**Prestige Motors :**

10/07/2023

Une image contenant texte, Police, écriture manuscrite, calligraphie

Description générée automatiquement

Auteurs :

*BINET Florentin*

*DIAW Ahmadou*

*MAGNETTE Arthur*

Table des matières

[I. Objectifs du Site Internet Prestige Motors 3](#_Toc139833424)

[a) Fonctionnement du site internet général 3](#_Toc139833425)

[b) Outils utilisés pour le site web 3](#_Toc139833426)

[II. Spécifications du site internet 3](#_Toc139833427)

[a) Diagramme de cas d’utilisation du site internet 3](#_Toc139833428)

[b) Maquettes des différentes pages du site internet 4](#_Toc139833429)

[c) Schéma relationnel de la base de données 4](#_Toc139833430)

[III. Roadmap du Projet 4](#_Toc139833431)

[IV. Détails techniques des éléments du site internet 5](#_Toc139833432)

[a) Formulaire du client pour la réservation, BACKEND 5](#_Toc139833433)

[b) Fonctionnement du filtre, modification et suppression, BACKEND 10](#_Toc139833434)

# Objectifs du Site Internet Prestige Motors

L’objectif premier du site Internet est la réservation et location de voitures disponibles dans une liste. Les clients peuvent prendre rendez-vous pour la récupérer et devront la rendre après un nombre de jours payés.

## Fonctionnement du site internet général

L’utilisateur arrivera sur la page d’accueil du site internet. Il verra le logo, l’entête, une barre de navigation pour pouvoir se déplacer entre les différentes pages.

L’utilisateur pourra entrer un formulaire pour pouvoir réserver un véhicule de la catégorie qu’il veut, par exemple collection, urbaine, sport, berline, SUV. Il pourra choisir le véhicule selon une liste affichée avec le filtre défini. Il prendra rendez-vous pour pouvoir récupérer le véhicule au garage.

L’employé pourra accéder à une liste de tous les véhicules spécifiés et pourra voir, modifier et supprimer manuellement les réservations.

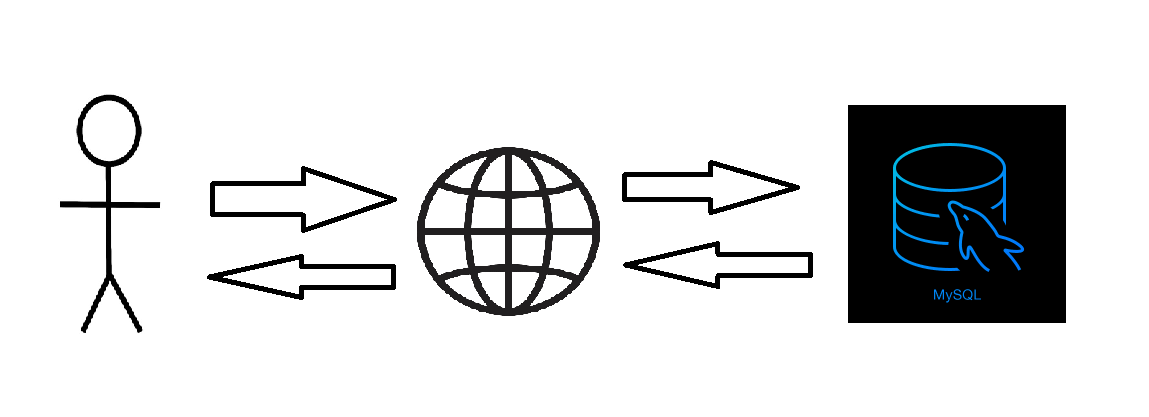
L’utilisateur pourra ensuite naviguer sur les pages A propos et avoir des renseignements approfondis sur le garage et les auteurs du site internet.

## Outils utilisés pour le site web

Le site web sera écrit en html et php et sera stylisé en CSS. Il tournera de manière locale sur XAMPP. Il sera animé par Javascript sur certaines pages pour permettre certaines actions possibles.

L’accès à la base de données MySQL se fera sur le serveur distant hébergé par Florentin sur instance OVH. Nous sommes venus avec cette solution car il fallait centraliser la base pour éviter d’avoir des différences d’enregistrement dans la base de données entre les différents membres du groupe. Nous avons géré et ajouté les données, création de tables grâce à My SQL Workbench.

