

Un bureau virtuel de gestion des locations de voitures

Projet PWEB 2020 – Rapport

Hamilcaro Chloé 209

I – PRÉAMBULE

A – Présentation du projet

Ce projet s'intitule « Un bureau virtuel de gestion des locations de voitures », et consiste en le développement d'une application MVC de gestion de locations. Cette application repose sur le langage **PHP** et **SQL**.

Le **MVC** est un **motif d'architecture** logicielle qui consiste à séparer les différents fichiers d'une application en trois répertoires : un répertoire `model` pour les fichiers PHP qui interagissent avec la base de données MySQL ; un répertoire `vue` qui contient les templates, les feuilles de style, et tout ce qui rentre dans le cadre de la partie visuelle du site ; et un répertoire `control` qui contient tous les autres fichiers PHP.

Ainsi, **mon application est structurée en MVC**, même si cela n'était pas le cas lors de la pré-soutenance.

Voici la liste des **fonctionnalités** que j'ai développées :

ENTREPRISES NON-ABONNÉES	ENTREPRISES ABONNÉES	LOUEURS
Afficher le site et voir la liste des véhicules avec photos et caractéristiques	S'inscrire / Se connecter	S'inscrire / Se connecter
S'inscrire / Se connecter	Afficher sa flotte de véhicules en cours de location avec dates de début et fin	Afficher les véhicules de son stock
	Consulter ses factures	Afficher ses véhicules en cours de location (flottes des clients)
		Entrer une voiture dans le stock via un formulaire
		Consulter ses factures

Toutes les fonctionnalités présentées sont opérationnelles.

Les services en rouge ont été ajoutés après le pré-rapport en Octobre.

J'ai utilisé le framework **Bootstrap**, la bibliothèque de scripts JavaScript **jQuery**, et ajouté une feuille de style CSS personnelle pour la mise en page du site web, qui répond aux critères du responsive design (j'ai notamment ajouté un menu déroulant).

B – Informations utiles

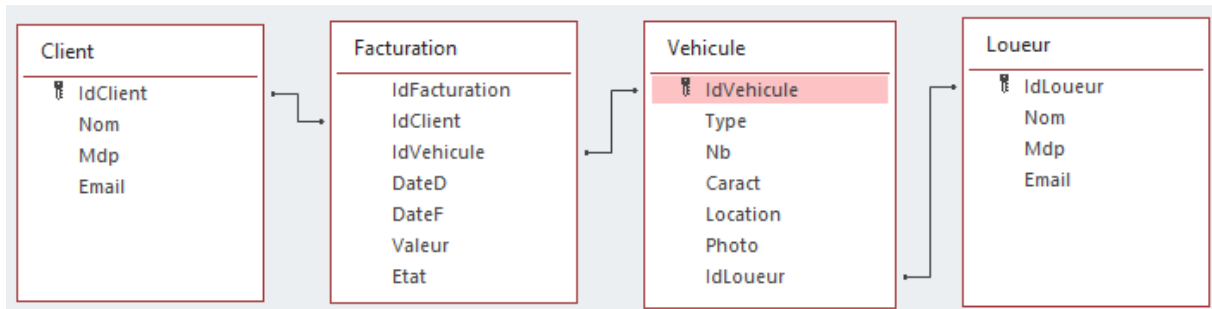
Pour tester les fonctionnalités déjà implémentées, voici des identifiants valides :

	Adresse Mail	Mot de passe
Une entreprise cliente	richard.cooper@societe.fr	RC202020
Un loueur	diva@gmail.com	DAVIDDIVA

Il ne faut également pas oublier de modifier le fichier `connect.php` (`modele/connect.php`) pour remplacer le mot de passe d'accès à MySQL par le vôtre.

C – Schéma de la base de données

Voici le schéma de la base MySQL :



Le champ « `Caract` » de la table « `Vehicule` » est un élément json.

La table `Facturation` sert à la fois à l’affichage des factures et des locations en cours.

