

# Dossier de projet

## Développeur Web et Web Mobile

Aout 2023 – Février 2024



Par Roizot Victor

# Sommaire :

1.	Liste des compétences du référentiel qui sont couvertes par le projet.....	2
2.	Résumé de projet.....	3
3.	Cahier des charges, .....	4
	a. Présentation du projet.....	4
	b. Description des besoins.....	5
4.	Spécifications techniques du projet.....	6
	a. Langages et Applications.....	6
	b. Maquettage.....	8
	c. Conception de la base de données .....	11
5.	Réalisations du CRUD (Create, Read, Update, Delete) items.....	14
	a. Create.....	14
	b. Read.....	21
	c. Update.....	24
	d. Delete.....	28
6.	Présentation du jeu d'essai de la fonctionnalité.....	30
7.	Description de la veille sur les vulnérabilités de sécurité.....	34
8.	Description d'une situation de travail ayant nécessité une recherche effectuée à partir d'un site anglophone.....	35
9.	Extrait du site anglophone accompagné de la traduction en français....	36
10.	Conclusion.....	39
11.	Remerciement.....	40

# 1. Liste des compétences du référentiel qui sont couvertes par le projet :

<b>Activité-type 1</b>  Développer la partie Front-End d'une application web ou web mobile en intégrant les recommandations de sécurité	1	Maquetter une application
	2	Réaliser une interface utilisateur web statique et adaptable
	3	Développer une interface utilisateur web dynamique
<b>Activité-type 2</b>  Développer la partie Back-End d'une application web ou web mobile en intégrant les recommandations de sécurité	5	Créer une base de données
	6	Développer les composants d'accès aux données
	7	Développer la partie back-end d'une application web ou web mobile

## **2. Résumé de projet :**

Mon projet « Cabin Escapes » porte sur la location de cabanes insolites.

Les utilisateurs en quête de dépaysement, de déconnexion pour une nuit ou bien un week-end pourront louer un bien dans un cadre hors du commun et en lien avec la nature.

L'administrateur du site pourra rajouter, modifier ou supprimer des cabanes à louer.

Sur mon site de réservation, l'utilisateur pourra voir les différentes cabanes qu'ils soient en couple ou en famille, le tout accompagné de photo et description du lieu.

S'ils souhaitent réserver, rien de plus facile : un calendrier sera présent dans chaque article afin de choisir ces disponibilités.

Un lien les enverra vers un formulaire de réservation, où ils retrouveront le calendrier ainsi que des choix à faire-t-elle que le nombre de personne ou le repas intégré.

La réservation pourra se faire après la création d'un compte utilisateur afin de se connecter et valider la location.

Une rubrique « offrir » sera également présente et qui reprendra quelques éléments du formulaire de réservation avec un choix limité d'options afin d'avoir la possibilité de faire un cadeau à un proche.

Une page sera dédiée pour les infos avec un formulaire de contact, un plan, les distances du site des villes voisinent, train et sortie d'autoroute.

Un dernier onglet présentera des activités par thèmes, proche de la location, un plan sera présent également.

Après leur séjour, les clients pourront de nouveau se connecter pour déposer un avis avec une note et des commentaires et également réserver un autre bien.

### **3. Cahier des charges, description des besoins, ou spécifications fonctionnelles du projet :**

#### **a. Présentation du projet**

Mon projet a été réfléchi dans le but de créer un site de réservation pour un membre de ma famille travaillant le bois et qui souhaite se diversifier dans la location sans passer par une plateforme de réservation.

Dans un premier temps le site permettra de faire découvrir les cabanes, les lieux avoisinants, les activités, les produits du terroir.

Dans un second temps, un utilisateur qui souhaiterait réserver pourra le faire directement sur le site après création d'un compte. Un avis pourra être également être déposé, privilégiant ainsi le circuit court et le contact.

## **b. Description des besoins**

### **Côté utilisateur :**

L'utilisateur peut naviguer sur le site découvrir les activités de la région, se situer sur un plan dynamique, il aura également à disposition un formulaire de contact.

Si l'utilisateur veut louer un logement, il devra créer son compte et se connecter avec son nom, prénom, adresse complète, email et numéro et de téléphone. Une page reprendra ces informations avec un bouton de modification et de suppression de compte.

Lors de la réservation, une date devra être sélectionnée, donnant sur une durée. Des options seront également disponibles pour le nombre de personnes, la catégorie de la cabane (en duo ou en famille), le repas déjeuner compris ou non et les conditions de location.

Enfin, après la réservation chaque client pourra déposer un avis directement sur le site via un bouton sous chaque cabane.

### **Côté administrateur :**

L'administrateur pourra de son côté ajouter une cabane à louer, la rendre visible de tous, la modifier ou la supprimer.

Il ajoutera supprimera ou modifiera également les activités proposées dans la région, ou des lieux à visiter.

## 4. Spécifications techniques du projet :

### a. Langages et application

#### Langages :



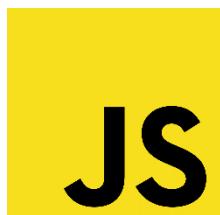
**HTML 5** « HyperText Markup Language »

Langage de balise utilisé pour l'architecture.



**CSS 3** « Cascading style sheets »

Langage qui style et permet la présentation du HTML.



**JAVASCRIPT** est un langage orienté objet, il permet de dynamiser les pages web.



**PHP** « PHP Hypertext Preprocessor »

Langage de script, Il collecte des données des formulaires, manipule les données MySQL.



**SQL** « Structured Query Language »

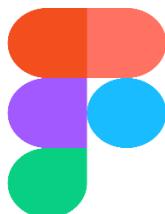
Langage exploitant les bases de données.

## Applications :



### **TRELLO**

Application de gestion, organisateur des tâches et visualisations des étapes.



### **FIGMA**

Application graphique qui permet un maquettage et Une visualisation physique.



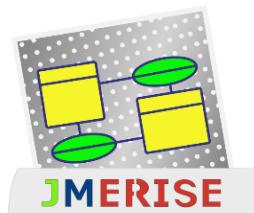
### **GIT et GITHUB**

Logiciel qui permet de sauvegarder des projets.



### **VSCODE « Visual Studio Code »**

Est un éditeur de texte afin d'écrire du code.



### **JMERISE**

Logiciel permettant de crée un MCD ( Modèle Conceptuel de Données) en SQL.



### **WAMPSERVEUR et PHPMYADMIN**

Plateforme environnementale et local d'un serveur pour le fonctionnement de l'applications PHPMyAdmin afin de gérer la gestion de base de données MySQL.

## b. Maquettage

- Lors du maquettage, j'ai commencé par créer deux persona afin de définir l'user story :

### Johanne B :

30 ans, informaticienne, en couple.

#### **Caractère :**

**Passionné (émotionnel, inactif, secondaire) et réservé.**

#### **Motivation :**

Recherche un lieu ou passer du temps en amoureux, à la découverte d'un lieu/environnement nouveau et féerique pour un week-end, et revenir régulièrement si le lieu lui plaît.

Aime les plantes et la nature.

#### **Problème :**

N'aime pas être au centre de l'attention, est réservée au grand public.  
Est connectée, suit l'actualité et est joignable.

### Eddy P :

45 ans, graphiste, marié 2 enfants.

#### **Caractère :**

Passionné (émotionnel, actif ; secondaire) et optimiste,

#### **Motivation :**

Recherche pour un week-end un lieu pour décompresser de la semaine et visiter une région pas trop loin de la maison.

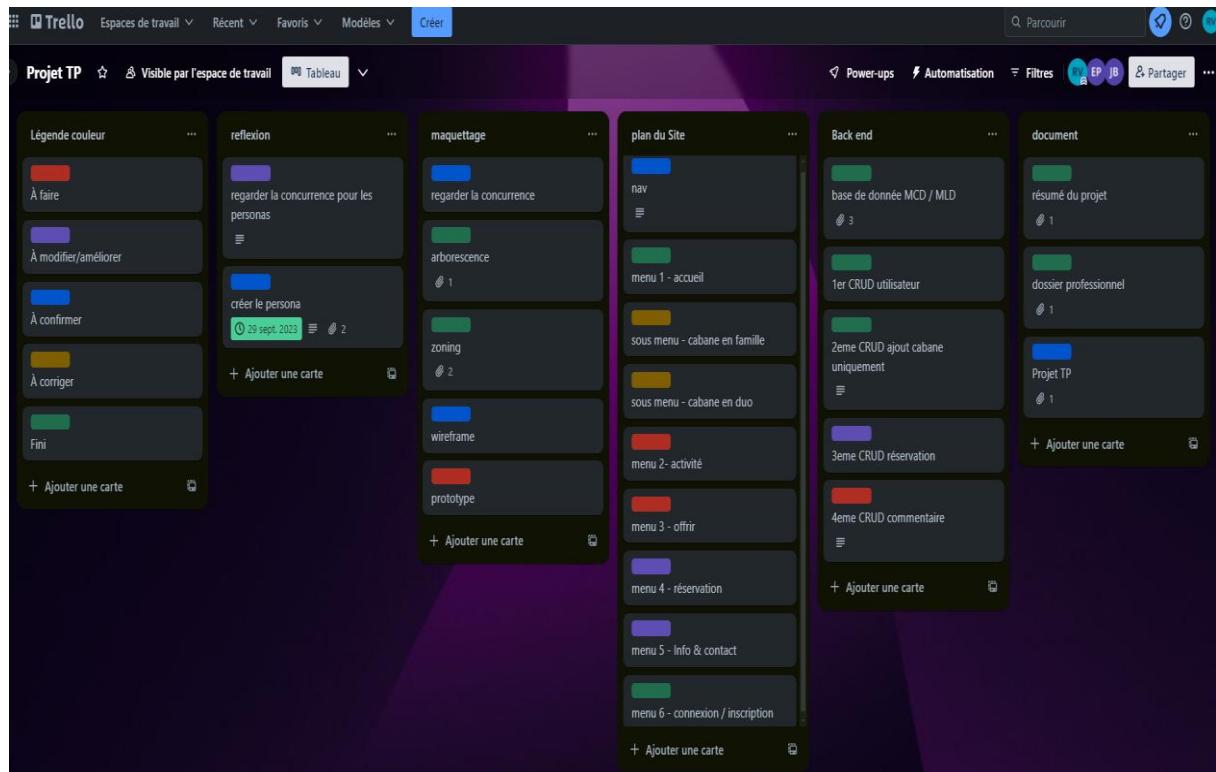
Il a besoin d'un hébergement pour 1 à 2 nuit en rapport avec sa visite pour rester en thème dans cet état de déconnexion, avec une formule tout compris évitant la gestion (petit déjeuner).