Schéma du fonctionnement du site internet :



## Utilisation de GIT

Pour le projet, nous avons mis en place un référentiel git pour pouvoir avoir une trace de l’historique et faire des modifications et les envoyer au serveur et avoir différentes versions. Nous avons fait différentes branches et mis en place une protection sur la branche Master pour éviter les erreurs. Il fallait faire une pull request, faire une review change pour accepter et enfin merge. On a une deuxième branche où l’on a fait la majorité des développements et une branche test pour tester certains ajouts.

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

# Spécifications du site internet

## Diagramme de cas d’utilisation du site internet

Une image contenant texte, capture d’écran, Police, blanc

Description générée automatiquement

## Maquettes des différentes pages du site internet

Une image contenant texte, capture d’écran, Police, Site web

Description générée automatiquement

Page d'accueil index.php 1

Ici, l’utilisateur est accueilli sur la page d’accueil. Il peut lire le descriptif de ce site web. Il peut naviguer dans la barre de navigation pour voir les différentes pages. Il peut regarder la liste des véhicules en images. Il peut aussi réserver son véhicule dans l’onglet Réservation ou il peut le faire en cliquant sur « ici ».

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Page de Réservation reservation.php 1

L’utilisateur devra remplir tous les champs du formulaire. Il sera amené à choisir la catégorie de véhicule qu’il veut obtenir et son modèle de véhicule. En cliquant sur « J’ai fini ! », son véhicule sera réservé.

Une image contenant texte, Police, Page web, logiciel

Description générée automatiquement

Page de changement de RDV modifrdv.php 1

Ici, le client sera amené à entrer son adresse mail considéré comme unique pour pouvoir modifier sa ou ses réservations de véhicules. Il pourra aussi les annuler.

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Page de contact contact.php 1

C’est la page contact du site Internet. Dès qu’on clique sur l’adresse mail, on est amené à sélectionner le client de mail pour que l’on puisse envoyer un mail à Prestige Auto.

Une image contenant texte, Appareils électroniques, capture d’écran, Page web

Description générée automatiquement

Page apropos.php 1

Ici, c’est du HTML et CSS classique et sur cette page, on peut en apprendre plus sur les tâches qu’ont effectué les membres du groupe.

## Schéma relationnel de la base de données

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

# Roadmap du Projet

# Détails techniques des éléments du site internet

## Formulaire du client pour la réservation, BACKEND

Une image contenant texte, capture d’écran, affichage, ordinateur

Description générée automatiquement

Ce fichier HTML contient un formulaire qui permet aux utilisateurs de soumettre leurs informations.

* Les scripts jQuery et **detect\_model.js** sont inclus via les balises **<script>**.
* Le formulaire principal est défini avec l'ID **main-form** et il envoie les données à **../php/formulaire.php** en utilisant la méthode POST.
* Les champs de formulaire tels que le nom, l'email, le téléphone, l'adresse et la date de rendez-vous sont requis (**required**).
* Il y a un composant secondaire qui contient des champs de sélection pour la catégorie de

véhicule et le modèle de véhicule. L'élément **select** avec l'ID **vehicle-model** n'a pas d'options statiques. Les options seront ajoutées dynamiquement en utilisant le script **detect\_model.js**. Quand on sélectionne une catégorie de voitures, par exemple SUV, le modèle du véhicule est dynamique. Les véhicules seront affichés dans la catégorie SUV. Il y a un deuxième formulaire qui est en commentaire sur la page HTML (**sub-form**) qui n'a pas de contenu à l'intérieur au démarrage, mais il enverra les données à **../php/get\_model.php** lorsqu'il sera utilisé.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement Avant SélectionUne image contenant texte, capture d’écran, Police, nombre

Description générée automatiquementAprès sélection

Quand on sélectionne une catégorie de voitures, par exemple SUV, le modèle du véhicule est dynamique. Les véhicules seront affichés dans la catégorie SUV.

* Il y a également un champ de formulaire caché (**input** de type **hidden**) avec l'ID **vehicle-id**, qui stockera l'ID du véhicule sélectionné.

Cette page html est écoutée par un script Javascript et la technologie jQuery pour détecter quel est la catégorie de véhicule affichée à l’écran.