II – RÉALISATIONS

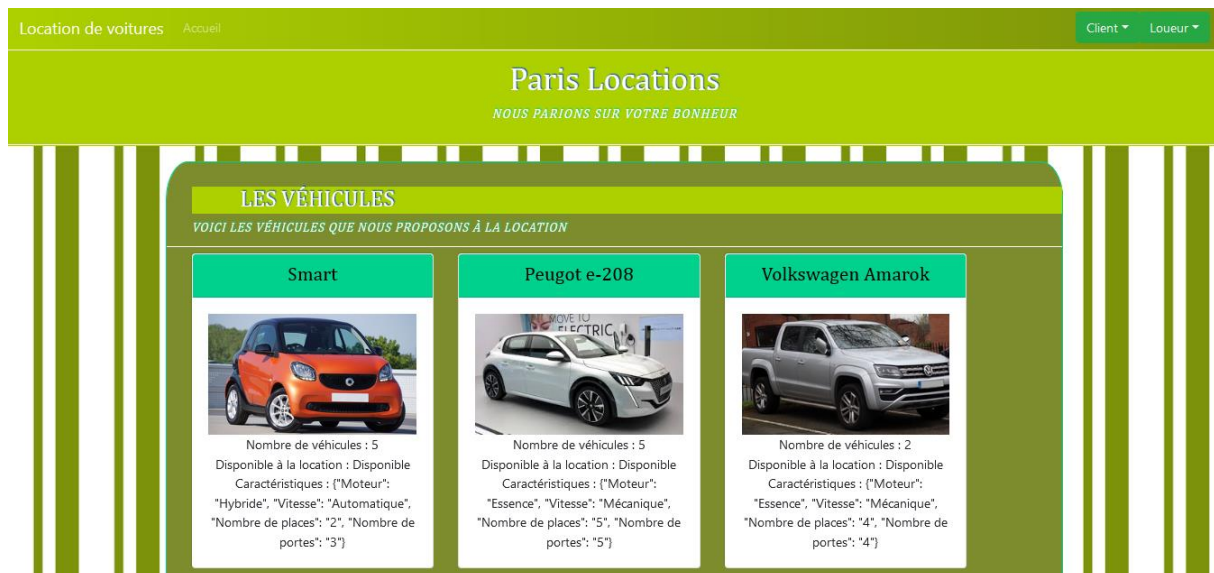
A – Entreprises non-abonnées / Visiteurs

Les fonctionnalités dédiées aux **entreprises non-abonnées**, aussi dénommées « utilisateurs » dans mon code MVC dans le sens où ce sont des utilisateurs qui ne se sont pas encore identifiés, ont été assez simple à réaliser.

a. Accueil

Mon application repose sur l'hypothèse que, dès lors que le site web s'affiche, le profil de l'utilisateur n'est pas défini : ainsi, il s'agit d'une entreprise ou d'un loueur non-connecté au service.

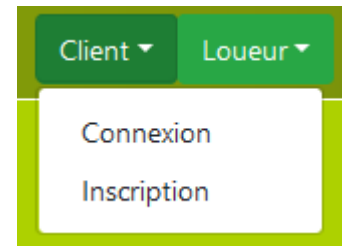
L'**accueil** de mon application se présente donc sous la forme d'**une page avec un menu**, l'affichage des différents services de connexion et d'inscription détaillés plus bas dans ce rapport, et l'affichage de l'**intégralité des véhicules** enregistrés dans la base, qu'ils soient disponibles ou non à la location.



Accueil de mon application

Sur cet écran, le visiteur a donc la possibilité de consulter l'intégralité des voitures disponibles dans la base, avec le nombre de véhicules, le fait qu'ils soient disponible à la location ou non, et leurs caractéristiques récupérées à partir du champ **json** de la base de données.

Une **barre de menu fixe** s'affiche en haut de l'écran : comme l'utilisateur n'est pas encore connecté, il peut soit retourner à l'accueil du site web en cliquant sur le lien à gauche, soit **s'identifier ou s'inscrire** en tant que loueur ou client grâce aux boutons à droite. Ces deux boutons sont des boutons avec menu déroulants, implémentés grâce à Bootstrap.