Eventuellement pour venir une semaine lors des vacances scolaire et visité davantage ainsi que des monuments historiques.

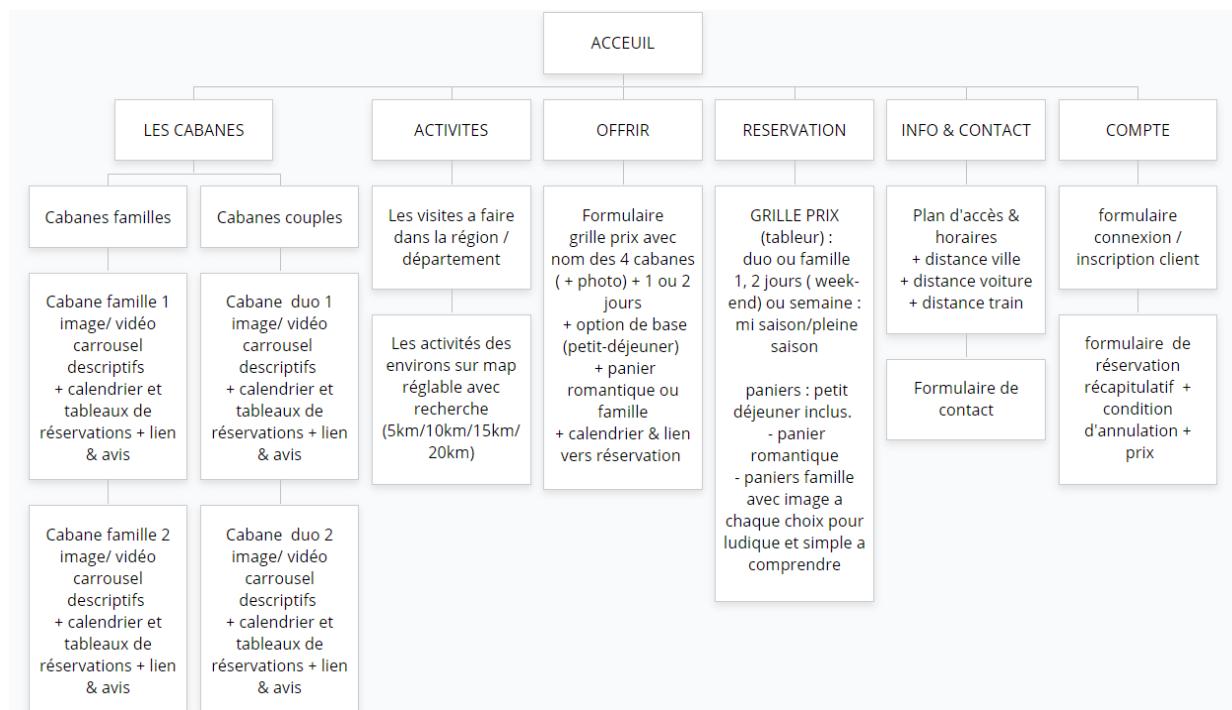
#### **Problème :**

N'a pas de connaissance spécifique à la nature ou lieu à visiter.

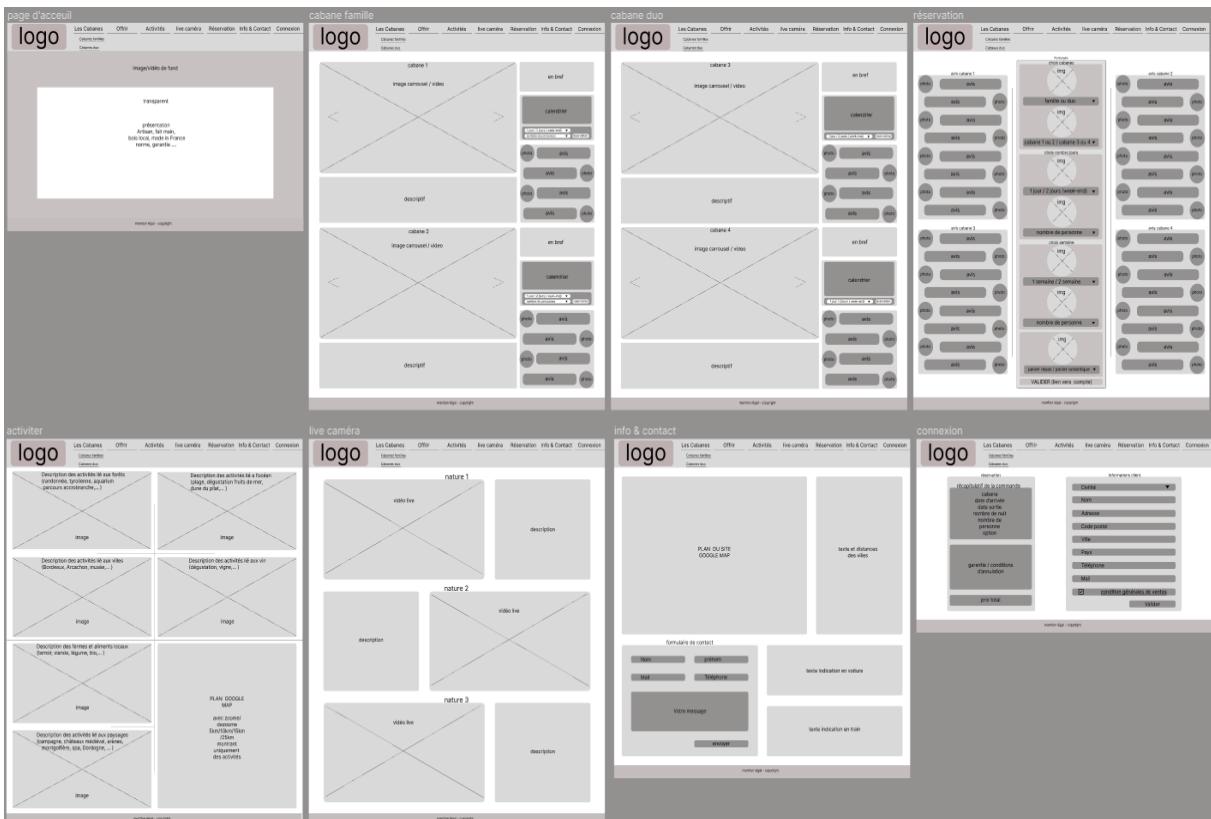
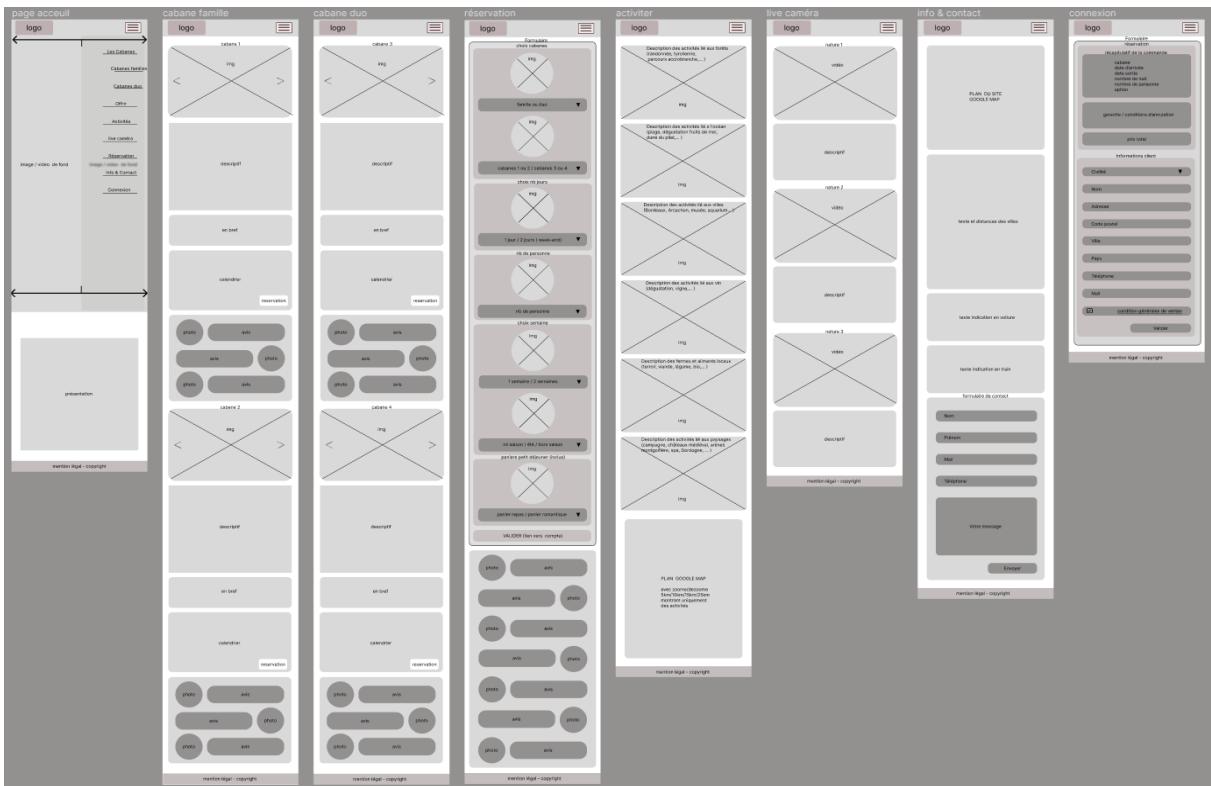
- Ensuite j'ai utilisé Trello pour organiser mes tâches :



