2018-2019

DOCUMENTATION TECHNIQUE LOCAMOTO

CFPT

v.1.0

LUDOVIC JACOT-DIt-MONTDANDON

Table des matières

[INTRODUCTION 3](#_Toc8809004)

[RAPPEL DU CAHIER DES CHARGES 4](#_Toc8809005)

[OBJECTIFS DU PROJET 4](#_Toc8809006)

[MATERIEL ET LOGICIEL A DISPOSITION 4](#_Toc8809007)

[PLANNIFICATION 4](#_Toc8809008)

[LIVRABLES 4](#_Toc8809009)

[ANALYSE FONCTIONNELLE 5](#_Toc8809010)

[FONCTIONNALITES 5](#_Toc8809011)

[S’INSCRIRE SUR LE SITE 5](#_Toc8809012)

[SE CONNECTER A L’APPLICATION 5](#_Toc8809013)

[MODIFIER SON PROFIL 5](#_Toc8809014)

[LOUER UNE MOTO 5](#_Toc8809015)

[VALIDER UNE LOCATION 5](#_Toc8809016)

[INTERFACES 6](#_Toc8809017)

[BARRE DE NAVIGATION 6](#_Toc8809018)

[PAGE D’ACCUEIL 7](#_Toc8809019)

[PAGE DE LOCATION 9](#_Toc8809020)

[PAGE DE GESTION DES UTILISATEURS 10](#_Toc8809021)

[PAGE DE GESTION DU PARKING 11](#_Toc8809022)

[PAGE DE GESTION DES LOCATIONS 12](#_Toc8809023)

[FORMULAIRE DE SAUVEGARDE D’UTILISATEURS 13](#_Toc8809024)

[FORMULAIRE DE SAUVEGARDE DE MOTOS 14](#_Toc8809025)

[LOGIN 15](#_Toc8809026)

[CONFIRMATION DE SUPPRESSION 15](#_Toc8809027)

[CAS D’UTILISATION 16](#_Toc8809028)

[VISITEUR ANONYME 16](#_Toc8809029)

[UTILISATEUR AUTHENTIFIE 17](#_Toc8809030)

[ADMINISTRATEUR 18](#_Toc8809031)

[ANALYSE ORGANIQUE 20](#_Toc8809032)

[BASE DE DONNEES 20](#_Toc8809033)

[DIAGRAMME DE BASE DE DONNEES 20](#_Toc8809034)

[DETAILS DES TABLES 21](#_Toc8809035)

[ARBORESCENCE DE FICHIER 22](#_Toc8809036)

[CALENDRIER 22](#_Toc8809037)

[DESCRIPTION DES FONCTIONS 22](#_Toc8809038)

[FICHIER « dbconnectino.php » 22](#_Toc8809039)

[FICHIER « users.php » 22](#_Toc8809040)

[FICHIER « motos.php » 23](#_Toc8809041)

[FICHIER « images.php » 24](#_Toc8809042)

[FICHIER « locations.php » 25](#_Toc8809043)

[TESTS ET PROTOCOLE DE TESTS 27](#_Toc8809044)

[PLAN DE TESTS 27](#_Toc8809045)

[RAPPORT DE TESTS 29](#_Toc8809046)

[CONCLUSION 30](#_Toc8809047)

[AMELIORATIONS POSSIBLES 30](#_Toc8809048)

[BILAN PERSONNEL 30](#_Toc8809049)

[BIBLIOGRAPHIE 30](#_Toc8809050)

[CODE REPRIS 30](#_Toc8809051)

[SITE UTILISES 30](#_Toc8809052)

[PLANNING 30](#_Toc8809053)

[PLANNING PREVISIONNEL 30](#_Toc8809054)

[PLANNING EFFECTIF 30](#_Toc8809055)

[TABLE DES ILLUSTRATIONS 30](#_Toc8809056)

[FIGURES 30](#_Toc8809057)

[TABLEAU 30](#_Toc8809058)

[ANNEXES 30](#_Toc8809059)

# INTRODUCTION

Cette documentation détaille les étapes réalisées pour le site web intitulé « Locamoto », réalisé dans le cadre du Travail Pratique Individuel nécessaire à l’obtention du CFC d’informaticien.

Le projet « Locamoto » a pour but de créer un site web permettant aux utilisateurs de consulter des motos avec des variantes de cylindrée, et ce dans le but de louer différents modèles. Les motos sont de marque spéciales.

Les motos sont dans un parking qui donne la possibilité de modifier leurs caractéristiques ou même les supprimer. De ce fait, il est possible de faire des recherches par marque ou cylindré.

Le visiteur anonyme peut consulter la page d’accueil en recherchant les motos par marques ou cylindrée.

L’utilisateur qui créer un compte sur le site peut, après s’être connecté :

* Modifier ses informations
* Louer une moto par jour
* Donner un avis sur une moto louée

L’administrateur a la possibilité de gérer l’ensemble du site :

* Gérer un utilisateur :

1. Création d’un utilisateur
2. Modification d’un utilisateur
3. Suppression d’un utilisateur

* Gérer le parking :

1. Création d’une moto
2. Modification d’une moto

* Gérer les locations

1. Validation d’une location
2. Gestion d’un calendrier

Le calendrier pour louer une moto est un framework. J’ai choisi un calendrier nommé « Fullcalendar », qui offre une documentation assez simple. Il est disponible gratuitement à cette adresse : <https://fullcalendar.io/>

Le style de ce site est inspiré d’un template Bootstrap pour rendre le site adapté à l’écran de l’utilisateur, que ce soit sur un mobile, sur un ordinateur, sur une tablette, etc. qui se nomme : <https://getbootstrap.com/docs/4.2/examples/blog/>

# RAPPEL DU CAHIER DES CHARGES

## OBJECTIFS DU PROJET

La solution est fonctionnelle. Il est donc possible de s’inscrire et de se connecter en tant qu’utilisateur et ainsi effectuer une location. La validation des locations est gérée et le prix de ces dernières est affiché.

Le code est découpé suivant le paterne Modèle-Vue-Contrôleur et le site est responsif.

Les fonctionnalités de gestion par l’administrateur des motos, des locations et du calendrier sont implémentées.

La recherche sur la page d’accueil est implémentée.

L’utilisateur peut donner des avis sur la moto précédemment louée.

## MATERIEL ET LOGICIEL A DISPOSITION

* Un Pc standard école avec 2 écrans
* Windows 10
* EasyPHP ou autre WAMP
* Visual code
* Suite Office

## PLANNIFICATION

* Analyse : 12h
* Implémentation : 36h
* Test : 16h
* Documentation : 24h

## LIVRABLES

* Une planification initiale
* Un exemplaire format PDF de son mémoire avec annexe

1. Documentation technique format PDF
2. Code source

* Un accès aux sources du projet ainsi qu’une version exécutable
* Résumé du travail (page A4 pdf)

# ANALYSE FONCTIONNELLE

Cette partie de la documentation technique explique ce que le projet doit faire au respectant le cahier des charges. Vu qu’il y a deux type de comptes (utilisateur classique et administrateur), les fonctionnalités sont

## FONCTIONNALITES

### S’INSCRIRE SUR LE SITE

Cette fonctionnalité permet à un visiteur anonyme de pouvoir créer un compte en y rentrant son nom, son prénom, un pseudo, un email, un mot de passe et une date de naissance.

### SE CONNECTER A L’APPLICATION

Cette fonctionnalité permet à un utilisateur qui possède un compte de pouvoir se connecter afin de pouvoir effectuer une réservation.

### MODIFIER SON PROFIL

Sur le site, quand on possède un compte, il est possible de consulter son profil et d’en changer les informations.

### LOUER UNE MOTO

Cette fonctionnalité offre la possibilité à un utilisateur authentifié de pouvoir louer une moto. Il peut consulter un calendrier et entrer la date du début de sa location souhaitée ainsi que la date de fin. Le prix pour cette location est affiché en conséquence du nombre de jour choisi.

### VALIDER UNE LOCATION

Cette fonctionnalité permet à l’administrateur d’accepter ou refuser une réservation quémandée par un utilisateur.

## INTERFACES

### BARRE DE NAVIGATION

#### BARRE DE NAVIGATION D’UN VISITEUR ANONYME



Figure 1: BARRE DE NAVIGATION D’UN VISITEUR ANONYME

Le lien « Accueil » est la seule page disponible pour le visiteur (voir Figure 4 : Page d’accueil en tant que visiteur).