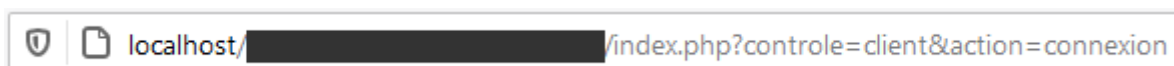
Boutons déroulants : connexion ou inscription

b. Connexion : pour les loueurs et les clients

Tout en haut de cette page d'accueil se trouve donc une division qui permet à un client ou loueur de s'identifier ou de s'inscrire. Ces liens sont tous conçus de la manière suivante :

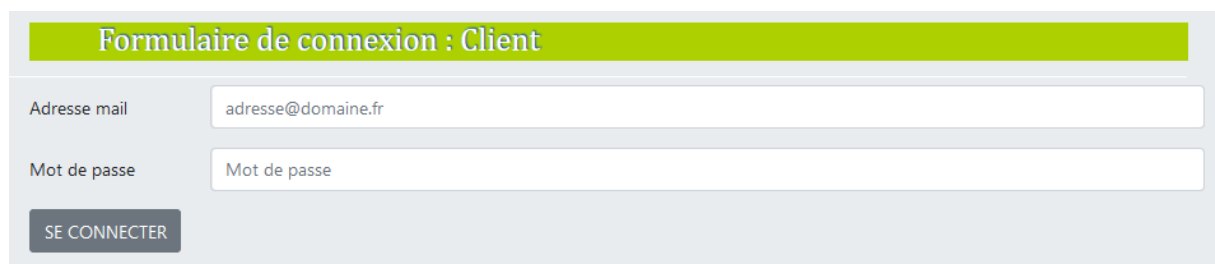
```
<a href="index.php?controle=client&action=connexion"></a>
```

L'application étant structurée en MVC, tous les services passent en effet par le fichier `index.php`, fichier qui se lance dès que le répertoire de l'application est ouvert.



Barre d'adresse : l'accès au service se fait bien par index.php

Le service de connexion pour le client et le loueur sont **opérationnels**, et fonctionnent de la même manière, seul le texte affiché diffère. Ici, la barre de menu trouve toute son utilité : si l'utilisateur ne souhaite finalement plus se connecter, mais seulement regarder les voitures disponibles à la location, il peut choisir de retourner à l'accueil.



Service de connexion : exemple du Client

Formulaire de connexion : Client

Adresse mail
richard.cooper@societe.fr

Mot de passe
●●●●●●

SE CONNECTER

Les informations saisies ne permettent pas de vous identifier

Lorsque les identifiants saisis **ne permettent pas** d'identifier l'utilisateur, un **message d'erreur** s'affiche en bas du formulaire, invitant l'utilisateur à réessayer de se connecter.

La fonctionnalité de connexion fait appel à la base de données, et exécute une requête pour savoir si la combinaison de l'adresse mail et du mot de passe renvoient à un utilisateur inscrit dans la base.

Service de connexion : la combinaison de l'adresse mail et du mot de passe n'ont pas permis d'identifier le client

c. Inscription : pour les loueurs et les clients

Un utilisateur peut également choisir de **s'inscrire**, en tant qu'entreprise cliente ou loueur. Cette **fonctionnalité** est **également opérationnelle**.

Les deux formulaires d'inscription fonctionnent également de la même manière, seul leur affichage diffère.

Tout comme pour le formulaire de connexion, **différentes erreurs sont gérées**. Tout d'abord, le code va vérifier si un utilisateur ayant les mêmes identifiants existe déjà dans la **base de données**. Si c'est le cas, un message s'affiche en bas du formulaire :

Formulaire d'inscription : Client

Nom
NOM

Adresse mail
adresse@domaine.fr

Mot de passe
Mot de passe

ENREGISTRER

Service d'inscription : exemple du Client

Formulaire d'inscription : Client

Nom

Richard Cooper

Adresse mail

richard.cooper@societe.fr

Mot de passe

.....

ENREGISTRER

L'utilisateur existe déjà

Tout comme pour le formulaire de connexion, le formulaire d'inscription **vérifie si** un utilisateur répondant au même nom et utilisant la même adresse mail **existe déjà dans la base** de données, en exécutant une requête de vérification. Si l'utilisateur existe déjà, l'inscription ne s'effectue pas.

La **validité** du nom, de l'adresse mail, du mot de passe sont gérées de différentes manières. Tout d'abord directement depuis la page HTML avec l'utilisation des attributs **required** dans le **formulaire** :

```
<input class="form-control col-sm-10" name="email" type="email" value="<?php echo($email);?>" placeholder="adresse@domaine.fr" pattern="(.)@(.)\" pattern="(.)@(.)\" required="required"/><br/>
```

Nom

NOM

Veuillez compléter ce champ.