- J'ai utilisé GlooMaps pour créer l'arborescence et un premier visuel :

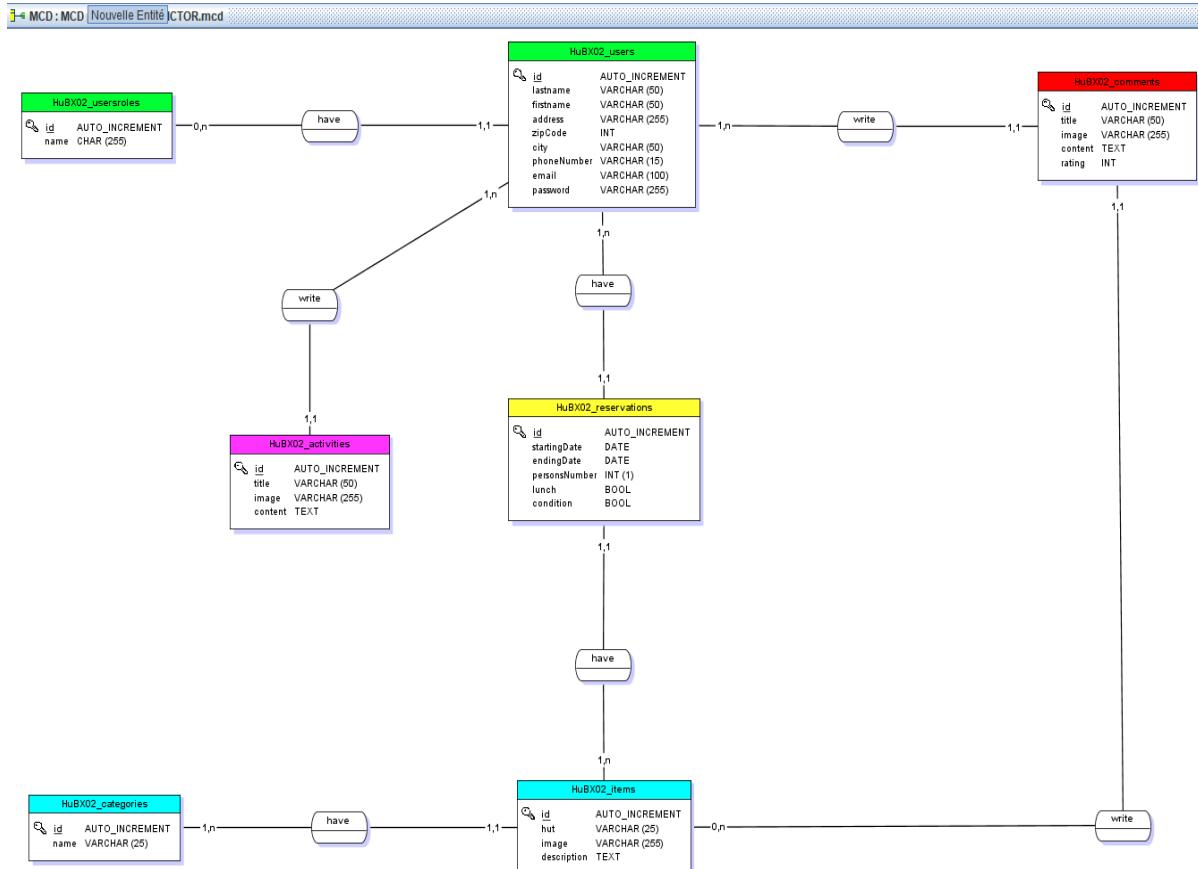


- Le zoning a été réalisé avec Figma en version mobile et ordinateur pour une visualisation physique :



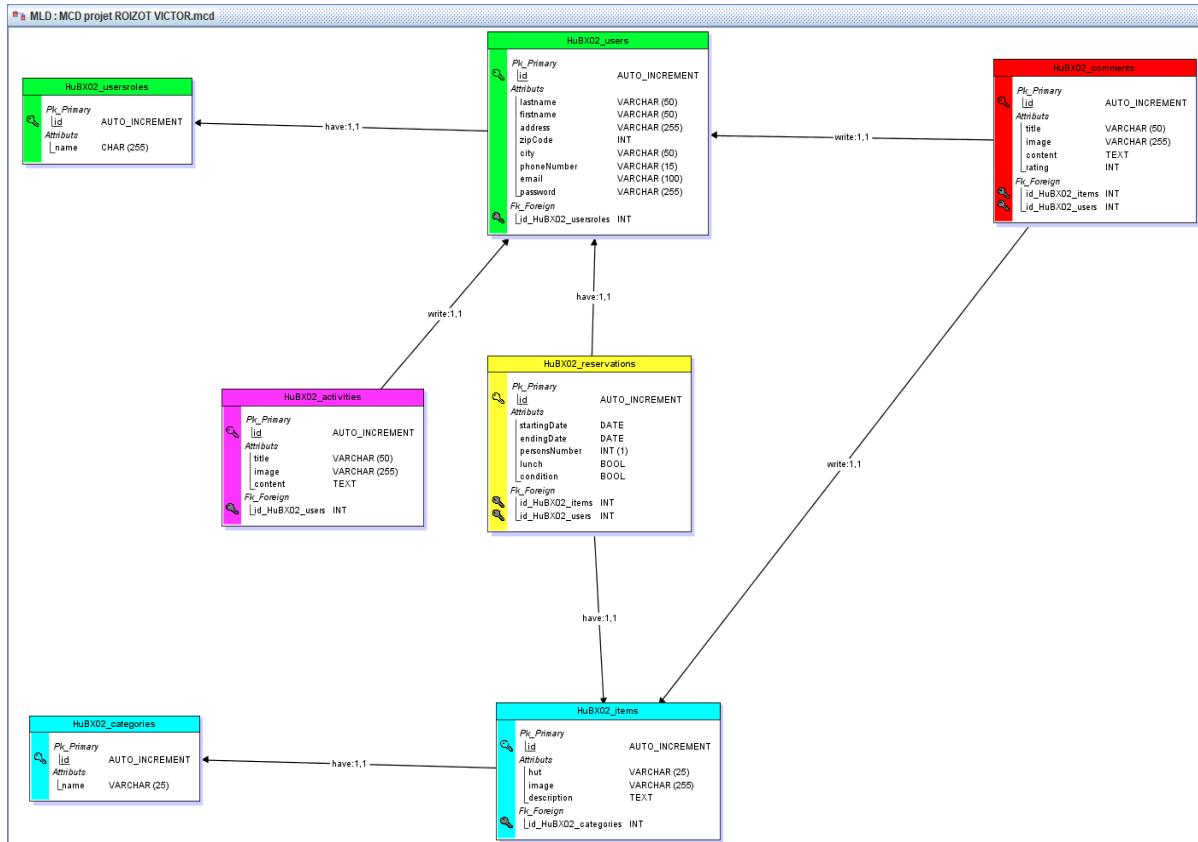
### c. La conception de la base de données :

- Je crée le MCD (Modèle conceptuel de données) avec JMerise afin d'avoir une représentation des données à manipuler.