Le lien « Connexion » permet au visiteur de s’identifier ou de créer un compte (voir ).

#### BARRE DE NAVIGATION D’UN UTILISATEUR AUTHENTIFIE



Figure 2: BARRE DE NAVIGATION D’UN UTILISATEUR AUTHENTIFIE

Le lien « Accueil » permet à l’utilisateur d’aller à la page principale (voir Figure 5 : Page d’accueil en tant qu’utilisateur connecté).

Le lien « Profil » permet à l’utilisateur de consulter son profil et d’en modifié les informations (voir Figure 10 : Formulaire de sauvegarde d’utilisateurs).

Le lien « Déconnexion » permet à l’utilisateur de se déconnecté et ainsi passer à simple visiteur.

#### BARRE DE NAVIGATION D’UN ADMINISTRATEUR



Figure 3: BARRE DE NAVIGATION D’UN ADMINISTRATEUR

Le lien « Utilisateurs » permet à l’administrateur d’accéder à la page de gestion des utilisateurs (voir Figure 7 : Page de gestion des utilisateurs).

Le lien « Parking » permet à l’administrateur d’accéder à la page de gestion des motos (voir Figure 8 : Page de gestion du parking).

Le lien « Locations » permet à l’administrateur d’accéder à la page de gestion des réservations (voir Figure 9 : Page de gestion des locations).

### PAGE D’ACCUEIL

#### PAGE D’ACCUEIL EN TANT QUE VISITEUR

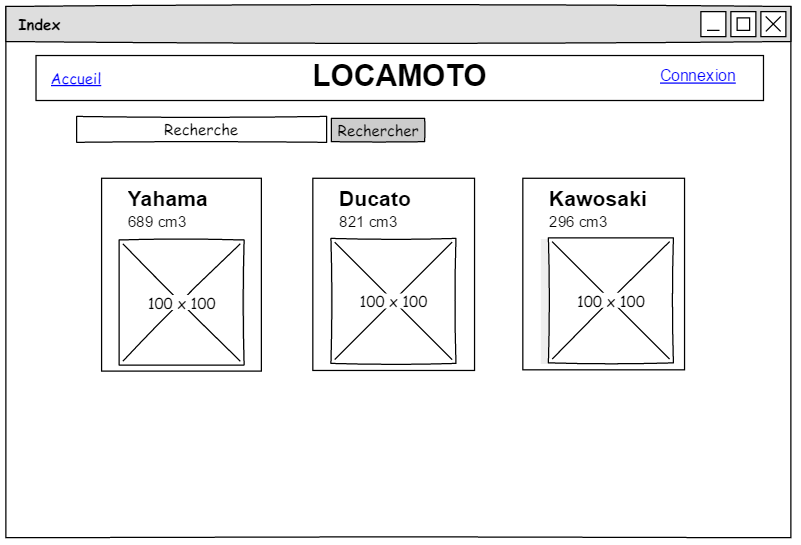


Figure 4: PAGE D’ACCUEIL EN TANT QUE VISITEUR

La page d’accueil en tant que visiteur affiche toutes les motos enregistrées sur le site avec leurs marques, leurs cylindrées et leurs photos.

La barre de recherche invite à écrire les mots-clés que l’on souhaite trouver. Une fois le bouton rechercher pressé, seul les noms de marque ou cylindré correspondant à la recherche sont affichés.

#### PAGE D’ACCUEIL EN TANT QU’UTILISATEUR CONNECTE

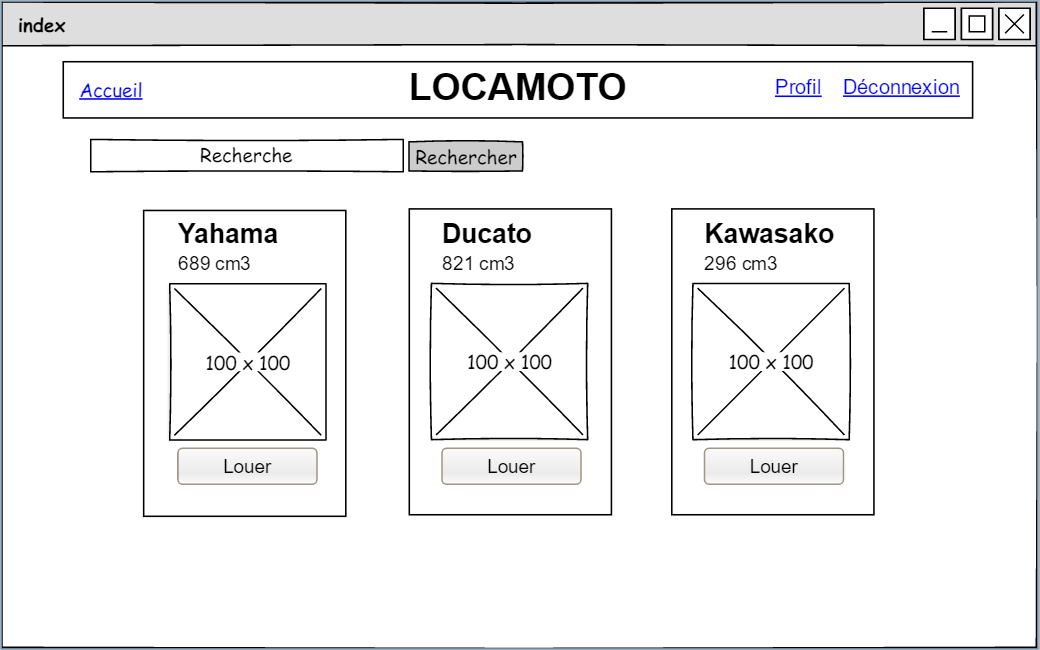


Figure 5 : PAGE D’ACCUEIL EN TANT QU’UTILISATEUR CONNECTE

La page d’accueil en tant que visiteur affiche toutes les motos enregistrées sur le site avec leurs marques, leurs cylindrées et leurs photos.

La barre de recherche invite à écrire les mots-clés que l’on souhaite trouver. Une fois le bouton rechercher pressé, seul les noms de marque ou cylindré correspondant à la recherche sont affichés.

Le bouton « Louer » envoie sur la page de réservation (voir Figure 6 : Page de location).

### PAGE DE LOCATION

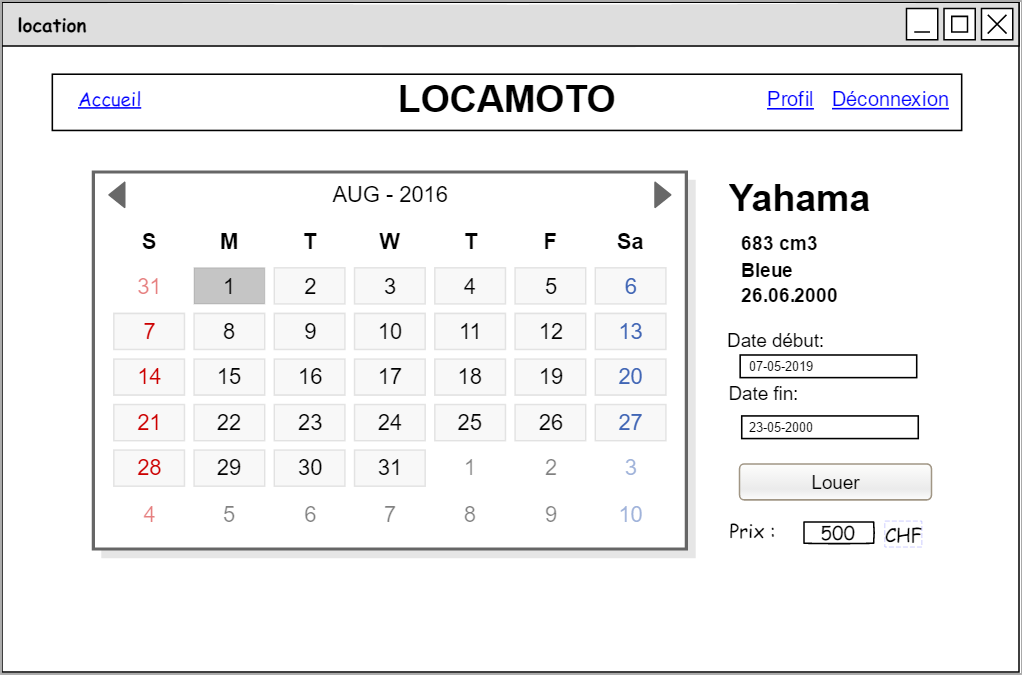


Figure 6: PAGE DE LOCATION

La page de location affiche toute les informations sur la moto voulue (marque, cylindrée, couleur, date d’immatriculation) sur la droite.