Conséquence de required : si l'utilisateur valide le formulaire sans avoir rempli ce champ, un message s'affiche à l'écran

Ensuite, j'ai créé une fonction pour m'assurer qu'aucune chaîne de caractère ne soit vide et j'ai utilisé la fonction **preg_match** pour m'assurer que le mot de passe ait au minimum 8 caractères.

Si ce n'est pas le cas, un nouveau message d'erreur apparaît sous le formulaire, à la place de l'ancien s'il y en avait un :

Nom

JP

Adresse mail

jean@paris.fr

Mot de passe

...

ENREGISTRER

Champs incorrectement remplis, merci de vérifier ce que vous avez rempli.

Pour rappel :

Pas de chaîne vide

Adresse mail contient un @

Mot de passe entre 8 et 10 caractères

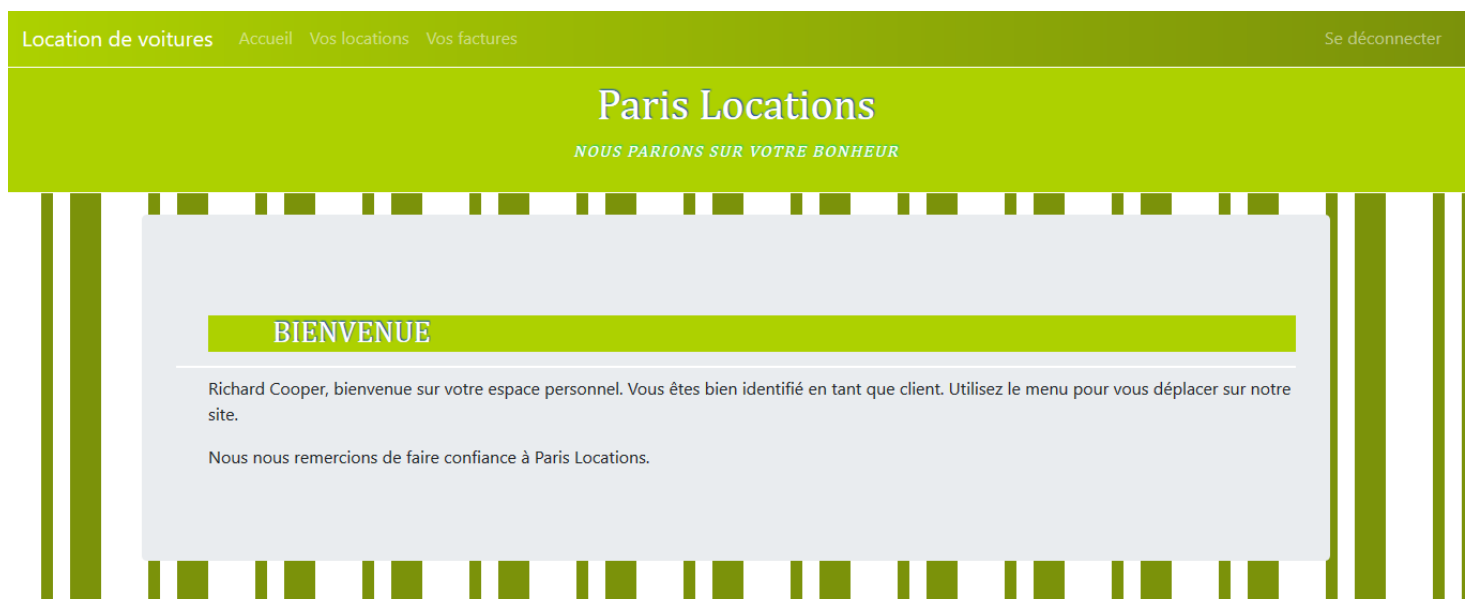
Si toutes les données saisies sont correctes, le nouveau client ou loueur est inscrit

dans la base de données. De plus, une **inscription réussie équivaut à une connexion** : ainsi, si l'inscription ou l'identification d'un client ou d'un loueur sont correctes, une page d'accueil spécifique à chaque type d'utilisateur s'affiche à l'écran. Cette fonctionnalité est développée dans la prochaine partie.