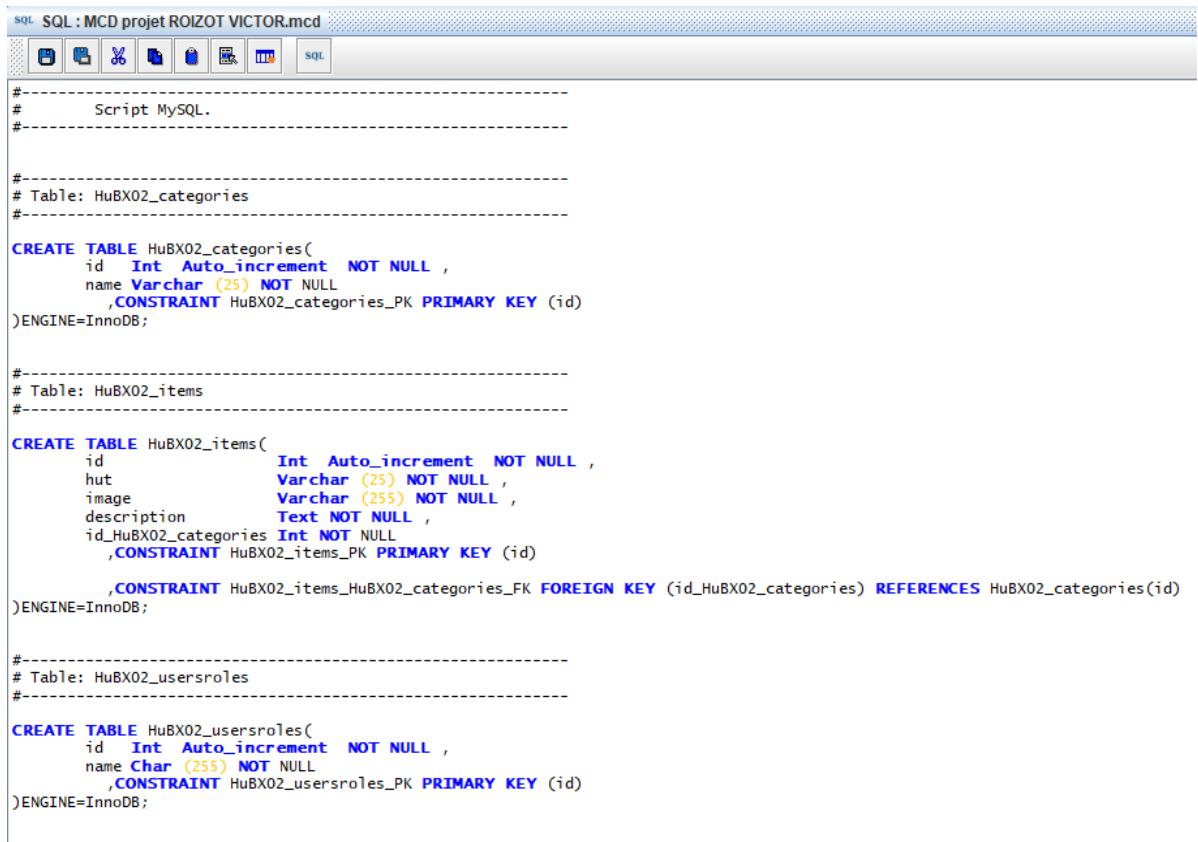


- JMérisé m'a générée le MLD (Modèle Logique de données).

Les informations sont stockées dans des tables, reliées entre elles par des clés étrangères que les cardinalités ont prédéfinies. Elles me serviront afin de faire mes jointures.



- Enfin, le script a été généré à partir du MCD que j'ai créé.



The screenshot shows the MySQL Workbench interface with a tab titled "SQL : MCD projet ROIZOT VICTOR.mcd". The main area displays a multi-line SQL script. The script begins with a header for a MySQL script, followed by three CREATE TABLE statements for tables HuBX02\_categories, HuBX02\_items, and HuBX02\_usersroles. The HuBX02\_items table includes a foreign key constraint linking to the HuBX02\_categories table. The entire script is enclosed in a comment block starting with "#-----".

```

SQL : MCD projet ROIZOT VICTOR.mcd
[File] [Edit] [View] [Tools] [Help] SQL

#-----
#      Script MySQL.
#-----

#-----#
# Table: HuBX02_categories
#-----#

CREATE TABLE HuBX02_categories(
    id      Int Auto_increment NOT NULL ,
    name    Varchar (25) NOT NULL ,
    CONSTRAINT HuBX02_categories_PK PRIMARY KEY (id)
)ENGINE=InnoDB;

#-----#
# Table: HuBX02_items
#-----#

CREATE TABLE HuBX02_items(
    id          Int Auto_increment NOT NULL ,
    hut         Varchar (25) NOT NULL ,
    image       Varchar (255) NOT NULL ,
    description Text NOT NULL ,
    id_HuBX02_categories Int NOT NULL ,
    CONSTRAINT HuBX02_items_HuBX02_categories_FK FOREIGN KEY (id_HuBX02_categories) REFERENCES HuBX02_categories(id)
)ENGINE=InnoDB;

#-----#
# Table: HuBX02_usersroles
#-----#

CREATE TABLE HuBX02_usersroles(
    id      Int Auto_increment NOT NULL ,
    name    Char (255) NOT NULL ,
    CONSTRAINT HuBX02_usersroles_PK PRIMARY KEY (id)
)ENGINE=InnoDB;

```

Nous pouvons voir sur l'image ci-dessus la création de 3 tables :

HuBX02\_categories

HuBX02\_items

HuBX02\_usersroles

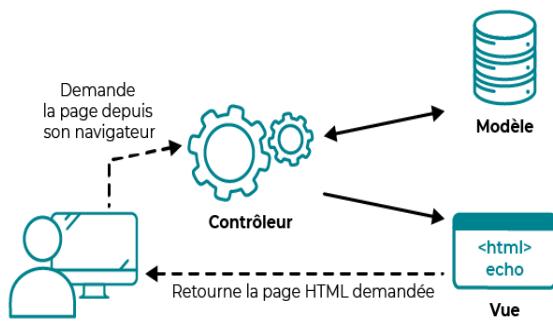
Dans la table HuBX02\_items nous pouvons voir la clé étrangère id\_HuBX02\_categories qui me permettra de faire une jointure avec la table HuBX02\_categories pendant les requêtes SQL.

Le préfixe HuBX02 est un identifiant unique qui permet de sécuriser l'accès.

## 5. Réalisations du CRUD items :

J'ai créé un CRUD items qui est l'article (cabane) à louer. Seul l'administrateur du site pourra le créer, le modifier ou le supprimer mais sera visible par tous les visiteurs sur le site, connectés ou non.

Je vais utiliser l'architecture MVC (Model View Controller) :



### a. Create :

Une fois connecté, administrateur pourra ajouter une cabane à louer par sa page admin. Il devra mettre une photo, un titre choisir une catégorie prédefinie et une description.

Je commence par créer mon modèle *itemsModel.php*

Je crée une *class items* avec ces attributs. Elle me permettra d'envoyer les valeurs récupérées dans le formulaire par les attributs et seront envoyées dans ma base de données grâce au *pdo*.

J'utilise la méthode magique *\_\_construct()* qui est appelée automatiquement lorsque j'instancie la classe.

Je créé une méthode *create* afin de lancer mes requêtes de création.



```
1 <?php
2 class Items
3 {
4     public int $id;
5     public string $hut;
6     public string $image;
7     public string $description;
8     public int $id_categories;
9     private $pdo;
10
11    public function __construct()
12    {
13        try {
14            $this->pdo = new PDO('mysql:host=localhost;dbname=projet_tp;charset=utf8', 'Neo512Mo', '26qJq2Sqb=28C[');
15        } catch (PDOException $e) {
16            header('Location: /erreur-connexion');
17            exit;
18        }
19    }
20 /**
21 * Ajoute un item (cabane à loué) dans la base de données
22 * @param string $hut Le titre de la cabane
23 * @param string $image L'image de la cabane
24 * @param string $description La description de la cabane
25 * @param int $id_categories L'id de la catégorie
26 * @return bool
27 */
28 public function create()
29 {
30     $sql = 'INSERT INTO `hubx02_items` (`hut`, `image`, `description`, `id_categories`)
31 VALUES (:hut, :image, :description, :id_categories)';
32     $req = $this->pdo->prepare($sql);
33     $req->bindValue(':hut', $this->hut, PDO::PARAM_STR);
34     $req->bindValue(':image', $this->image, PDO::PARAM_STR);
35     $req->bindValue(':description', $this->description, PDO::PARAM_STR);
36     $req->bindValue(':id_categories', $this->id_categories, PDO::PARAM_INT);
37     return $req->execute();
38 }
```

Je vais ensuite créer la page de vue *addItem.php*, C'est un formulaire que l'administrateur va remplir. Les informations seront transmises par la méthode ***POST*** pour sécuriser l'envoi via le ‘*name*’ de chaque *input*. Ils seront récupérés dans le contrôleur grâce aux variables *superglobales* `$_POST`.

```
● ○ ●
1 <section>
2     <h1>Ajouter une cabane</h1>
3
4     <?php if (isset($success)) { ?>
5         <p class="success"><?= $success ?></p>
6         <!--Vérification des erreurs liées à l'ajout des cabanes -->
7     <?php } else if (isset($errors['itemAdd'])) { ?>
8         <p class="errors"><?= $errors['itemAdd'] ?></p>
9     <?php } ?>
10
11    <div class="formContainer">
12        <form class="form" action="/ajout-item" method="POST" enctype="multipart/form-data">
13
14            <label for="image">Image de la cabane</label>
15            <input type="file" name="image" id="image">
16            <?php if (isset($errors['image'])) { ?>
17                <p class="errors"><?= $errors['image'] ?></p>
18            <?php } ?>
19
20            <label for="hut">Nom de la cabane</label>
21            <input type="text" name="hut" id="hut">
22            <?php if (isset($errors['hut'])) { ?>
23                <p class="errors"><?= $errors['hut'] ?></p>
24            <?php } ?>
25
26            <label for="categories">Catégorie de la cabane</label>
27            <select name="categories" id="categories">
28                <option selected disabled>Choisissez une catégorie</option>
29                <?php foreach ($categoriesList as $c) { ?>
30                    <option value=<?= $c->id ?>><?= $c->name ?></option>
31                <?php } ?>
32            </select>
33            <?php if (isset($errors['categories'])) { ?>
34                <p class="errors"><?= $errors['categories'] ?></p>
35            <?php } ?>
36
37            <label for="description">Description de la cabane</label>
38            <textarea name="description" id="description"></textarea>
39            <?php if (isset($errors['description'])) { ?>
40                <p class="errors"><?= $errors['description'] ?></p>
41            <?php } ?><br>
42
43            <input class="submit" type="submit" value="Créer">
44
45        </form>
46    </div>
47 </section>
```