Le prix évolue en fonction du nombre de jour loué.

Le bouton louer réserve les dates indiquées avec cette moto et il devra être validé par l’administrateur (voir Figure 9 : Page de gestion des locations).

### PAGE DE GESTION DES UTILISATEURS

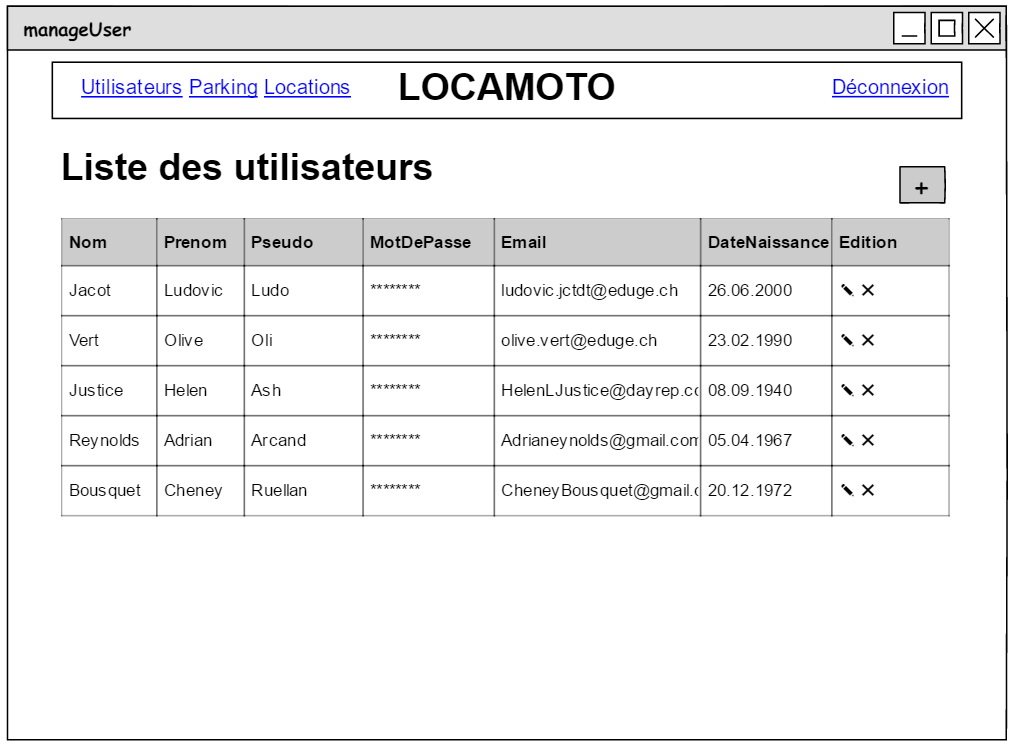


Figure 7: PAGE DE GESTIONS DES UTILISATEURS

La page de gestion des utilisateurs est accessible uniquement par l’administrateur. Elle affiche un tableau avec tous les utilisateurs inscrits sur le site.

Le bouton plus « + » permet de créer un nouvel utilisateurs (voir Figure 10 : Formulaire de sauvegarde d‘utilisateurs).

L’icône du crayon « ✎ » permet d’éditer l’utilisateur correspondant à l’aide du formulaire prévu à cet effet (voir Figure 10 : Formulaire de sauvegarde d‘utilisateurs).

L’icône de la croix « ✖ » permet de supprimer l’utilisateur correspondant après une confirmation (voir Figure 13 : Confirmation de suppression).

### PAGE DE GESTION DU PARKING



Figure 8: PAGE DE GESTION DU PARKING

La page de gestion du parking est accessible uniquement par l’administrateur. Elle affiche un tableau de toutes les motos disponibles à la location dans le parking.

Le bouton plus « + » permet de créer une nouvelle moto (voir Figure 11 : Formulaire de sauvegarde de motos).

L’icône du crayon « ✎ » permet d’éditer la moto correspondante à l’aide du formulaire prévu à cet effet (voir Figure 11 : Formulaire de sauvegarde de motos).

### PAGE DE GESTION DES LOCATIONS



Figure 9: PAGE DE GESTION DES LOCATIONS

La page gestion des locations est accessible uniquement par l’administrateur. Elle affiche un tableau de toutes les locations en attente d’être validée.

L’icône du vu « ✔ » existe pour confirmer la réservation correspondante et ainsi, un email est envoyé à l’utilisateur.

L’icône de la croix « ✖ » refuse la location et supprime celle-ci après une validation (voir Figure 13 : Confirmation de suppression).

### FORMULAIRE DE SAUVEGARDE D’UTILISATEURS

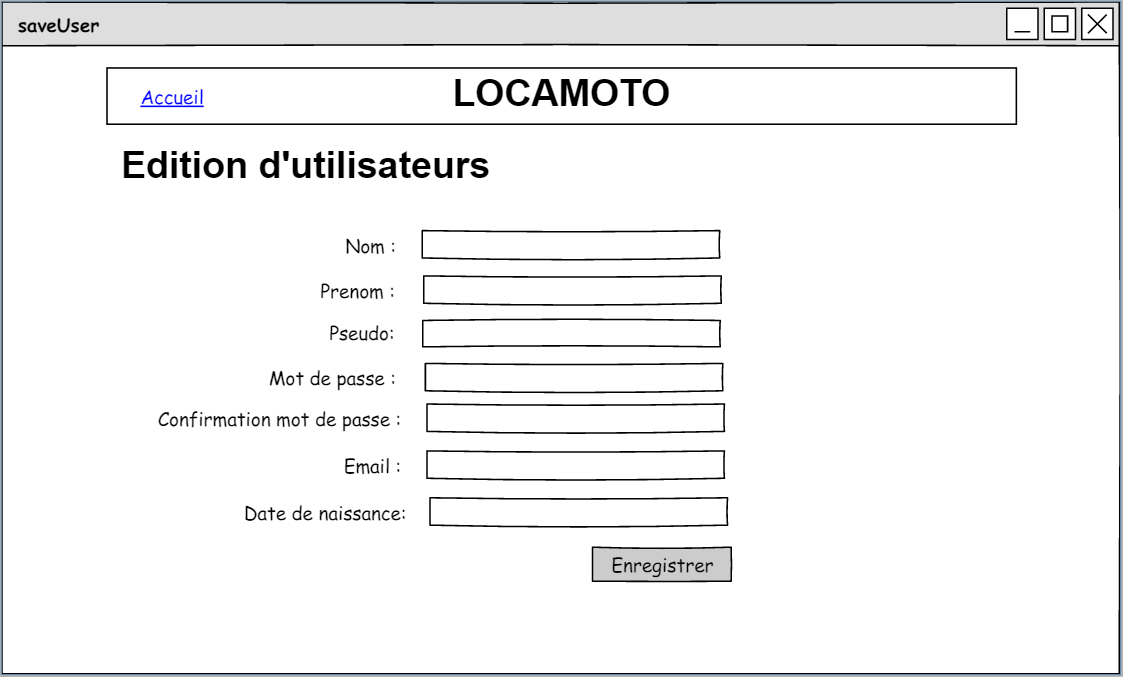


Figure 10: FORMULAIRE DE SAUVEGARDE D’UTILISATEURS

Le formulaire indique sept champs obligatoires « Nom », « Prénom », « Pseudo », « Mot de passe », Confirmation de mot de passe », « Email » et « Date de Naissance ». Il utilise une méthode « POST ».

Le nom, le prénom, le pseudo sont des inputs de type « text ».

Le mot de passe et la confirmation de mot de passe sont des inputs de type « password ».

L’email est un input de type « email ».

La date est

Le bouton enregistrer est de type « submit ».

Il est utilisé pour l’inscription ainsi que pour l’édition d’un utilisateur. En cas d’édition, les informations sont automatiquement remises dans les champs correspondants.