B – Entreprises clientes

J'ai développé plusieurs services pour les **clients**, en plus de l'identification et de l'inscription : l'affichage d'un **accueil personnalisé** après une identification ou une inscription réussie, l'affichage des **locations en cours** et des **factures**.

a. Accueil et menu spécifiques : Clients



Cette page d'accueil affiche un **message de bienvenue personnalisé** avec le nom du client, et l'informe du succès de sa connexion.

Un **menu spécifique** s'affiche à la place de l'ancien, avec la liste des fonctionnalités propres à un client que j'ai pu développer : retourner à l'accueil, consulter ses locations, ses factures, et bien évidemment se déconnecter.

b. Affichage des locations en cours

Tout d'abord, les entreprises clientes peuvent **consulter leur flotte de véhicules** dont la location est en cours.


Les **véhicules sont classés par ordre de date de début de location** croissante : les locations les plus anciennes ont généralement une date d'échéance plus proche que celles ayant commencé ensuite, l'utilisation d'un **ORDER BY** à la fin de la requête qui sélectionne les locations du client est donc justifiée.

Location de voitures
Accueil
Vos locations
Vos factures
Se déconnecter

Paris Locations
NOUS PARIONS SUR VOTRE BONHEUR


Liste de vos locations en cours :

Hyundai i20



Caractéristiques : {"Moteur": "Essence", "Vitesse": "Mécanique", "Nombre de places": "5", "Nombre de portes": "5"}
Loué du 2020-11-01 au 2020-12-01

Peugot e-208



Caractéristiques : {"Moteur": "Essence", "Vitesse": "Mécanique", "Nombre de places": "5", "Nombre de portes": "5"}
Loué du 2020-11-06 au 2020-11-15

c. Affichage des factures

Enfin, un client peut choisir d'**afficher ses factures**, c'est-à-dire ses locations passées qu'il doit – ou a – payé.

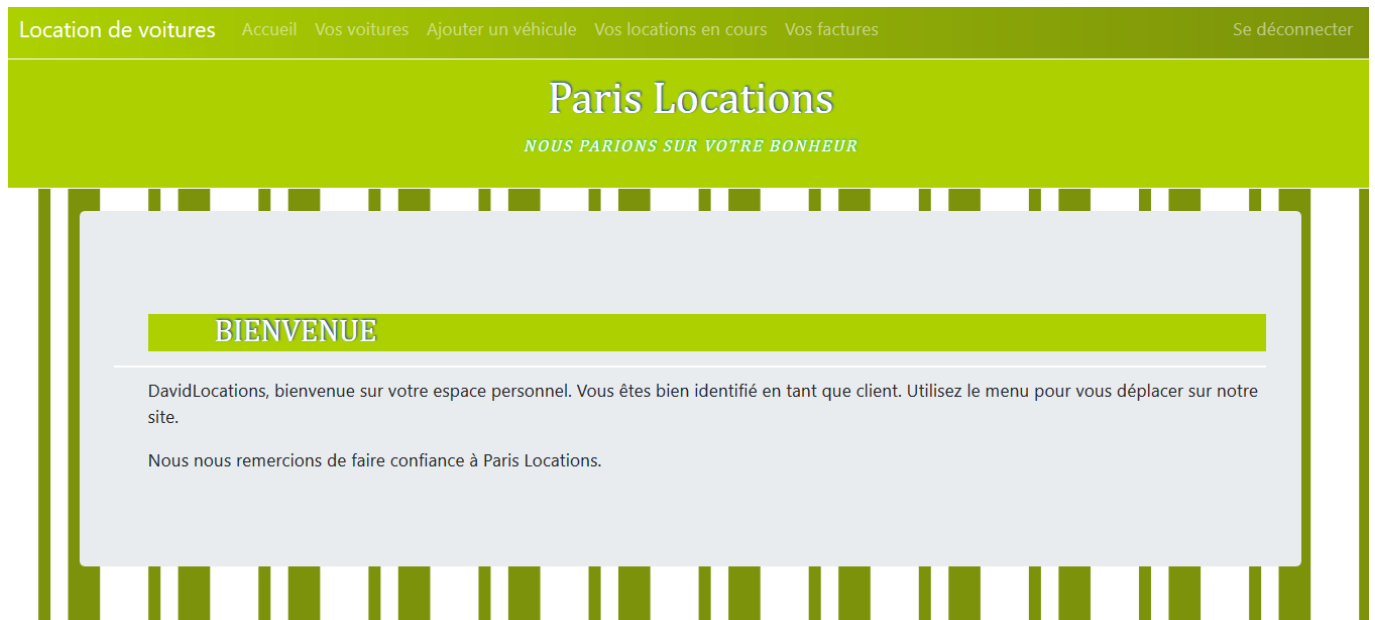
Ces données sont récupérées depuis la table « Facturation », et disposées dans un **tableau**. Le montant est à la fin du tableau, pour une meilleure lisibilité : c'est l'information que l'on cherche en premier lorsqu'on consulte une facture.