Je crée le contrôleur *addItemController* afin de créer toutes les vérifications :

Je commence par démarrer une *session\_start()*; et je récupère les informations de l'administrateur sinon je suis renvoyé à la connexion.

```
● ● ●  
1 // Démarrage de la session  
2 session_start();  
3  
4 // Vérification si l'utilisateur est connecté sinon renvoie vers la connexion  
5 if (empty($_SESSION['user'])) {  
6     header('Location: /connexion');  
7     exit();  
8 }
```

Ensuite j'instancie ma *class items* qui est dans *itemsModel.php* et je vérifie que le formulaire depuis *addItem* est bien envoyé en *POST*.

```
● ● ●  
1 $item = new Items();  
2  
3 $categories = new categories();  
4  
5 $categoriesList = $categories->getList();  
6  
7 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

Je vérifie chaque *input* du formulaire, je le fait correspondre au *regex*, avant de le nettoyer avec une fonction *clean* qui utilise *trim()* qui supprime les espaces au début et à la fin de la chaîne de caractères et *strip\_tags()* qui supprime toute les balises HTML et PHP dans une chaîne de caractères.

Si le formulaire est vide, je stocke le message d'erreur prédéfini da, le tableau *errors*, pareil si le *regex* est pas conforme.

```
● ● ●  
1  if (!empty($_POST['hut'])) {  
2      if (preg_match($regex['hut'], $_POST['hut'])) {  
3          $item->hut = clean($_POST['hut']);  
4      } else {  
5          $errors['hut'] = ITEM_TITLE_ERROR_INVALID;  
6      }  
7  } else {  
8      $errors['hut'] = ITEM_TITLE_ERROR_EMPTY;  
9  }
```

La fonction *clean* :

```
● ● ●  
1  function clean($string)  
2  {  
3      $string = trim($string);  
4  
5      $string = strip_tags($string);  
6      return $string;  
7  }
```

**Les regex :** Elles permettent de filtrer les chaînes de caractères.

```
1 $regex = [
2   'name' => '/^([A-zÄ-ÿ]{1})([ \'-]{1}[A-zÄ-ÿ]{1,}){0,}$/,
3   'address' => '/^(0-9){1,4}( [a-z]{1,})? )?([A-zÄ-ÿ0-9])2,)([- ]{1}[A-zÄ-ÿ0-9]{1,}){1,}$/,
4   'zipCode' => '/^(([0-8]{1}[0-9]{1})|(9[0-5]{1}))[0-9]{3}$/',
5   'city' => '/^([A-zÄ-ÿ]{1})([ \'-]{1}[A-zÄ-ÿ]{1,}){0,}$/,
6   'phoneNumber' => '/(^0[1-79])1{1} ([0-9]{2}){4}$/',
7   'password' => '/^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*\W).{8,}$/,
8   'hut' => '/^([A-zÄ-ÿ0-9]{1})[ \'-]{1}[A-zÄ-ÿ0-9]?\\!:\\/]{1,}$/,
9   'description' => '/(<script>|(&lt;script&gt;))/',
10 ];

```

Les constantes contenant les erreurs qui seront stockées dans le tableau *errors* si les conditions ne sont pas respectées :

```
1 $errors = [];
2 // ITEMS
3 // hut
4 define('ITEM_TITLE_ERROR_EMPTY', 'Le titre est requis');
5 define('ITEM_TITLE_ERROR_INVALID', 'Le titre est invalide. Il ne peut contenir que des lettres, des chiffres, des espaces, des tirets et des apostrophes');
6
7 // image
8 define('ITEM_IMAGE_ERROR_EMPTY', 'L\'image est requise');
9 define('ITEM_IMAGE_ERROR_INVALID', 'L\'image est invalide');
10 define('ITEM_IMAGE_ERROR_EXTENSION', 'L\'image est invalide. Elle doit être au format jpg, jpeg, png, gif ou webp');
11 define('ITEM_IMAGE_ERROR_SIZE', 'L\'image est invalide. Elle doit faire moins de 10Mo');
12 define('ITEM_IMAGE_ERROR', 'Une erreur est survenue lors de l\'envoi de l\'image');
13
14 // description
15 define('ITEM_DESCRIPTION_ERROR_EMPTY', 'Le contenu est requis');
16 define('ITEM_DESCRIPTION_ERROR_INVALID', 'Le contenu est invalide. Il ne peut pas contenir de balises script.');
17
18 // categories
19 define('CATEGORIES_ERROR_EMPTY', 'La catégorie est requise');
20 define('CATEGORIES_ERROR_INVALID', 'La catégorie est invalide');
```

Ci-dessus, vous trouverez un exemple du premier champ '**'hut'**' à remplir dans le formulaire qui correspond au titre de la cabane.

Si aucune erreur est détectée, les informations sont stockées dans la variable superglobale `$_POST['hut']` et récupérées par `itemsModel.php` afin d'être envoyées dans la base de données.

Les différentes pages *php* contenant les vérifications, les requêtes, les messages d'erreurs sont inclus une fois grâce à la commande *require\_once*.

```
1 <?php
2 require_once '../../models/itemsModel.php';
3 require_once '../../models/categoriesModel.php';
4 require_once '../../../../../utils/regex.php';
5 require_once '../../../../../utils/messages.php';
6 require_once '../../../../../utils/functions.php';
```

Les différentes pages *php* dédiées à la vue sont également rappelées.

```
1 // Requires vues
2 require_once '../../views/part/header.php';
3 require_once '../../views/items/addItem.php';
4 require_once '../../views/part/footer.php';
```

## b. Read :

Après la création de l'article, chaque navigateur web pourra consulter l'ensemble des cabanes depuis l'onglet cabane et en cliquant sur « lire la suite » sous chaque article, il ouvrira une page avec seulement la cabane choisie et la description détaillée.

Ma *class items* déjà créée dans *itemsModel.php*, je créé une méthode *getList()* pour récupérer la liste de tous les articles sous forme de tableau avec *fetchAll*

Et je demande qu'elle soit affichée par ordre croissant avec la commande *ORDER BY* et *ASC*

Et je crée la méthode *getById()* afin de récupérer les valeurs d'une cabane sous forme d'objet dans ma base de données.

```
● ● ●
1 public function getById()
2 {
3     $sql = 'SELECT `hut`, `image`, `description`, `name` AS `categorie`, `id_categories`
4         FROM `hubx02_items` AS `i`
5         INNER JOIN `hubx02_categories` AS `c` ON `id_categories` = `c`.`id`
6         WHERE `i`.`id` = :id';
7     $req = $this->pdo->prepare($sql);
8     $req->bindValue(':id', $this->id, PDO::PARAM_INT);
9     $req->execute();
10    return $req->fetch(PDO::FETCH_OBJ);
11 }
12
13 public function getList()
14 {
15     $sql = 'SELECT `i`.`id`, `hut`, `image`, SUBSTR(`description`, 1, 500) AS `description`, `name` AS `categorie`
16         FROM `hubx02_items` AS `i`
17         INNER JOIN `hubx02_categories` AS `c` ON `id_categories` = `c`.`id`
18     ORDER BY `i`.`id` ASC';
19     $req = $this->pdo->query($sql);
20     return $req->fetchAll(PDO::FETCH_OBJ);
21 }
```

Dans ma page *itemListController.php*, j'instancie ma class *items* depuis ma base de données.

```
1 <?php
2 require_once '../../models/itemsModel.php';
3
4 session_start();
5 $item = new Items();
6 $itemsList = $item->getList();
7
8 require_once '../../views/parts/header.php';
9 require_once '../../views/items/itemList.php';
10 require_once '../../views/parts/footer.php';
```

Je fais de même avec le contrôleur *viewItemController.php*. Je fais passer l'*id* de la cabane choisie depuis la liste par la variable superglobale *\$\_GET['id']* car elle ne contient pas d'information sensible.

```
1 <?php
2 require_once '../../models/itemsModel.php';
3 session_start();
4
5 $item = new Items();
6 $item->id = $_GET['id'];
7 if ($item->checkIfExists() == 0) {
8     header('Location: /list-cabanes');
9     exit;
10 }
11 $itemDetails = $item->getById();
12
13 require_once '../../views/parts/header.php';
14 require_once '../../views/items/viewItem.php';
15 require_once '../../views/parts/footer.php';
```

Je passe au visuel de la liste avec la page *itemList.php*. Le *foreach()* me permet de rappeler en boucle chaque variable de la liste.

Ici c'est l'image (*i->image*), le titre (*i->hut*), la catégorie (*i->catgorie*) et la description (*i->description*)

```
● ● ●
1 <section>
2     <h1>cabanes</h1>
3     <div>
4         <?php foreach ($itemsList as $i) { ?>
5             <div>
6                 
7                 <div>
8                     <h2><?= $i->hut ?></h2>
9                     <p>
10                         <b>Catégorie :</b> <?= $i->categorie ?><br>
11                     </p>
12                     <p>
13                         <?= strip_tags($i->description) ?>...
14                     </p>
15                     <a class=lien href="/cabane-<?= $i->id ?>">Lire la suite</a>
16                 </div>
17             </div>
18         <?php } ?>
19     </div>
20 </section>
```

C'est le même principe dans la vue *viewItem.php* à l'exception que nous n'avons pas besoin de boucle, car un seul article nous intéresse.