### FORMULAIRE DE SAUVEGARDE DE MOTOS

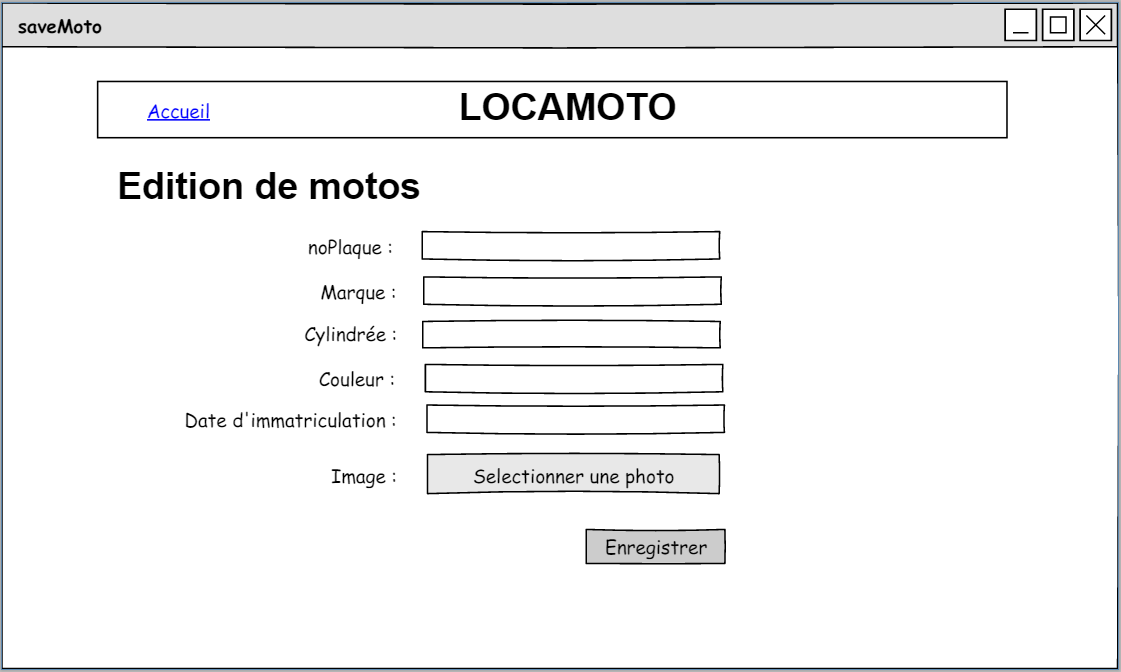


Figure 11: FORMULAIRE DE SAUVEGARDE DE MOTOS

Le formulaire est composé de 6 champs « noPlaque », « Marque », « Cylindrée », « Couleur », « Date d’immatriculation » et « Image ». Il utilise une méthode « POST ».

Le numéro de plaque est un input de type « number »

La marque, la cylindrée et la couleur sont des inputs de type « text ».

La date d’immatriculation est

L’image est un input de type « file ».

Le bouton enregistrer est de type « submit ». Il ajoute ou met à jour une moto dans la base de données. Si on édite une moto existante, les informations sont automatiquement insérées dans les champs.

### LOGIN



Figure 12: LOGIN

Le formulaire utilise une méthode « POST ». Le login comporte deux champs « Pseudo » et « Mot de passe ».

Le bouton connexion est de type « submit »

### CONFIRMATION DE SUPPRESSION

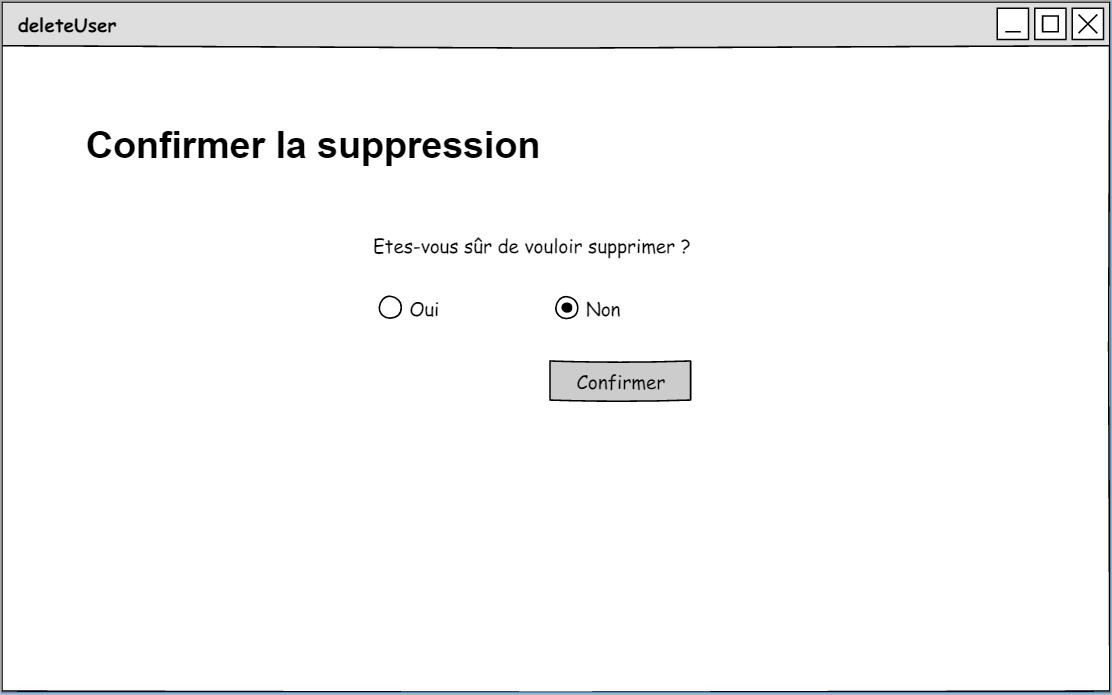


Figure 13: CONFIRMATION DE SUPPRESSION

Le formulaire est de type « POST ».

Le bouton confirmer est type « submit ». Il supprime définitivement l’utilisateur ou la location.

## CAS D’UTILISATION

### VISITEUR ANONYME



**Nom UC :** Se connecter

**Acteur principal :** Visiteur anonyme

**Déclencheur :** L’utilisateur accède au site

**Flot principal :**

1. Le système affiche la page d’accueil.
2. L’utilisateur clique sur « Connexion ».
3. Le système redirige l’utilisateur sur le formulaire de connexion qui contient deux champs : « Pseudo » et « Mot de passe ».
4. L’utilisateur entre les informations demandées et clique sur « Connexion ».
5. Le système affiche la page principale.

**Flot alternatif :**

4a. L’utilisateur ne possède pas de compte et clique sur « inscription ».

4a1. Le système redirige l’utilisateur sur le formulaire d’édition d’utilisateur qui contient sept champs : « Nom », « Prénom », « Pseudo », « Mot de passe », « Confirmation de mot de passe », « Email » et « Date de naissance ».

4a2. L’utilisateur renseigne tous les champs et clique sur « Enregistrer ».

4a3. Le système affiche la page d’accueil.

**Nom UC :** Rechercher par moto ou cylindrée

**Acteur principal :** Visiteur anonyme et utilisateur connecté

**Déclencheur :** L’utilisateur accède au site

**Flot principal :**

1. Le système affiche la page d’accueil.
2. L’utilisateur entre les mots-clés qu’il souhaite atteindre et clique sur rechercher.
3. La page affiche uniquement les résultats qui correspondent.

### UTILISATEUR AUTHENTIFIE



**Nom UC :** Modifier son profil

**Acteur principal :** Utilisateur inscrit

**Déclencheur :** L’utilisateur clique sur « Profil »

**Flot principal :**

1. L’utilisateur clique sur « profil ».
2. Le système redirige l’utilisateur sur le formulaire d’édition d’utilisateur qui contient sept champs : « Nom », « Prénom », « Pseudo », « Mot de passe », « Confirmation de mot de passe », « Email » et « Date de naissance ». Ils sont tous rempli avec les informations correctes.
3. L’utilisateur édite les informations qu’il souhaite et clique sur « enregistrer ».
4. Le système affiche la page d’accueil.

**Nom UC :** Réserver une moto

**Acteur principal :** Utilisateur inscrit

**Déclencheur :** L’utilisateur consulte les motos.

**Flot principal :**

1. Le système affiche la page d’accueil avec les différentes motos.
2. L’utilisateur clique sur « louer ».
3. Le système redirige l’utilisateur sur la page de location. Elle affiche un calendrier.
4. L’utilisateur renseigne la date du début de la location ainsi que la fin de location et clique sur « Louer ».
5. Le système affiche la page d’accueil.