Date	Fournisseur	Véhicule	Statut	Montant
2020-10-25	DavidLocations	Dacia Duster	Impayée	800 €
2020-11-04	DavidLocations	Mercedes Citan	Payée	500 €

C – Loueurs

J'ai développé plusieurs services pour les **loueurs**, en plus de l'identification et de l'inscription : l'affichage d'un **accueil personnalisé** après une identification ou une inscription réussie, l'**affichage des véhicules** qu'il propose à la location, de ses **locations en cours** et de ses **factures**. Le loueur a également la possibilité d'**ajouter un véhicule**.

a. Accueil et menu spécifiques : Loueurs



Cette page d'accueil affiche un **message de bienvenue personnalisé** avec le nom du client, et l'informe du succès de sa connexion.

Un **menu spécifique** s'affiche à la place de l'ancien, avec la liste des fonctionnalités propres à loueur. Il peut ainsi voir ses voitures, ses locations en cours, ses factures, ajouter un véhicule et se déconnecter. Tout comme le menu du client, un clic sur ce bouton ne renvoie pas seulement à la page d'accueil principale, mais **supprime** également la **variable session profil**.

b. Affichage des véhicules disponibles à la location

L'affichage des **véhicules du loueur** suit l'affichage de toutes les voitures sur la page d'accueil principale, seule la requête change : elle prend en compte l'identifiant du loueur pour n'afficher que les voitures qui lui appartiennent.

VOS VÉHICULES

[Ajouter un véhicule](#)

Ford Fiesta



Nombre de véhicules : 20
 Disponible à la location : Disponible
 Caractéristiques : {"Moteur":
 "Essence", "Vitesse": "Mécanique",
 "Nombre de places": "5", "Nombre
 de portes": "5"}

Hyundai i20



Nombre de véhicules : 80
 Disponible à la location : Disponible
 Caractéristiques : {"Moteur":
 "Essence", "Vitesse": "Mécanique",
 "Nombre de places": "5", "Nombre
 de portes": "5"}

Renault Kadjar



Nombre de véhicules : 5
 Disponible à la location : Disponible
 Caractéristiques : {"Moteur":
 "Essence", "Vitesse": "Automatique",
 "Nombre de places": "5", "Nombre
 de portes": "5"}

c. Ajout d'un véhicule

Un des points importants de ce projet a été **l'enregistrement d'un véhicule** dans la base de données.

Les deux points importants à gérer ont été **l'import d'image** et son enregistrement dans la base de données, et **l'enregistrement des caractéristiques** au format **json**.

Location de voitures

Accueil Vos voitures Ajouter un véhicule Vos locations en cours Vos factures
Se déconnecter

Paris Locations

NOUS PARIONS SUR VOTRE BONHEUR

Ajout d'un véhicule

Nom/Type du véhicule

Caractéristiques

Moteur

Vitesse

Nombre de places

Nombre de portes

Nombre de véhicules

Image

Aucun fichier sélectionné.

Je me suis d'abord occupée de l'import de l'image. Pour cela, il a fallu modifier la structure du formulaire et ajouter `enctype="multipart/form-data"`. Mon principal problème a été la lecture de ce fichier, qui semblait poser problème et était dû à une mauvaise rédaction de ma part. Une fois cela géré et la sauvegarde du fichier opérationnelle grâce à l'utilisation de la fonction php qui permet de copier un fichier vers un autre dossier :

```
move_uploaded_file($image['tmp_name'], $dossier . $image['name']);
```

Ainsi, l'**image téléchargée** par l'utilisateur est récupérée puis copiée dans le dossier prévu à cet effet dans mon application.