```
● ● ●
1 <section>
2     <h1>cabanes</h1>
3     <div>
4         
5         <h1><?= $itemDetails->hut ?></h1>
6         <p>
7             <b>Catégorie :</b> <?= $itemDetails->categorie ?><br>
8         </p>
9         <p>
10            <?= strip_tags($itemDetails->description) ?>
11        </p>
12    </div>
13 </section>
```

### c. Update :

Nous continuons avec la modification de l'article. Si l'administrateur souhaite effectuer un changement, sur la photo, le titre, la catégorie ou le descriptif, il le pourra directement depuis l'article, à condition d'avoir le rôle administrateur.

Depuis la page *itemsModel.php*, une nouvelle méthode *updateItem()* est créée, la requête ‘**UPDATE**’ permet de faire les modifications dans la base de données.

Nous préparons la méthode avec la ligne `$req = $this->pdo->prepare($sql);` ;

Elle sera ensuite exécutée avec la ligne `return $req->execute();` ;

Une fois que les marqueurs nominatifs (ici représentés par `:hut` pour le premier exemple) seront remplis depuis le formulaire de la page *updateItem.php*

```
1 public function updateItem()
2 {
3     $sql = 'UPDATE `hubx02_items` SET `hut` = :hut, `image` = :image, `description` = :description, `id_categories` = :id_categories WHERE `id` = :id';
4     $req = $this->pdo->prepare($sql);
5     $req->bindValue(':hut', $this->hut, PDO::PARAM_STR);
6     $req->bindValue(':image', $this->image, PDO::PARAM_STR);
7     $req->bindValue(':description', $this->description, PDO::PARAM_STR);
8     $req->bindValue(':id_categories', $this->id_categories, PDO::PARAM_INT);
9     $req->bindValue(':id', $this->id, PDO::PARAM_INT);
10    return $req->execute();
```

Nous pouvons remarquer dans cette requête plusieurs éléments qui nous garantissent une plus grande sécurité contre les injections SQL :

Par exemple sur la ligne `$req->bindValue(':hut', $this->hut, PDO::PARAM_STR)`

`$req->bindValue()` permet de vérifier le type de donnée.

Il est suivi de **PDO: :PARAM\_STR** qui indique que le type de donnée est une chaîne de caractère (string).

Dans le formulaire de la page *updateItem.php*, l'administrateur remplira de nouveau les champs comme dans la partie create.

J'ai mis des variables *\$itemDetails->hut* dans chaque *INPUT* (comme dans la ligne 21 sur le titre de la cabane avec *\$itemDetails->hut*) afin de faire revenir les informations déjà existantes depuis la base de données dans le formulaire à remplir par l'administrateur.



```
1 <section>
2     <h1>Modifier ma cabane</h1>
3
4     <?php if (isset($success)) { ?>
5         <p class="success"><?= $success ?></p>
6     <?php } ?>
7
8     <?php if (isset($errors['updateItem'])) { ?>
9         <p class="errors"><?= $errors['updateItem'] ?></p>
10    <?php } ?>
11    <div class="formContainer">
12        <form class="form" action="/modifier-item-<?= $item->id ?>" method="post" enctype="multipart/form-data">
13
14            <label for="image">Image de la cabane</label>
15            <input type="file" name="image" id="image">
16            <?php if (isset($errors['image'])) { ?>
17                <p class="errors"><?= $errors['image'] ?></p>
18            <?php } ?>
19
20            <label for="hut">Nom de la cabane</label>
21            <input type="text" name="hut" id="hut" value="<?= $itemDetails->hut ?>">
22            <?php if (isset($errors['hut'])) { ?>
23                <p class="errors"><?= $errors['hut'] ?></p>
24            <?php } ?>
25
26            <label for="categories">Catégorie de la cabane</label>
27            <select name="categories" id="categories">
28                <option selected disabled>Choisissez une catégorie</option>
29                <?php foreach ($categoriesList as $c) { ?>
30                    <option value="<?= $c->id ?>" <?= $itemDetails->id_categories == $c->id ? 'selected' : '' ?><?= $c->name ?></option>
31                <?php } ?>
32            </select>
33            <?php if (isset($errors['categories'])) { ?>
34                <p class="errors"><?= $errors['categories'] ?></p>
35            <?php } ?>
36
37            <label for="description">Description de la cabane</label>
38            <textarea name="description" id="description"><?= $itemDetails->description ?></textarea>
39            <?php if (isset($errors['description'])) { ?>
40                <p class="errors"><?= $errors['description'] ?></p>
41            <?php } ?>
42
43            <input class="submit" type="submit" value="Modifier" name="updateItem">
44        </form>
45    </div>
```

Enfin depuis le contrôleur *updateItemController.php*, Nous vérifions si l'administrateur a les droits, avant d'instancier plusieurs éléments et vérifier les conditions liées à la sécurité comme lors de l'étape de la création, avec les *regex*, la fonction *clean*, Sinon des messages d'erreur seront affichés.

```
1  <?php
2  require_once '../../models/itemsModel.php';
3  require_once '../../models/categoriesModel.php';
4  require_once '../../models/usersModel.php';
5  require_once '../../utils/regex.php';
6  require_once '../../utils/messages.php';
7  require_once '../../utils/functions.php';
8
9  session_start();
10
11 if (!isset($_SESSION['user'])) {
12     header('Location: /connexion');
13     exit;
14 }
15
16 if ($_SESSION['user']['id_usersRoles'] != 255) {
17     header('Location: /list-cabane');
18     exit;
19 }
20
21 $item = new Items();
22 $categories = new categories();
23 $user = new Users();
24 $user->id = $_SESSION['user']['id'];
25 $item->id = $_GET['id'];
26 $imageUploaded = true;
27
28 if ($item->checkIfExists() == 0) {
29     header('Location: /list-cabanes');
30     exit;
31 }
32
33 $itemDetails = $item->getById();
34 $userAccount = $user->getById();
35 $categoriesList = $categories->getList();
36
```

```

1  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
2      if (isset($_POST['updateItem'])) {
3          if (!empty($_POST['hut'])) {
4              if (preg_match($regex['hut'], $_POST['hut'])) {
5                  $item->hut = clean($_POST['hut']);
6              } else {
7                  $errors['hut'] = ITEM_TITLE_ERROR_INVALID;
8              }
9          } else {
10             $errors['hut'] = ITEM_TITLE_ERROR_EMPTY;
11         }
12
13         if (!empty($_FILES['image']['name'])) {
14             $imageMessage = checkImage($_FILES['image']);
15             if ($imageMessage != '') {
16                 $errors['image'] = $imageMessage;
17             } else {
18                 $item->image = uniqid() . '.' . pathinfo($_FILES['image']['name'], PATHINFO_EXTENSION);
19                 while (file_exists('../..../assets/img/items/' . $item->image)) {
20                     $item->image = uniqid() . '.' . pathinfo($_FILES['image']['name'], PATHINFO_EXTENSION);
21                 }
22             }
23         } else {
24             $item->image = $itemDetails->image;
25             $imageUploaded = false;
26         }
27
28         if (!empty($_POST['categories'])) {
29             $categories->id = $_POST['categories'];
30             if ($categories->checkIfExistsById() == 1) {
31                 $item->id_categories = clean($_POST['categories']);
32             } else {
33                 $errors['categories'] = CATEGORIES_ERROR_INVALID;
34             }
35         } else {
36             $errors['categories'] = CATEGORIES_ERROR_EMPTY;
37         }
38
39         if (!empty($_POST['description'])) {
40             if (!preg_match($regex['description'], $_POST['description'])) {
41                 $item->description = clean($_POST['description']);
42             } else {
43                 $errors['description'] = ITEM_DESCRIPTION_ERROR_INVALID;
44             }
45         } else {
46             $errors['description'] = ITEM_DESCRIPTION_ERROR_EMPTY;
47         }
48
49         if (empty($errors)) {
50             if ($imageUploaded == true) {
51                 if (move_uploaded_file($_FILES['image']['tmp_name'], '../..../assets/img/items/' . $item->image)) {
52                     if ($item->updateItem()) {
53                         unlink('../..../assets/img/items/' . $itemDetails->image);
54                         $success = ITEM_UPDATE_SUCCESS;
55                     } else {
56                         unlink('../..../assets/img/items/' . $item->image);
57                         $errors['updateItem'] = ITEM_UPDATE_ERROR;
58                     }
59                 } else {
60                     $errors['updateItem'] = ITEM_UPDATE_ERROR;
61                 }
62             } else {
63                 if ($item->updateItem()) {
64                     $success = ITEM_UPDATE_SUCCESS;
65                 } else {
66                     $errors['updateItem'] = ITEM_UPDATE_ERROR;
67                 }
68             }
69         }
70     }
71 }
72 if (isset($_POST['delete'])) {
73     if ($item->delete()) {
74         unlink('../..../assets/img/items/' . $itemDetails->image);
75         header('Location: /list-cabane');
76         exit;
77     }
78 }
79 $itemDetails = $item->getById();
80
81 require_once '../..../views/parties/header.php';
82 require_once '../..../views/items/updateItem.php';
83 require_once '../..../views/parties/footer.php';

```

## Delete :

Nous terminons avec le dernier CRUD celui de la suppression de l'article.

Je crée toujours depuis la page *itemsModel.php*, une nouvelle méthode *delete()*.

Celle-ci enverra une requête de ‘*DELETE*’ dans la base de données.