**Nom UC :** Donner un avis

**Acteur principal :** Utilisateur inscrit

**Déclencheur :** L’utilisateur clique sur « Laisser un avis »

**Flot principal :**

1. Le système affiche la page d’accueil avec les différentes motos.
2. L’utilisateur clique sur « Laisser un avis »
3. Le système affiche une page avec une zone de texte et toutes les réservations effectuées par l’utilisateur sur cette moto.
4. L’utilisateur entre un message, sélectionne la location qui correspond et clique sur « Envoyer »
5. Le système redirige sur la page d’accueil.

**Flot secondaire :**

3a. L’utilisateur n’a jamais fait de réservation sur cette moto

3a1. Le système affiche un message d’erreur.

### ADMINISTRATEUR



**Nom UC :** Gérer les utilisateurs

**Acteur principal :** Administrateur

**Déclencheur :** L’administrateur clique sur « Utilisateurs »

**Flot principal :**

1. L’administrateur clique sur « Utilisateurs »
2. Le système affiche un tableau avec tous les utilisateurs inscrits sur le site.
3. L’administrateur choisi un profil et clique sur éditer.
4. Le système redirige l’administrateur sur le formulaire d’édition d’utilisateur qui contient sept champs : « Nom », « Prénom », « Pseudo », « Mot de passe », « Confirmation de mot de passe », « Email » et « Date de naissance ». Ils sont tous rempli avec les informations correctes.
5. L’administrateur édite les informations qu’il souhaite et clique sur « enregistrer ».
6. Le système affiche la page principale.

**Flot alternatif :**

3a. L’administrateur choisi un profil et clique sur supprimer.

3a1. Le système affiche un formulaire de confirmation.

3a2. L’utilisateur clique sur « Confirmer ».

3a3. Le système supprime l’utilisateur et affiche la page d’accueil.

**Nom UC :** Gérer les motos

**Acteur principal :** Administrateur

**Déclencheur :** L’administrateur clique sur « Parking »

**Flot principal :**

1. L’administrateur clique sur « Parking »
2. Le système affiche un tableau avec toutes les motos inscrites sur le site.
3. L’administrateur clique sur le bouton plus « + ».
4. Le système redirige l’administrateur sur le formulaire d’édition de moto. Les champs à remplir sont : « noPlaque », « Marque », « Cylindrée », « Couleur » et « Date d’immatriculation ».
5. L’administrateur entre les champs stipulés et clique sur enregistrer
6. Le système affiche la page précédente.

**Flot alternatif :**

3a. L’administrateur choisi une moto et clique sur éditer.

3a1. Le système redirige l’administrateur sur le formulaire d’édition de moto. Les champs à remplir sont : « noPlaque », « Marque », « Cylindrée », « Couleur » et « Date d’immatriculation ».

3a2. L’administrateur édite les champs souhaités et clique sur enregistrer.

3a3. Le système affiche la page précédente.

3b. L’administrateur choisi une moto et clique sur supprimer.

3b1. Le système affiche un formulaire de confirmation.

3b2. L’utilisateur clique sur « Confirmer ».

3b3. Le système supprime l’utilisateur et affiche la page précédente.

**Nom UC :** Gérer les locations

**Acteur principal :** Administrateur

**Déclencheur :** L’administrateur clique sur « Locations »

**Flot principal :**

1. L’administrateur clique sur « Locations »
2. Le système affiche un tableau avec toutes les réservations demandées par les utilisateurs.
3. L’administrateur choisi de validé une réservation en cliquant sur le vu « ✔ ».
4. Le système la valide, envoie un mail à l’utilisateur correspondant et recharge la page.

**Flot alternatif :**

3a. L’administrateur choisi de refuser une réservation en cliquant sur la croix « ✖ ».

3a1. Le système affiche un formulaire de confirmation.

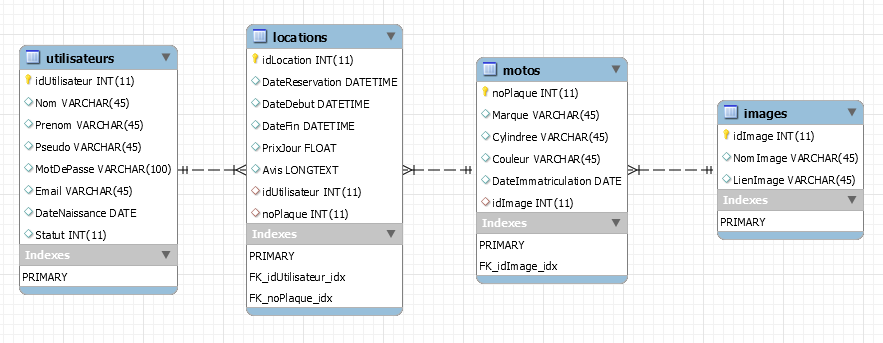
3a2. L’utilisateur clique sur « Confirmer ».

3a3. Le système supprime l’utilisateur et affiche la page précédente.

# ANALYSE ORGANIQUE

## BASE DE DONNEES

### DIAGRAMME DE BASE DE DONNEES



Le site comporte quatre tables dans la base de données.

La table utilisateurs contient tous les champs soumis par l’utilisateur ainsi que son identifiant et son statut. « idUtilisateurs » est donc la clé primaire.

La table locations contient la date à laquelle elle a été faite ainsi que les dates auxquelles la réservation est effectuée. Les avis y sont aussi stockés avec le prix. « idLocation » est la clef primaire. La table possède aussi deux clefs étrangères, l’identifiant de l’utilisateur qui l’a effectué et le numéro de plaque de la moto réservée.

La table motos contient toute les informations relatives à une moto. Le numéro de plaque « noPlaque » est la clef primaire. Elle possède aussi une clef étrangère, l’identifiant de l’image qui lui est relié.

La table image possède une clef primaire qui est l’identifiant de cette dernière. Le nom et le lien de l’image y sont stocké.

### DETAILS DES TABLES

|  |  |
| --- | --- |
| **Abréviations utilisées** | **Définition** |
| PK | Clefs primaire |
| NN | Non nul (Impossible d’être vide – NULL) |
| UQ | Unique (Impossible d’avoir deux fois la même valeur pour cette occurrence) |
| AI | Auto incrémentation |

#### TABLE « UTILISATEURS »

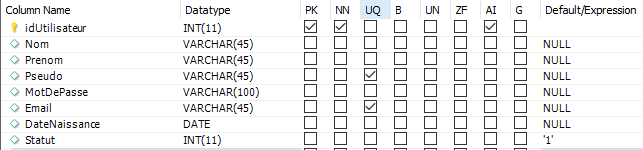


Figure 14: TABLE UTILISATEURS

#### TABLE LOCATIONS

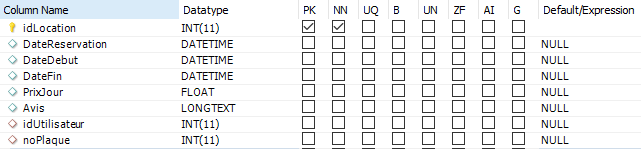


Figure 15: TABLE LOCATIONS

#### TABLE MOTOS

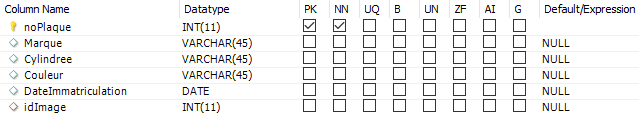


Figure 16: TABLE MOTOS

#### TABLE IMAGES

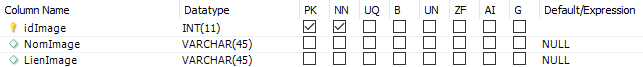


Figure 17: TABLE IMAGES

## ARBORESCENCE DE FICHIER

IMAGE

* Le dossier « config » contient un fichier avec des constantes utiles à la connexion à la base de données et un fichier avec des constantes utiles à l’envoie de mail.
* Le dossier « fullcalendar-4.1.0 » contient les fichiers du plugin FullCalendar
* Le dossier « css » contient les fichiers bootstrap ainsi qu’un fichier style.css
* Le dossier « img » contient toutes images visibles sur le site.
* Le dossier « model » contient tous les fichiers de CRUD exercés sur les tables de la base de données ainsi qu’un fichier de connexion à la base de données.
* Le dossier test contient les
* Le dossier « view » contient les pages du site.
* A la racine il y a tous les contrôleurs

## DESCRIPTION DES FONCTIONS

### FICHIER « dbconnection.php »

#### Fonction « connectDB »

Cette fonction est déclarée dans le fichier « dbconnection.php » et sert à établir la connexion à la base de données. Elle ne reçoit pas de paramètre mais utilise cinq constantes déclarée dans le fichier « config/confparam.php ». Ces constantes sont « DB\_DBTYPE » pour le type de base de données, « DB\_HOST » pour l’hôte du serveur, « DB\_DBNAME » pour le nom de la base de données, « DB\_USER » pour l’utilisateur qui se connecte et « DB\_PASS » pour le mot de passe utilisé lors de la connexion. Cette fonction retourne une instance de l’objet PDO.