J'ai utilisé `preg_match` sur le nom du fichier pour s'assurer qu'il s'agissait bien d'une image et ajouter une **sécurité** à l'application. Si ce n'est pas le cas, un **message d'erreur** apparaît :

Erreur dans la saisie :
Le type du véhicule doit comporter au moins 5 caractères.
L'image doit être au format .gif, .png, .jpg, .jpeg, .svg

Si l'image n'est pas au bon format, un message d'erreur apparaît sous le formulaire

Ensuite, je me suis occupée de l'**enregistrement des données au format json**, récupérées de quatre champs du formulaire.

Pour cela, j'ai tout d'abord inséré les données dans un **tableau associatif** :

```
$caract= array(  
    'Moteur' => trim($moteurV),  
    'Vitesse' => trim($vitesseV),  
    'Nombre de places' => $nbplacesV,  
    'Nombre de portes' => $nbportesV  
);
```

Puis, la fonction chargée d'ajouter le véhicule dans la base de données transforme ce tableau associatif au **format json** :

```
$commande->bindValue(':Caracteristiques', json_encode($car));
```

J'ai utilisé `bindValue` au lieu du traditionnel `bindParam`, ce dernier occasionnant une erreur quand il est utilisé avec un tableau json.

d. Affichage des véhicules loués

Je me suis par la suite occupée de l'affichage des **véhicules en cours de location**. Pour cela, j'ai utilisé la requête suivante, qui me permet de ne sélectionner que les factures pour lesquelles la date de fin de location n'est pas encore atteinte (elles ne sont donc pas encore à payer) :

```
$sql="SELECT * FROM vehicule INNER JOIN (facturation INNER JOIN client ON facturation.IdClient=client.IdClient) ON vehicule.IdVehicule=facturation.IdVehicule WHERE vehicule.IdLoueur=:idL AND DateF > CURDATE() ORDER BY facturation.DateD";
```

La requête...

... le résultat

VOS LOCATIONS EN COURS

Renault Kadjar



Caractéristiques : {"Moteur": "Essence", "Vitesse": "Automatique", "Nombre de places": "5", "Nombre de portes": "5"}
Locataire : Armanthe
Loué du 2020-11-01 au 2020-11-30

Hyundai i20



Caractéristiques : {"Moteur": "Essence", "Vitesse": "Mécanique", "Nombre de places": "5", "Nombre de portes": "5"}
Locataire : Richard Cooper
Loué du 2020-11-01 au 2020-12-01

Ici, le nom du locataire et la date de location sont mentionnées : le loueur peut ainsi **voir l'intégralité de la flotte** qu'il a louée au client. Les véhicules sont classés par **date de début de location croissante**, pour permettre une meilleure lisibilité des données.

e. Affichage des factures

Location de voitures
Accueil Vos voitures Ajouter un véhicule Vos locations en cours Vos factures
Se déconnecter

Paris Locations

NOUS PARIONS SUR VOTRE BONHEUR

Vos factures :

Date	Client	Véhicule	Statut	Montant
2020-10-20	Armanthe	Mercedes Citan	Payée	800 €
2020-10-25	Richard Cooper	Dacia Duster	Impayée	800 €
2020-11-04	Richard Cooper	Mercedes Citan	Payée	500 €

Enfin, la dernière fonctionnalité développée pour les loueurs est l'**affichage des factures**. J'ai choisi de lister celles-ci dans un tableau responsive, par ordre chronologique et avec les montants en bout de ligne car c'est ce qui intéresse le loueur.

Cet affichage repose sur l'utilisation d'une requête similaire à celle de l'affichage des locations en cours, excepté la condition finale : **les factures concernent des locations terminées**.

III – ASPECT VISUEL

L'aspect visuel de mon application est géré par **Bootstrap**, qui permet de rendre mon site web entièrement **responsive**.

Les parties suivantes décrivent plus en détail des fonctionnalités visuelles intéressantes développées grâce à Bootstrap et **jQuery**.

A – Un menu hamburger

Bootstrap m'a permis de créer un **menu responsive**, aussi appelé un « menu hamburger » : lorsque l'utilisateur réduit sa fenêtre, le menu disparaît derrière un petit bouton. Il suffit de cliquer sur celui-ci pour faire apparaître toutes les fonctionnalités. C'est une fonctionnalité qui nécessite l'utilisation de **jQuery**.