```
● ● ●  
1 public function delete()  
2 {  
3     $sql = 'DELETE FROM `hubx02_items` WHERE `id` = :id';  
4     $req = $this->pdo->prepare($sql);  
5     $req->bindValue(':id', $this->id, PDO::PARAM_INT);  
6     return $req->execute();  
7 }
```

Je rajoute la suppression sous le formulaire de modification *updateItem.php*.

Je fais confirmer la suppression par une fenêtre modale afin de confirmer et éviter une action humaine liée à un mauvais geste.

```
● ● ●  
1 <div class="DivDelete">  
2     <div class="containerDelete">  
3         <h2>Supprimer la cabane</h2>  
4  
5         <button class="submitDelete" id="openModalBtn">Supprimer</button>  
6     </div>  
7 </div>  
8  
9 <div id="modalContainer">  
10    <div id="modal">  
11        <span id="closeModalBtn">&times;</span>  
12        <p id="modalText">Êtes-vous sûr de vouloir supprimer la cabane ?</p>  
13        <form class="FormBtnDelete" action="/modifier-item-<?= $item->id ?>" method="POST">  
14            <button class="btnDelete" type="submit" name="delete">Supprimer</button>  
15        </form>  
16    </div>  
17 </div>
```

Dans le contrôleur, la suppression se situe sous la modification de l'article dans la page `updateItemController.php`. Je récupère la validation de la suppression par le `name 'delete'` de l'input contenu dans la fenêtre modale et qui envoie toujours par la variable superglobale `$_POST['delete']`.

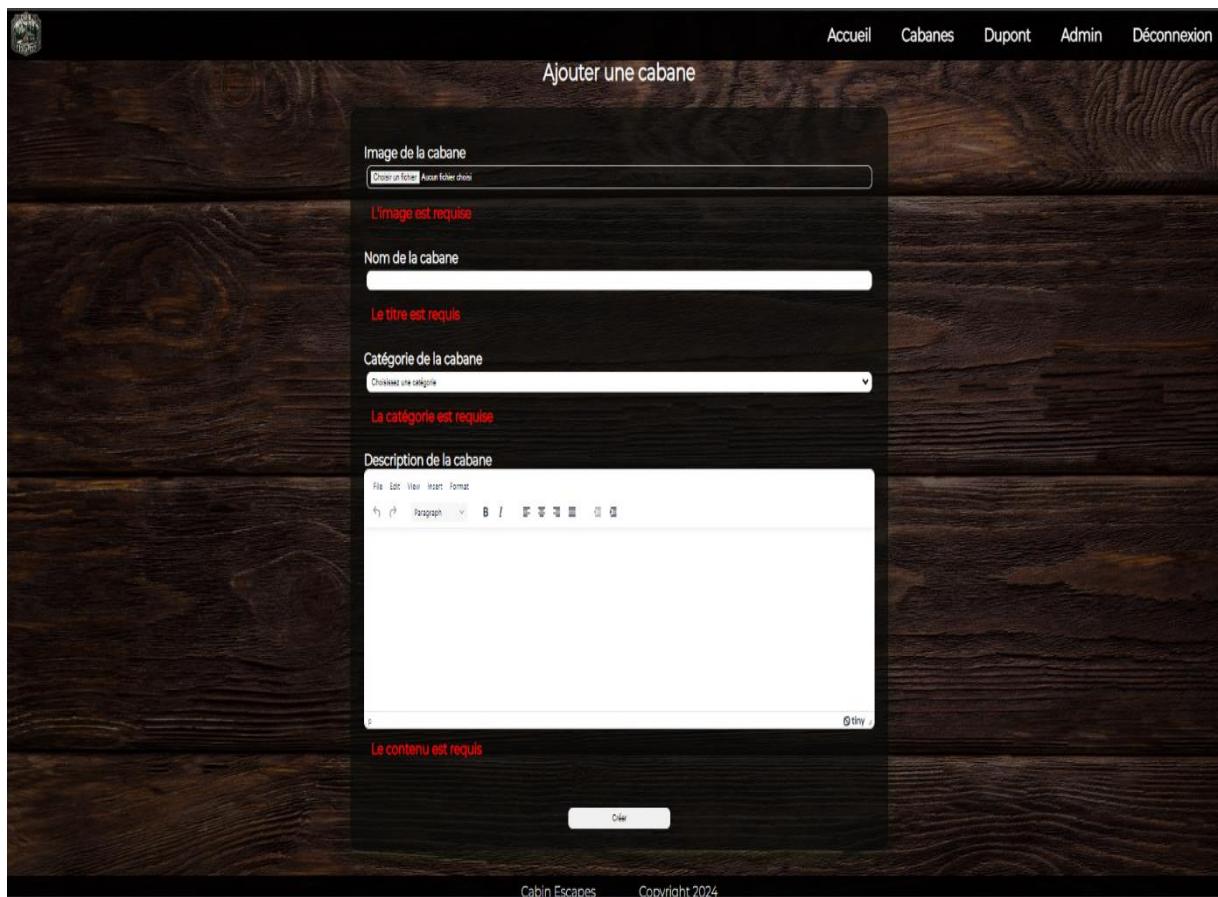
La fonction `unlink()` supprimera l'image qui est stockée dans le dossier `assets/img/items` et non dans la base de données.

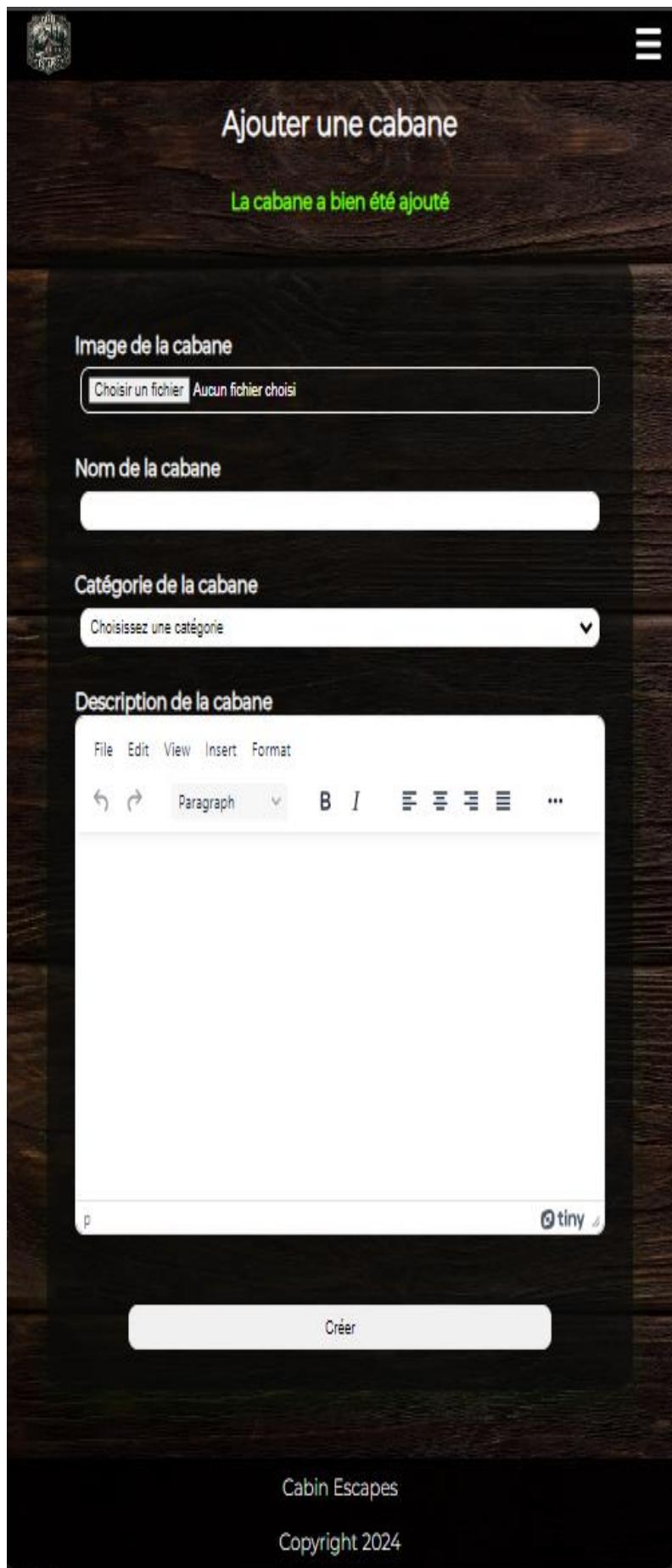
```
● ● ●  
1  if (isset($_POST['delete'])) {  
2      if ($item->delete()) {  
3          unlink('.../.../assets/img/items/' . $itemDetails->image);  
4          header('Location: /list-cabane');  
5          exit;  
6      }  
7  }  
8  
9  $itemDetails = $item->getById();  
10  
11 require_once '.../.../views/parts/header.php';  
12 require_once '.../.../views/items/updateItem.php';  
13 require_once '.../.../views/parts/footer.php';
```

## 6. Présentation du jeu d'essai de la fonctionnalité :

Je teste mon CRUD d'ajout de la cabane avec les messages d'erreur et de succès.

En version ordinateur avec les messages d'erreur :





Et en version mobile.

Avec un message de succès d'ajout de l'article.

La partie visible de la liste des articles à louer depuis l'onglet Cabane en format ordinateur.

The screenshot shows a dark-themed website interface for 'Cabin Escapes'. At the top right are navigation links: Accueil, Cabanes, Dupont, Admin, and Déconnexion. Below them is a title 'Les cabanes'. The main content area displays two cabin listings. On the left, a wooden cabin is shown in a forest setting with the caption 'La cabane du Cerf'. Below it, the category 'Catégorie : Famille' and a detailed description of the cabin's location and features are listed. A 'Lire la suite' button is at the bottom. On the right, another cabin is shown in a clearing with the caption 'La cabane des lapins'. Below it, the category 'Catégorie : Couple' and a similar descriptive text are listed, also with a 'Lire la suite' button. The footer contains the text 'Cabin Escapes Copyright 2024'.

Et la visualisation détaillée d'une cabane après avoir cliqué sur « lire la suite ».

This screenshot shows the detailed view of 'La cabane du Cerf' from the previous page. The top navigation and 'Cabanes' link are identical. The main image is a larger, more detailed view of the cabin in its forest setting. Below the image, the title 'La cabane du Cerf' and category 'Catégorie : Famille' are repeated. A significantly longer and more detailed description follows, mentioning a scavenger hunt for 10 small deer figurines, a large double bed, simple single beds, a dining table, chairs, and transats. It also describes a breakfast service by a fairy chimney. The description ends with a note about local products and a WiFi connection. A 'modifier' button is at the bottom left.

Enfin, si l'utilisateur est l'administrateur il pourra cliquer sur le bouton « modifier ma cabane » pour modifier l'article ou la supprimer comme ci-joint en format mobile.

The image consists of two side-by-side screenshots of a mobile application interface. Both screenshots have a dark wood-grain background.

**Screenshot 1 (Left):** This screenshot shows the 'Modifier ma cabane' (Edit Cabin) screen. It includes fields for 'Image de la cabane' (Cabin Image), 'Nom de la cabane' (Cabin Name), 'Catégorie de la cabane' (Cabin Category), and 'Description de la cabane' (Cabin Description). The 'Description' field contains a rich text editor with a toolbar and a preview area showing a paragraph about a cabin named 'La cabane du Cerf'. A 'Modifier' button is at the bottom.

**Screenshot 2 (Right):** This screenshot shows a confirmation dialog box titled 'Supprimer la cabane' (Delete Cabin). The dialog asks 'Etes-vous sûr de vouloir supprimer la cabane ?' (Are you sure you want to delete the cabin?). It features a red 'Supprimer' (Delete) button. Below the dialog, the text 'Cabin Escapes' and 'Copyright 2024' is visible.

## 7. Description de la veille sur les vulnérabilités de sécurité :

Nous avons évoqué plusieurs manière de protéger mon site un ainsi que la base de données (Les préfixes HuBX02, les variables superglobales `$_POST`, les fonction `trim()` et `strip_tags()`, le `bindValue()`, Le **PDO: :PARAM\_STR**, les regex, le tableau d'erreur...) durant la partie 5 avec les CRUD, car il était plus simple de l'expliquer avec des images.

Nous pouvons rajouter la fonction `filter_var()` pour l'email qui a la même fonctionnalité que le `regex`, il filtre et valide une adresse email ou encore `password_hash()` pour hasher et chiffrer le mot de passe, que j'ai utilisé lors de l'inscription d'un utilisateur.