### FICHIER « users.php »

#### Fonction « userExist »

Cette fonction permet de savoir si un couple pseudo de l’utilisateur / mot de passe est correct et existe dans la base. Elle est utilisée lors de la connexion d’utilisateur (classique ou administrateur). Elle reçoit deux paramètre, le pseudo de l’utilisateur ainsi que le mot de passe, qu’elle utilise pour construire une requête SQL (un SELECT avec le nom et le mot de passe dans la clause WHERE) qui lui permet de savoir si le couple pseudo d’un utilisateur / mot de passe est correct, et donc si le login est correct. Effectivement, si le SQL retourne une ligne, c’est que les données correspondent. Elle retourne un tableau contenant tous les champs de la table coïncidant avec ces paramètres (des données qui pourront être réutilisées sur le site) si le login est juste, sinon « False ».

#### Fonction « getAllUsers »

Cette fonction permet de récupérer tous les utilisateurs inscrit dans le base de données. Elle est utilisée pour la page « manageuser.php ». Elle ne reçoit aucun paramètre, vu qu’il s’agit simplement d’un SELECT général effectué sur la table « utilisateurs ». Elle retourne un tableau à deux dimensions, dont chaque ligne contient l’identifiant de l’utilisateur, son nom, son prénom, son mot de passe encrypté, son email, sa date de naissance.

#### Fonction « getUserById »

Cette fonction permet de récupérer les informations sur un utilisateur d’après son identifiant. Elle est utilisée sur la page « saveuser.php », lorsqu’il faut modifier un utilisateur. Elle est donc très similaire à la fonction précédente (Fonction « getAllUsers »), sauf que celle-ci reçoit en paramètre l’identifiant. Celui-ci est utilisé dans la clause WHERE de la requête SQL. Elle retourne un tableau qui contient l’identifiant de l’utilisateur, son nom, son prénom, son mot de passe encrypté, son email, sa date de naissance.

#### Fonction « getUserId »

Cette fonction, en cas de succès, retourne l’identifiant d’un utilisateur d’après son pseudo. Elle est utilisée dans le contrôleur « login.php » pour mettre en session l’identifiant de la personne qui vient de se connecter. Elle reçoit en paramètre le pseudo qui vient de se connecter qui est utilisé dans la clause WHERE de la requête SQL. Elle retourne l’identifiant de l’utilisateur.

#### Fonction « getUserStatut »

Cette fonction, en cas de succès, retourne le statut d’un utilisateur d’après son pseudo. Elle est utilisée dans le contrôleur « login.php » pour mettre en session le statut de la personne qui vient de se connecter. Elle reçoit en paramètre le pseudo qui vient de se connecter qui est utilisé dans la clause WHERE de la requête SQL. Elle retourne le statut de l’utilisateur.

#### Fonction « addUser »

Cette fonction permet d’ajouter un nouvel utilisateur à la base de données. Elle est utilisée dans le contrôleur « saveuser.php » pour entrer toutes les informations du formulaire dans la base de données. Elle reçoit en paramètre chaque champ nécessaire pour un utilisateur : nom, prénom, pseudo, mot de passe, email, date de naissance. Ils sont tous utilisé comme valeur à insérer. Elle retourne l’identifiant de l’utilisateur à l’instant ajouté ou « False » en cas de problème.

#### Fonction « updateUser »

Cette fonction permet de mettre à jour un utilisateur d’ores et déjà présent dans la base de données. Elle est utilisée dans le contrôleur « saveuser.php » pour entrer toutes les nouvelles informations du formulaire dans la base de données. Elle reçoit en paramètre chaque champ possible pour un utilisateur : nom, prénom, pseudo, email, date de naissance. Ils sont tous utilisé comme valeur à insérer. Elle retourne « True » en cas de succès ou « False » en cas de problème.

#### Fonction « deleteUser »

Cette fonction permet de supprimer définitivement un utilisateur de la base de données. Elle est utilisée sur la page « deleteuser.php » où, après confirmation, il est justement possible pour l’administrateur de supprimer un utilisateur. Elle reçoit en paramètre l’identifiant de l’utilisateur à supprimer qui est utilisé dans la clause WHERE du DELETE. Elle retourne « True » en cas de succès ou « False » en cas de problème.

### FICHIER « motos.php »

#### Fonction « getMotos »

Cette fonction permet de récupérer toutes les motos présentent dans le base de données. Elle est utilisée pour la page « manageparking.php ». Elle ne reçoit aucun paramètre, vu qu’il s’agit simplement d’un SELECT général effectué sur la table « motos ». Elle retourne un tableau à deux dimensions, dont chaque ligne contient le numéro de plaque d’une moto, sa marque, sa cylindrée, sa couleur, sa date d’immatriculation.

#### Fonction « getMotosImages »

Cette fonction permet de récupérer toutes les motos présentent dans le base de données avec leurs images assignées. Elle est utilisée pour la page « index.php ». Elle est donc très similaire à la fonction précédente (Fonction « getMotos ») mais fait une jointure entre la table « motos » et la table « images ». Elle ne reçoit aucun paramètre, vu qu’il s’agit simplement d’un SELECT général effectué sur la table « motos » en plus de la table « images ». Elle retourne un tableau à deux dimensions, dont chaque ligne contient le numéro de plaque d’une moto, sa marque, sa cylindrée, sa couleur, sa date d’immatriculation, le nom de son image.

#### Fonction « getMotosSearch »

Cette fonction permet de récupérer toutes les motos présentent dans le base de données avec leurs images assignées correspondant à la recherche demandée. Elle effectue le même SELECT que la fonction précédente (Fonction « getMotosImages ») mais utilise en plus l’opérateur LIKE entre la marque et le paramètre ou la cylindrée et le paramètre. Elle retourne un tableau à deux dimensions, dont chaque ligne contient le numéro de plaque d’une moto, sa marque, sa cylindrée, sa couleur, sa date d’immatriculation, le nom de son image.

#### Fonction « getMotoByPlaque »

Cette fonction permet de récupérer toutes les informations d’une moto par rapport à son numéro de plaque. Elle est utilisée dans le fichier « savemoto.php ». Elle effectue un SELECT de « Marque », « Cylindree », « Couleur », « DateImmatriculation ». Elle reçoit donc le numéro de plaque « noPlaque » qui est la clé primaire en paramètre. Elle retourne un tableau avec chaque champ.

#### Fonction « addMoto »

Cette fonction permet d’ajouter une nouvelle moto à la base de données. Elle est utilisée dans le contrôleur « manageparking.php » pour entrer toutes les informations du formulaire dans la base de données. Elle reçoit en paramètre chaque champ nécessaire pour une moto : noPlaque, marque, cylindrée, couleur, date immatriculation. Ils sont tous utilisé comme valeur à insérer. Elle retourne « True » en cas de succès ou « False » en cas de problème.

#### Fonction « updateMoto »

Cette fonction permet de mettre à jour une moto d’ores et déjà présente dans la base de données. Elle est utilisée dans le contrôleur « manageparking.php » pour entrer toutes les nouvelles informations du formulaire dans la base de données. Elle reçoit en paramètre chaque champ possible pour une moto : noPlaque, marque, cylindrée, couleur, date immatriculation. Ils sont tous utilisé comme valeur à insérer. Elle retourne « True » en cas de succès ou « False » en cas de problème.

### FICHIER « images.php »

#### Fonction « addImage »

Cette fonction permet d’ajouter une image dans la base de données. Elle est utilisée dans le contrôleur « savemoto.php ». Elle reçoit en paramètre le nom de l’image et le lien de l’image. Ils sont utilisés comme valeur à insérer. Elle renvoie le dernier identifiant entré ou « FALSE ».

#### Fonction « updateImage »

Cette fonction permet de mettre à jour des informations concernant une moto existante. Elle est utilisée dans le contrôleur « manageparking.php » pour changer les informations en fonction du formulaire. Elle reçoit en paramètre le nom de l’image et le lien de l’image. Les deux champs sont utilisés comme valeur à mettre à jour. Elle retourne « True » en cas de succès ou « False » en cas d’erreur.

