pour que la date du slider de fin puisse s’actualiser en temps réel, cela évitera de le changer manuellement chaque année.

Je vais ensuite préparer mon composant qui va suivre l’état local des filtres ‘ famille , marque , kilométrage et année’ et ainsi suivre les valeurs que l’utilisateur aura sélectionné.

Là, je vais créer le composant handleChange qui sera passé dans les composant fils en props et mettra à jour l’état local des filtres en fonction des modifications apportées par l’utilisateurs. Elle prendra aussi 2 paramètres name (le nom du filtre à mettre à jour et newValue, la nouvelle valeur du filtre.

Je vais maintenant faire le rendu de tous les composants avec leur propre valeurs.

Chaque filtre sera configuré avec des options spécifiques, telles que les marques , les plages de prix, kilométrage et prix.

**Diffculté : Cette partie de code a été la plus difficile pour moi par rapport aux autres fonctionnalité que j ai pu faire, tout d abord je me suis compliqué la vie à créer moi-même les sliders, checkbox… alors qu’il y a des bibliothèques pour cela mais j’ai encore du mal à faire correctement mes recherches aux bons endroits, heureusement que l’on m’a conseillé et ça m’a bien facilité la vie, maintenant, je le sais et je vais travailler su cela. Ensuite, j ai du mal par rapport à la fonction handleChange et comment le passer en props. Je savais que parent s et enfants ne pouvaintt pas communiquer directement et on m’a suggèrer de créer un pont entre les 2 grâce à cette fonction qui m’a changé vie. J’ai voulu ici commencé par le plus dure car je voulais savoir comment faire ces filtres et chose faite mais j’ai bien galéré.**

\*\*Conclusion\*\*

En fin de compte, la configuration de la base de données a été une étape importante de mon stage, et bien que j'aie rencontré des difficultés, j'ai pu les surmonter en utilisant des méthodes manuelles. Cette expérience m'a permis de développer mes compétences en gestion de bases de données et de mieux comprendre les aspects pratiques du développement

=======

import BasicCheckbox from "../filters/BasicCheckbox";

import BasicSelect from "../filters/BasicSelect";

import BasicRange from "../filters/BasicRange";

//La fonction SearchFilters prend en paramètre une fonction onSearch qui sera appelée lorsque l'utilisateur clique sur le bouton "Rechercher".

const VehiculeFilters = ({ onSearch }) => {

  //Fonction pour obtenir l'année actuelle en utilisant l'objet date.

  const getCurrentYear = () => {

    const dateActuelle = new Date();

    const anneeActuelle = dateActuelle.getFullYear();

    return anneeActuelle;

  };

  const [filtres, setFiltres] = useState({

    // Le composant va suivre l'état local qui est filtres qui va suivre les valeurs séléctionnéées par l'utilisateur.

    famille: [],

    marque: "",

    prix: [5000, 50000],

    kilometrage: [0, 200000],

    annee: [2000, getCurrentYear()], // On appelle la fonction getCurrentYear() pour avoir l'année actuelle

  });

  const handleChange = (name, newValue) => {

    //Intermediaure entre parent et fils pour pouvoir modifier le state car entre fils et parent, c'est pas possible, il sera passer en props pour chaque composant fils

    setFiltres({ ...filtres, [name]: newValue }); //prendra 2 paramètres name (le nom du filtre à mettre à jour et newValue, la nouvelle valeur du filtre.

  };

  const handleCheckBoxChange = (e) => {

    const { name, value, checked } = e.target;

    if (checked) {

      setFiltres({ ...filtres, [name]: [...filtres.famille, value] });

    } else {

      setFiltres({

        ...filtres,

        [name]: filtres.famille.filter((ele) => ele !== value),

      });

    }

  };

  useEffect(() => {

    fetch(

      "http://localhost/GarageBack/API/vehicule.php?kilometremin=0&kilometremax=200000&anneemin=2000&anneemax=2023&prixmin=0&prixmax=50000"

    )

      .then((res) => res.json())

      .then((data) => console.log(data))

      .catch((err) => console.log(err));

  }, []);

  const annee = getCurrentYear();

  return (

    <>

      <div className="search-filters">

        <h2>Recherche par filtres</h2>

        <div className="checkbox-filter">

          <BasicCheckbox

            handleCheckBoxChange={handleCheckBoxChange}

            label="Utilitaire"

            name="famille"

            value="utilitaire"

          />

          <BasicCheckbox

            handleCheckBoxChange={handleCheckBoxChange}

            label="berline"

            name="famille"

            value="berline"

          />

          <BasicCheckbox

            handleCheckBoxChange={handleCheckBoxChange}

            label="familiale"

            name="famille"

            value="familiale"

          />

          <BasicCheckbox

            handleCheckBoxChange={handleCheckBoxChange}

            label="citadine"

            name="famille"

            value="citadine"

          />

          <BasicCheckbox

            handleCheckBoxChange={handleCheckBoxChange}

            label="SUV"

            name="famille"

            value="SUV"

          />

        </div>

        <div className="filter-row">

          <BasicSelect

            label="Marque:"

            name="marque"

            handleChange={handleChange}

            options={[

              { value: "", text: "toutes" },

              { value: "citroen", text: "citroen" },

              { value: "peugeot", text: "peugeot" },

              { value: "kia", text: "kia" },

              { value: "BMW", text: "BMW" },

            ]}

          />

        </div>

        <div className="filter-row">

          <BasicRange

            handleChange={handleChange}

            label="prix :"

            name="prix"

            marks={[

              { value: 5000, label: "5 000 €" },

              { value: 50000, label: "50 000 €" },

            ]} //En variable pour les props, mettre une accolable

            range={[5000, 50000]}

          />

        </div>

        <div className="filter-row">

          <BasicRange

            handleChange={handleChange}

            label="Année :"

            name="annee"

            marks={[

              { value: 2000, label: "2000 " },

              { value: annee, label: `${annee}` },

            ]} //En variable pour les props, mettre une accolable

            range={[2000, annee]}

          />

        </div>

        <div className="filter-row">

          <BasicRange

            handleChange={handleChange}

            label="Kilométrage :"

            name="kilometrage"

            marks={[

              { value: 0, label: "0 km" },

              { value: 200000, label: "200 000 km" },

            ]} //En variable pour les props, mettre une accolable

            range={[0, 200000]}

          />

        </div>

        <button>Rechercher</button>

      </div>

    </>

>>>>>>> filters

  );

};