Ce script utilise jQuery et repose sur l'événement "document ready" pour s'assurer que le code JavaScript s'exécute une fois que le document HTML est complètement chargé. Les points clés du script sont :

* La fonction **updateVehicleModels()** est définie pour mettre à jour la liste des modèles de véhicules en fonction de la catégorie de véhicule sélectionnée.
* Lorsque la catégorie de véhicule est modifiée (**change**), la fonction **updateVehicleModels()** est appelée pour récupérer les modèles correspondants via une requête AJAX.
* La requête AJAX est envoyée au fichier **../php/get\_model.php** en utilisant la méthode POST. La valeur de la catégorie de véhicule est envoyée en tant que paramètre dans l'objet de données.
* En cas de succès, la fonction **success** est exécutée. Elle réinitialise d'abord la liste des modèles de véhicules, puis ajoute des options pour chaque modèle récupéré à partir des données JSON renvoyées par le fichier PHP.
* Lorsqu'un modèle de véhicule est sélectionné (**change**), la fonction anonyme associée est appelée. Elle récupère l'ID du modèle sélectionné et met à jour la valeur du champ de formulaire caché (**#vehicle-id**).
* Un message est également affiché dans la console pour afficher l'ID du véhicule sélectionné à des fins de débogage.
* Enfin, l'appel initial de **updateVehicleModels()** est effectué pour mettre à jour les modèles de véhicules en fonction de la catégorie de véhicule par défaut.

Le fonctionnement du script php get\_model est le suivant :

* Une connexion à la base de données est établie en utilisant les informations d'identification appropriées (serveur, nom d'utilisateur, mot de passe et nom de la base de données).
* Si la connexion échoue, un message d'erreur est affiché et le script se termine.
* La catégorie de véhicule sélectionnée est récupérée à partir de la requête AJAX POST avec la clé "vehicle-category".
* Une requête SQL est exécutée pour sélectionner les modèles de véhicules correspondant à la catégorie spécifiée. La requête utilise la clause WHERE pour filtrer les véhicules disponibles (**dispo = 1**) et la catégorie de véhicule.

SELECT id,marque, modele, annee, couleur FROM voitures WHERE dispo = 1 AND classe = '$vehicleCategory'

Les résultats de la requête sont parcourus, et pour chaque modèle de véhicule, les informations pertinentes (ID, marque, modèle, année, couleur) sont extraites et ajoutées à un tableau.



* La connexion à la base de données est fermée.
* Les données des modèles de véhicules sont renvoyées au format JSON en utilisant la fonction **json\_encode()**.
* Le résultat JSON est renvoyé en tant que réponse à la requête AJAX effectuée dans le script JavaScript.

Ce fichier PHP (**formulaire.php**) est responsable de la gestion du formulaire de réservation. Voici les points clés du script :

* Les informations de connexion à la base de données sont configurées (**$servername**, **$username**, **$password** et **$database**).
* Une connexion à la base de données est établie en utilisant les informations d'identification fournies.
* Si la connexion échoue, un message d'erreur est affiché et le script se termine.
* Les données du formulaire sont récupérées à l'aide de la méthode POST (**$\_POST**).
* Une transaction est démarrée pour garantir l'intégrité des données lors de l'insertion dans les tables.Le script effectue les opérations suivantes :
  + Vérifie si l'email existe déjà dans la table 'clients'. Si oui, récupère l'ID correspondant.
  + Sinon, insère les données du client dans la table 'clients' et récupère l'ID généré.  
    INSERT INTO clients (nom, email, telephone, adresse) VALUES ('$name', '$email', '$phone', '$address')
  + Insère les données de réservation dans la table 'rdv' avec l'ID du client et l'ID du véhicule sélectionné.  
    INSERT INTO rdv (client\_id, voiture\_id, date\_reservation) VALUES ('$clientId', '$vehicleId', '$appointmentDate')
  + Met à jour la disponibilité du véhicule dans la table 'voitures' en le marquant comme non disponible (**dispo = 0**).  
    UPDATE voitures SET dispo = 0 WHERE id = '$vehicleId'
* Si toutes les opérations sont réussies, la transaction est validée et un message de réussite est affiché. Un bouton est également affiché pour revenir à la page précédente.
* Si une erreur se produit à tout moment, la transaction est annulée et un message d'erreur approprié est affiché. Un bouton est également affiché pour revenir à la page précédente.