### FICHIER « locations.php »

#### Fonction « getLocationsJSON »

Cette fonction permet de récupérer toutes les locations présentent dans le base de données. Elle est utilisée pour la page « managelocation.php ». Elle ne reçoit aucun paramètre, vu qu’il s’agit simplement d’un SELECT général effectué sur la table « locations ». Elle retourne une transformation d’un tableau à deux dimensions, dont chaque ligne contient la date de réservation, la date de début, de date de début, le prix, l’avis, l’identifiant de l’utilisateur et le numéro de plaque en fichier JSON.

#### Fonction « getLocationsJSONByPlaque »

Cette fonction permet de récupérer toutes les locations présentent dans le base de données. Elle est utilisée pour la page « managelocation.php ». Elle reçoit en paramètre le numéro de plaque, vu qu’il s’agit simplement d’un SELECT général effectué sur la table « locations ». Elle retourne une transformation d’un tableau à deux dimensions, dont chaque ligne contient la date de réservation, la date de début, de date de début, le prix, l’avis, l’identifiant de l’utilisateur et le numéro de plaque en fichier JSON.

#### Fonction « getLocationsUserMoto »

Cette fonction récupère tous les enregistrements de la table locations ainsi que le pseudo de l’utilisateur qui a réservé et la marque de la moto qui viennent de la table utilisateurs et motos. Elle utilisée sur la page « managelocation.php ». Elle ne reçoit pas de paramètre, c’est un SELECT avec une jointure entre l’identifiant de l’utilisateur et une seconde jointure entre le numéro de plaque. Elle retourne un tableau à deux dimensions, dont chaque ligne contient la date de réservation, la date de début, de date de début, le prix, l’avis, l’identifiant de l’utilisateur, le numéro de plaque, le pseudo, l’email et la marque.

#### Fonction « getLocationsByPlaque »

Cette fonction permet de récupérer toutes les locations présentent dans le base de données. Elle est utilisée pour la page « managelocation.php ». Elle reçoit en paramètre le numéro de plaque, vu qu’il s’agit simplement d’un SELECT général effectué sur la table « locations ». Elle retourne un tableau à deux dimensions, dont chaque ligne contient la date de réservation, la date de début, de date de début, le prix, l’avis, l’identifiant de l’utilisateur et le numéro de plaque.

#### Fonction « addLocation »

Cette fonction ajoute un enregistrement dans le table locations. Elle est utilisée avec la page « savelocation.php ». Elle reçoit en paramètre chaque champ nécessaire pour une location : la date de réservation, la date de début, de date de début, le prix, l’avis, l’identifiant de l’utilisateur et le numéro de plaque. Ils sont tous utilisé comme valeur à insérer. Elle retourne « True » en cas de succès ou « False » en cas de problème.

#### Fonction « addAvis »

Cette fonction ajoute un avis sur une réservation dans la table locations. Elle est utilisée dans le fichier « saveadvice.php ». Elle reçoit en paramètre l’identifiant de la location et l’avis à inséré. Ils sont utilisés comme valeur à mettre à jour. Elle renvoie « TRUE » en cas de succès ou « FALSE ».

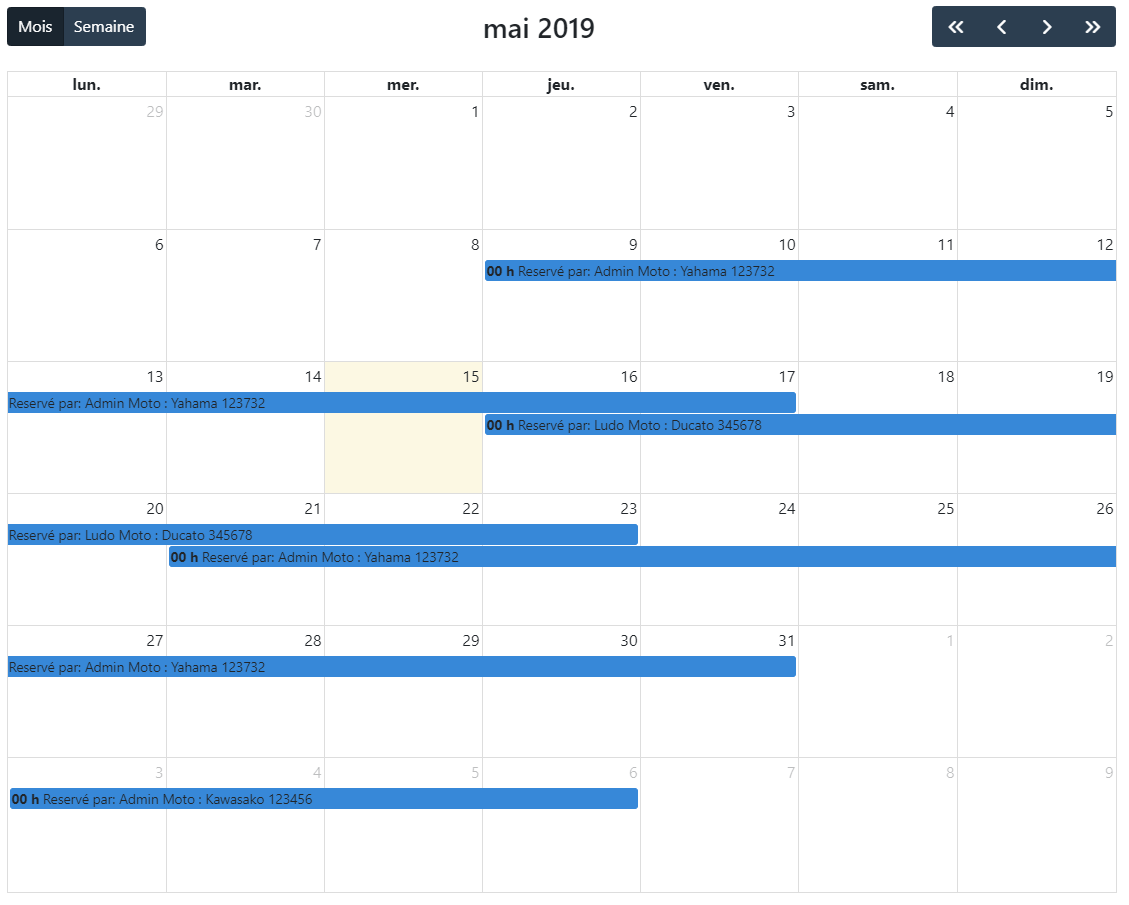
#### Fonction « updateLocation »

Cette fonction permet de mettre à jour des informations concernant une location existante. Elle est utilisée dans le contrôleur « managelocation.php » pour en changer les. Elle reçoit en paramètre le nom de l’image et le lien de l’image. Les dates sont utilisées comme valeur à mettre à jour. Elle retourne « True » en cas de succès ou « False » en cas d’erreur.

#### Fonction « deleteLocation »

Cette fonction permet de supprimer un enregistrement de la table locations. Elle est dans le contrôleur « deletelocation.php » où, après confirmation, il est justement possible pour l’administrateur de supprimer une réservation. Elle reçoit en paramètre l’identifiant de la location à supprimer qui est utilisé dans la clause WHERE du DELETE. Elle retourne « True » en cas de succès ou « False » dans le cas contraire.

## CALENDRIER



Pour générer le calendrier, j’utilise Full Calendar. Cet API créer un calendrier suivant les paramètres que l’on définit.

Pour l’utiliser, il faut placer le dossier fullcalendar-4.1.0 à la racine de l’espace de travail et ajouté différent lien et script dans la balise <head> de notre page. On a parmi ceux-ci des plugins css pour un affichage correct des plugins utilisés, des plugins JS pour le fonctionnement et des plugins pour sélectionner une date dans le calendrier et pour mettre le calendrier en français.

Le calendrier est créé dans une balise <div> comprenant un identifiant « calendar ».

Un script Javascript est codé en dehors du <body> pour que le HTML soit chargé en premier. Il va créer le calendrier dans la balise <div » qui contient l’identifiant « calendar ». L’entête est paramétré avec une vue par mois ou par semaine à gauche, le mois en cours ou la semaine en cours au milieu comme titre et les différents boutons de navigation à droite. La grille par mois est déni par défaut avec la langue en français et la possibilité de sélectionner un jour activée. La fonction « dateClick » place la date du jour sur lequel on vient de cliquer dans le formulaire. La fonction « select » place les dates sélectionnées avec un appui de souris constant dans le formulaire, logiquement le premier jour dans le champs « date de début » et le dernier jour dans le champs « date de fin ». Elle calcule aussi la différence entre le dernier jour et le premier pour actualisé le prix en conséquence. La fonction « selectAllow » empêche de sélectionner des jours déjà passés.