```
● ● ●  
1  (filter_var($_POST['email'], FILTER_VALIDATE_EMAIL))  
2      $user->password = password_hash($_POST['password'], PASSWORD_DEFAULT);
```

Je vais également évoquer le fichier `.htaccess` : il permet de configurer les différentes pages et répertoires en réécrivant l'adresse url pour plus de sécurité.

## **8. Description d'une situation de travail ayant nécessité une recherche effectuée à partir d'un site anglophone :**

Pendant la création du CRUD create, j'ai recherché des informations pour ajouter une image de l'items dans mon dossier assets sans pour autant écraser un autre fichier qui aurait le même nom ou si la photo est utilisée deux fois.

## 9. Extrait du site anglophone accompagné de la traduction en français :

J'ai trouvé ma réponse sur w3schools avec la fonction *uniqid()*.

Voici l'extrait du site : [https://www.w3schools.com/php/func\\_misc\\_uniqid.asp](https://www.w3schools.com/php/func_misc_uniqid.asp)

### PHP uniqid() Function

« PHP Misc Reference

#### Example

Get your own PHP Server

Generate a unique ID:

```
<?php  
echo uniqid();  
?>
```

[Try it Yourself »](#)

#### Definition and Usage

The *uniqid()* function generates a unique ID based on the microtime (the current time in microseconds).

#### Syntax

```
uniqid(prefix,more_entropy)
```

#### Parameter Values

Parameter	Description
<i>prefix</i>	Optional. Specifies a prefix to the unique ID (useful if two scripts generate ids at exactly the same microsecond)
<i>more_entropy</i>	Optional. Specifies more entropy at the end of the return value. This will make the result more unique. When set to TRUE, the return string will be 23 characters. Default is FALSE, and the return string will be 13 characters long

#### Technical Details

<b>Return Value:</b>	Returns the unique identifier, as a string
<b>PHP Version:</b>	4+
<b>Changelog:</b>	The <i>prefix</i> parameter became optional in PHP 5.0. The limit of 114 characters long for <i>prefix</i> was raised in PHP 4.3.1.

Nous pouvons traduire ici :

## PHP uniqid() fonction

< Divers référence PHP

### Exemple

Obtenez votre propre serveur PHP

Générer un unique ID :

```
< ?php  
Echo uniqid();  
?>
```

Essayer le vous-même »

### Définition et usage

La fonction uniqid() génère un identifiant unique basé sur le micro temps (l'heure actuelle en micro secondes).

### Syntaxe

```
uniqid(prefix,more_entropy)
```

### Valeurs des paramètres

Paramètres	Description
<i>Prefix</i>	optionnel. Rajouter un préfixe pour l'unique ID (utile si deux scripts génèrent des identifiants exactement à la même micro secondes)
<i>more_entropy</i>	optionnel. Rajouter <i>more_entropy</i> à la fin de la valeur de retour. Ça rendra le résultat encore plus unique. Lorsqu'elle est définie sur <i>vrai</i> , le retour de la chaîne de caractères sera de 23 caractères. La valeur par défaut est faux, et le retour de la chaîne de caractères sera longue de 13 caractères.

## Détails techniques

<b>Valeur de retour :</b>	Renvoie l'identifiant unique, sous forme de chaîne
<b>Version PHP</b>	4 +
<b>Journal des modifications :</b>	<p>Le paramètre préfixe est devenu optionnel dans PHP 5.0</p> <p>La limite de 114 caractères pour le préfixe a été augmentée dans PHP 4.3.1</p>

## 10. Conclusion :

Mon projet a commencé sein d'un cocon familial, pour mon frère, à quelque mois de la formation et que j'ai continué à faire murir au centre.

Malgré un début difficile, un déclic m'est survenu au moment de l'apprentissage du Back-End et plus particulièrement pendant l'apprentissage du SQL dont j'ai apprécié et pris du plaisir ainsi que le langage PHP.

Je souhaite poursuivre mon apprentissage en alternance si possible, sinon m'orienter davantage sur le Back-End.

Mon projet n'est pour l'heure pas encore fini, en effet je souhaite ajouter une partie réservation avec un calendrier, un autre CRUD avec des commentaires, une page d'activités et découvertes de la région accompagnée d'une carte dynamique, ainsi qu'une dernière page contenant un formulaire de contact, une carte et des informations sur le lieu.

Mais je n'en reste pas moins fier de pouvoir présenter mon projet et montrer les compétences que j'ai acquis durant cette étape enrichissante.

Je suis satisfait du chemin que j'ai parcouru durant ces 6 mois.

J'ai hâte de poursuivre mon apprentissage et continuer à évoluer.

## **11. Remerciement :**

Tout d'abord, je tiens à remercier Mme Margot CORBIN sans qui je n'aurais pas pu intégrer l'école de LA MANU dans cette formation de développeur web et web mobile.

Ensuite je remercie plus particulièrement et très sincèrement Johanne BADIN et Eddy PHILIPPO, mes formateurs, qui mon appris bien plus que de la connaissance, toujours à l'écoute et dans l'aide, sans qui ce projet n'aurais jamais été aussi abouti.

Je remercie également ma famille qui mon tous encourager dans ma reconversion professionnelle et plus particulièrement ma femme qui m'a soutenu et a su me libérer partiellement de mes tâches quotidiennes et de la charge des enfants afin de me consacrer pleinement à ma formation.

Pour finir, j'adresse mes derniers remerciements à ma promo et aux autres apprenants, avec qui nous avons passé des bons moments comme des moments difficile mais toujours dans l'entraide et la bonne humeur.