## Fonctionnement du filtre, modification et suppression, BACKEND



Ce fichier HTML contient un formulaire de filtre permettant de sélectionner les rendez-vous d'un client spécifique en entrant son adresse e-mail. Voici les points clés à noter :

* Le formulaire envoie les données via la méthode GET vers le fichier **../php/update\_bdd.php**.
* Un champ de saisie (**input** de type **text**) est utilisé pour entrer l'adresse e-mail du client à filtrer.
* Le champ de saisie est requis (**required**) et est identifié par l'ID **Filtre**.
* Un champ de formulaire caché (**input** de type **hidden**) avec le nom **list\_records** est inclus et défini sur la valeur "true". Cela permettra au script PHP de reconnaître que la demande concerne l'affichage des rendez-vous du client.
* Un bouton de soumission (**input** de type **submit**) est affiché avec l'étiquette "Vos RDV" pour envoyer le formulaire.

Le fichier php update\_bdd.php est appelé dès que l’on clique sur le bouton submit. Sa logique est :

* La session est démarrée en utilisant **session\_start()** pour stocker l'e-mail du client dans une variable de session (**$\_SESSION['email']**).
* Une connexion à la base de données est établie en utilisant les informations d'identification fournies.
* Si la connexion échoue, un message d'erreur est affiché et le script se termine.
* L'e-mail du client est récupéré à partir de la variable **$\_GET['Filtre']** ou est initialisé à une chaîne vide si aucune valeur n'est fournie.
* Une requête SQL est préparée pour récupérer les données des rendez-vous du client correspondant à l'e-mail.  
  
* La valeur de l'e-mail est liée au paramètre de la requête préparée et la requête est exécutée.
* Les résultats de la requête sont récupérés à l'aide de la méthode **get\_result()**.
* Si des rendez-vous sont trouvés, les données sont affichées sous forme de tableau.  
  
* Pour chaque ligne de rendez-vous, des boutons "Modifier" et "Supprimer" sont affichés. Les boutons sont associés à des fonctions JavaScript (**modifyRow()** et **deleteRow()**) définies dans le fichier JavaScript **update\_delete.js**.  
  
* Des champs de formulaire cachés sont inclus pour stocker les informations nécessaires à la modification ou à la suppression d'un rendez-vous spécifique (ID du rendez-vous, ID de la voiture, date de réservation et e-mail du client).  
  
* Un bouton est également affiché pour revenir à la page précédente.
* Si aucun rendez-vous n'est trouvé, un message approprié est affiché.
* Si une modification a été effectuée via un formulaire de modification, un message de réussite est affiché.
* Les ressources JavaScript sont incluses à la fin du fichier.  
  

Le formulaire de modification est géré par le fichier update\_delete.js. Ce fichier gère aussi deux scripts php qui font la modification des lignes et la suppression des lignes.

Le résultat général du filtre sans CSS est :

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Ce fichier JavaScript (**update\_delete.js**) contient deux fonctions :

1. La fonction **modifyRow(id)** est utilisée pour modifier une ligne de rendez-vous. Voici comment elle fonctionne :

* La fonction prend en paramètre l'ID de la ligne de rendez-vous à modifier.
* Tout d'abord, elle vérifie si un formulaire de modification existe déjà en recherchant un élément avec la classe **.modification-form**. Si un formulaire existe, il est supprimé.
* Les valeurs actuelles de la ligne de rendez-vous (voiture, date de réservation, e-mail) sont récupérées à partir des éléments correspondants dans le document.
* Ensuite, un formulaire est créé dynamiquement en utilisant la méthode **createElement()** et les attributs appropriés sont définis (classe, méthode, action).
* Un champ caché est ajouté au formulaire pour stocker l'ID du rendez-vous à modifier.
* Une requête AJAX est effectuée pour récupérer les véhicules disponibles. Le résultat de la requête est analysé en tant que JSON.
* Une combobox est créée pour permettre la modification du véhicule. Les options correspondant aux véhicules disponibles sont ajoutées à la combobox.
* La valeur de la combobox est définie sur le véhicule actuel de la ligne de rendez-vous.
* Un champ de date est ajouté au formulaire pour permettre la modification de la date de réservation.
* Un champ caché est ajouté pour stocker l'e-mail du client.
* Un bouton de soumission et un bouton d'annulation sont ajoutés au formulaire.
* Enfin, le formulaire est inséré au début du document.