Menu responsive pour les loueurs

Ce menu est également un **menu fixé en haut** du navigateur. Ainsi, l'utilisateur n'aura pas à scroller jusqu'au début de la page pour pouvoir naviguer à travers les différents services que propose le site web.

B – Des tableaux responsive

Grâce à la classe **tab-responsive** de Bootstrap, mon application comporte des tableaux dont le contenu s'adapte à la taille de l'écran. En effet, lorsque l'écran devient trop petit pour afficher convenablement les données des tableaux de facture, une **barre de défilement horizontale** apparaît et permet de se déplacer horizontalement dans le tableau pour avoir accès à tous les champs. Cela évite d'avoir un tableau qui déborde sur d'autres divisions de la page ou illisible.

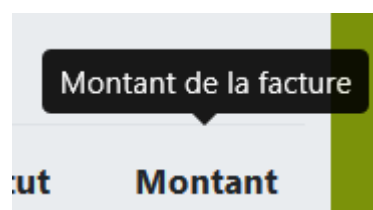
Location de voitures				
Vos factures :				
Date	Client	Véhicule	Statut	Montant
2020-10-20	Armanthe	Mercedes Citan	Payée	800
2020-10-25	Richard Cooper	Dacia Duster	Impayée	800
2020-11-04	Richard Cooper	Mercedes Citan	Payée	500

La colonne du montant ne passe pas à l'écran : une barre de défilement horizontale apparaît pour permettre de naviguer parmi les données

C – Des info-bulles

Grâce à **jQuery** et **Bootstrap**, j'ai également pu créer des **info-bulles**, des messages flottant qui informent l'utilisateur sur le contenu d'un champ.

J'ai choisi de mettre ces info-bulles sur les noms de colonnes du tableau de factures, après m'être rendu compte que certaines appellations pouvaient porter à confusion : ainsi, lorsque l'utilisateur **survole** une des cellules titre, la description du contenu s'affiche.



Une infobulle, au survol de la cellule « Montant »

D – Des card-decks

Les différents services de l'application qui nécessitent l'affichage des véhicules, avec leurs caractéristiques et leur photographie, font appel à la classe **card**, utilisée en complément de la classe **card-deck**.

Ces classes fonctionnent comme un jeu de carte : la classe **card-deck** est le paquet de cartes, et les cartes sont les classes disposées à l'intérieur. Cette classe fonctionne donc avec des éléments **flex-box** : si les cartes disposées au sein du **card-deck** ne tiennent pas sur une ligne, elles sont réparties sur plusieurs lignes. Les **card** sont donc des éléments très intéressants qui facilitent le travail du développeur, et permet de rendre le site web **responsive** plus facilement.

Une carte, composée de deux éléments : le titre (« Kia Picanto ») et le contenu

Kia Picanto

Nombre de véhicules : 60
 Disponible à la location : Disponible
 Caractéristiques : {"Moteur": "Essence", "Vitesse": "Automatique", "Nombre de places": "4", "Nombre de portes": "5"}

IV – CONCLUSION

Ce projet m'aura permis de mettre en place un site web selon l'architecture MVC, très utilisée dans le monde du travail, et d'approfondir mes connaissances en PHP.

Je n'ai malheureusement pas réussi à implémenter tous les services demandés, cependant **tous les services présentés fonctionnent** et disposent d'un aspect visuel satisfaisant, **entièrement responsive** grâce à Bootstrap et jQuery. De plus, mon application est **entièrement structurée en MVC**, une structuration que je trouvais difficile à mettre en place au départ mais qui s'est avérée être très pratique et m'a fait gagner un temps précieux. En effet, devoir passer par `index.php` au lancement de chaque service permet de gérer les problèmes de redirection et simplifie l'accès aux différents services. Enfin, mon application gère les **caractéristiques des véhicules en json**, un des points importants de ce projet. Aucun aspect n'a donc été négligé dans le travail réalisé.

J'ai utilisé toutes les ressources à ma disposition pour résoudre les différents problèmes que je rencontrais, notamment les problèmes de redirection et de passage des paramètres, et ai ainsi gagné en autonomie. Je ressors ainsi satisfaite de mon travail, et aimerai réaliser plus de projets de ce genre à l'avenir.