## ENVOIE DE MAIL

Le site utilise Swift Mailer 5 pour envoyer des emails. Swift Mailer est une solution de publipostage à base de composants pour PHP 7.

Une adresse mail a été créée exclusivement pour le site Locamoto : [locamoto.noreply@gmail.com](mailto:locamoto.noreply@gmail.com) avec le mot de passe : Super2018.

Pour utiliser Swift Mailer, le dossier contenant tous ses fichiers « swiftmailer5 » est placé à la racine du site. Un fichier « mailparam.php » contenant toutes les constantes relative à l’envoie de mail, c’est-à-dire, l’adresse mail qui envoie, son mot de passe, le serveur (SMTP : Simple Mail Transfer Protocol) etc.

Lorsque l’admin valide une réservation, le contrôleur « sendmail.php » va s’occuper d’inclure le fichier de configuration et de swiftmailer. Les constantes sont paramétrées et le corps de mail, donc son contenu se décide ici.

L’utilisateur va donc recevoir un mail de la part de [locamoto.noreply@gmail.com](mailto:locamoto.noreply@gmail.com) semblable à « Votre réservation du : 00.00.2000 jusqu’au : 00.00.2000 a été approuvée au prix de : 00.- CHF.

Exemple :



Figure 18: Mail

Nota Bene : Le réseau de l’école ne permet pas d’envoyer des mails, pour implémenter et tester cette fonctionnalité le partage de connexion d’un smartphone a été utilisé.

## TESTS ET PROTOCOLE DE TESTS

### PLAN DE TESTS

#### Tests sur la page « index.php »

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **A tester** | **Scénario** | **Résultat attendu** |
| 01 | Recherche | Inscrire « Ya » dans le barre de recherche et cliquer sur rechercher. | Seules les motos « Yahama » s’affiche sur la page. |
| 02 | Location | Se connecter et cliquer sur « Location ». Sélectionner du 22 mai ou 25 mai et cliquer sur « Louer ». | Notification « Réservation effectuée avec succès. » + La location est enregistrée dans le base de données. |
| 03 | Avis | Se connecter et cliquer sur « Laisser un avis ». Entrez « test » dans la zone de texte, cocher sur la première réservation et cliquer sur « Envoyer ». | Notification « Avis envoyé avec succès » + L’avis est enregistrer dans la base de données. |

#### Tests connexion et inscription

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **A tester** | **Scénario** | **Résultat attendu** |
| 00 | Vérifier que le mot de passe et la confirmation de mot de passe correspondent. | Entrer « Super2018 » dans le champ mot de passe et « Super2019 » dans le champ confirmation de mot de passe. | Notification : « Les mots de passe ne correspondent pas ». |
| 00 | Vérifier que le mot de passe est crypté en sha1 | Entrer toutes les informations demandées :  - Prénom : Test  - Nom : Test  - Pseudo : Test  - Email : [Test@gmail.com](mailto:Test@gmail.com)  - Date d’anniversaire : 01.01.2019  - Mot de passe : Super  Et cliquer sur enregistrer. | Dans la base de données le champs mot de passe doit être : « f6889fc97e14b42de c11a8c183ea791c5465b658 » |
| 00 | Vérifier que l’on se connecte uniquement si le pseudo et le mot de passe correspondent. | Entrer « test » dans le champ pseudo et « Super » dans le champ mot de passe et cliquer sur se connecter. | Notification : « Les identifiants n’existe pas ». |
| 00 | Obliger de remplir tous les champs du formulaire | Entrer « test » dans le champ pseudo et cliquer sur se connecter. | Notification : « Veuillez remplir tous les champs ». |
| 00 | Connexion réussi | Entrer « Ludo » dans le champ pseudo et « Super » dans le champ mot de passe et cliquer sur se connecter. | Notification : « Bienvenu ! Vous êtes connecté ». |

#### Tests sur la page « manageuser.php »

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **A tester** | **Scénario** | **Résultat attendu** |
| 00 | Ajouter un utilisateur | Cliquer sur « nouvel utilisateur », entrer toutes les informations demandées :  - Prénom : Test  - Nom : Test  - Pseudo : Test  - Email : [Test@gmail.com](mailto:Test@gmail.com)  - Date d’anniversaire : 01.01.2019  - Mot de passe : Super  Et cliquer sur enregistrer. | L’utilisateur s’affiche dans la liste de gestion des utilisateurs avec les mêmes informations et dans la base de données. |
| 00 | Modifier un utilisateur | Cliquer sur le crayon à la ligne de l’utilisateur « test » et remplacer son prénom « test » par « test2 ». | L’information est changée. |
| 00 | Supprimer un utilisateur | Cliquer sur la croix à la ligne de l’utilisateur « test » puis sur confirmer. | Notification : « Utilisateur supprimé avec succès ». |

#### Tests sur la page « gestionparking.php »

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **A tester** | **Scénario** | **Résultat attendu** |
| 00 | Ajouter une moto | Cliquer sur « nouvel utilisateur », entrer toutes les informations demandées :  - N° Plaque : 987654  - Marque : Test  - Cylindrée : Test  -Couleur : Test  - Date d’immatriculation : 01.01.2019  - Image : test.png  Et cliquer sur enregistrer. | La moto s’affiche dans la liste de gestion du parking ainsi que sur la page d’accueil. |
| 00 | Modifier une moto | Cliquer sur le crayon à la ligne « test » et changer la marque par « test2 ». Cliquer sur enregistrer. | L’information est changée. |

#### Tests sur la page « managelocation »

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **A tester** | **Scénario** | **Résultat attendu** |
| 00 | Valider une réservation | Cliquer sur le vu à la ligne de « test » puis sur valider. | Notification : « Réservation validée,  Un email est envoyé. |
| 00 | Email | Partager le réseau d’un smartphone par câble USB.  Se connecter à l’adresse mail « ludo.jctdt@eduge.ch ».  Cliquer sur le vu à la ligne où le pseudo correspond à « Ludo  » puis sur valider. | Un mail est reçu dans la boite de réception de « ludo.jctdt@eduge.ch » |
| 00 | Refuser une réservation | Cliquer sur la croix à la ligne de « test » puis sur confirmer. | La réservation est supprimée. |
| 00 | Calendrier Administrateur | Cliquer sur le bouton « Calendrier » | Un calendrier du mois en cours affiche toutes les réservations effectuées. |

### RAPPORT DE TESTS

#### Tests sur la page « index.php »

|  |  |  |
| --- | --- | --- |
| **N°** | **Etat** | **Résultat obtenu** |
| 01 | OK | Uniquement deux motos s’affichent portant le nom de « Yahama ». |
| 02 | OK | Une notification est affichée et la réservation a été prise en compte et donc ajoutée dans la base de données. |
| 03 | OK | Une notification est affichée et l’avis s’est ajouté à la bonne location. |

#### Tests connexion et inscription

|  |  |  |
| --- | --- | --- |
| **N°** | **Etat** | **Résultat obtenu** |
| 00 |  |  |
| 00 | OK | Une notification rouge s’affiche bien avec le texte : « Les identifiants n’existe pas ». |
| 00 | OK | Une notification rouge s’affiche bien avec le texte : « Veuillez remplir tous les champs  ». |
| 00 | OK | Une notification verte s’affiche bien avec le texte : « Bienvenu ! Vous êtes connecté  ». |

# CONCLUSION

## AMELIORATIONS POSSIBLES

## BILAN PERSONNEL

# BIBLIOGRAPHIE

## CODE REPRIS

* Style de la barre de navigation : <https://getbootstrap.com/docs/4.2/examples/blog/>

## SITE UTILISES

* Documentation Bootstrap : <https://getbootstrap.com/>
* Site W3Schools : <https://www.w3scools.com/>
* Documentation PHP : <http://www.php.net/>
* Documentation SQL : <https://sql.sh/>
* Documentation FullCalendar : <https://fullcalendar.io>

# PLANNING

## PLANNING PREVISIONNEL

## PLANNING EFFECTIF

# TABLE DES ILLUSTRATIONS

## FIGURES

# ANNEXES

* Code sources
* Résumé
* Journal de bord manuscrit