1. La fonction **deleteRow(id)** est utilisée pour supprimer une ligne de rendez-vous. Voici comment elle fonctionne :

* La fonction prend en paramètre l'ID de la ligne de rendez-vous à supprimer.
* Elle demande une confirmation à l'utilisateur à l'aide de la fonction **confirm()**.
* Si l'utilisateur confirme, un formulaire est créé dynamiquement avec les attributs appropriés (méthode, action).
* Un champ caché est ajouté au formulaire pour stocker l'ID du rendez-vous à supprimer.
* Un champ caché est ajouté pour stocker l'e-mail du client.
* Le formulaire est ajouté à la page et soumis.

La fonction **modifyRow(id)** après avoir construit le formulaire appelle **update.php**.

* Tout d'abord, il démarre une session et utilise la fonction **ob\_start()** pour mettre en mémoire tampon la sortie.
* Ensuite, il établit une connexion à la base de données en utilisant les informations de connexion.
* Les données du formulaire sont récupérées à l'aide de la superglobale **$\_POST**.
* L'e-mail du client est récupéré depuis la session créée sur **update\_bdd.php**.
* Une requête SQL est exécutée pour désactiver l'ancien véhicule associé au rendez-vous, en mettant à jour la colonne **dispo** de la table **voitures**.  
  
* Les données du rendez-vous sont mises à jour dans la table **rdv** en utilisant une autre requête SQL.  
  
* Le nombre de lignes affectées par la mise à jour est récupéré.
* Une requête est effectuée pour récupérer les données mises à jour à partir des tables **clients**, **voitures** et **rdv**.
* On vérifie si une ligne a été mise à jour avec succès et si des résultats ont été obtenus.
* Si la mise à jour a été effectuée avec succès et qu'il y a des résultats, un message "Data Inserted and modified." est affiché.
* Si aucune ligne n'a été mise à jour ou s'il n'y a pas de résultats, un message "Data Not Inserted." est affiché.
* Ensuite, tout contenu HTML précédent est nettoyé à l'aide de **ob\_end\_clean()** pour éviter toute sortie non souhaitée.
* Enfin, l'utilisateur est redirigé vers la page précédente (**update\_bdd.php**) avec l'e-mail du client en tant que paramètre GET.  
  

La fonction **deleteRow(id)** après avoir construit le formulaire appelle **update.php**.

* Tout d'abord, il démarre une session et utilise la fonction **ob\_start()** pour mettre en mémoire tampon la sortie.
* Ensuite, il établit une connexion à la base de données en utilisant les informations de connexion.
* Les données de la session sont récupérées pour obtenir le courriel du client.
* La variable POST **id** est vérifiée pour s'assurer qu'elle est définie.
* Ensuite, une requête SELECT est exécutée pour vérifier la correspondance de l'identifiant de rendez-vous avec une ligne dans la table **rdv**. Si aucune correspondance n'est trouvée, un message d'erreur est affiché.  
  
* Les valeurs de l'identifiant de rendez-vous sont affichées à des fins de débogage.
* Une autre requête SELECT est exécutée pour récupérer l'identifiant du véhicule associé au rendez-vous.  
  
* Si aucun véhicule n'est associé à l'identifiant de rendez-vous, un message d'erreur est affiché.
* La disponibilité du véhicule est mise à jour dans la table **voitures** en utilisant une requête UPDATE.  
  
* Une requête DELETE est utilisée pour supprimer la ligne de rendez-vous de la table **rdv**.  
  
* Si une erreur se produit lors de la mise à jour de la disponibilité du véhicule ou de la suppression du rendez-vous, un message d'erreur est affiché.
* Si la suppression est effectuée avec succès, un message "Suppression effectuée avec succès." est affiché.
* La connexion au serveur de base de données est fermée.
* La fonction **ob\_end\_clean()** est utilisée pour s'assurer qu'aucun contenu HTML n'est envoyé avant la redirection.
* Enfin, l'utilisateur est redirigé vers la page précédente (**update\_bdd.php**) avec l'e-mail du client en tant que paramètre GET.



# Conclusion et Remerciements

Au cours de ce projet, nous avons beaucoup appris en termes de programmation, d’interrogation en base de données, de travail en groupe et de pouvoir gérer son projet sur GitHub. Nous vous remercions d’avoir lu ce document pour mieux comprendre l’élaboration de ce projet.