Titre professionnel Développeur Web et Web Mobile - Niveau V

Dossier de projet



Projet de boutique E-commerce SEO

Résumé de projet

Pour mon projet de fin de formation, j’ai souhaité réaliser un site e-commerce simple et administrable à l’aide d’un back office.

Bien que trop souvent qualifiés *d’usines à gaz*, **Prestashop, Woo** **…** offrent des solutions parfaites permettant de déployer rapidement des business ou des concepts en ligne. Malgré des avantages indéniables, ces plateformes présentent néanmoins des inconvénients quant à la performance générale (temps de chargement des pages, du code non utilisé …).

Ayant étudié les différentes stratégies d’acquisition de trafic, le référencement naturel reste une solution pérenne et la moins onéreuse parmi les autres. C’est également la stratégie qui apporte le plus grand taux de conversion d’achats, à condition bien évidemment de se ranger en première page de la S.E.R.P.

Pour ce projet, mes efforts se sont donc concentrés sur les bonnes pratiques du **SEO On Site** en appliquant les conseils du livre *« Référencement Google – Mode d’emploi*» écrit par **Olivier Andrieu** ; sans pour autant en oublier la technique employée ainsi que la vigilance apportée autour de la sécurité du site.

Vous l’aurez donc compris, ce projet implémente des fonctionnalités de base d’un site e-commerce, où les tables de la base de données ont été enrichies de champs et de contenus optimisés pour le SEO.

Table des matières

[Table des matières 4](#_Toc97818719)

[1.Cahier des charges 5](#_Toc97818720)

[1.1 Descriptif de la demande 5](#_Toc97818721)

[1.2 Architecture technique 5](#_Toc97818722)

[1.3 Spécifications fonctionnelles 5](#_Toc97818723)

[a) Description 5](#_Toc97818724)

[b) Utilisation 6](#_Toc97818725)

[1.4 Spécifications techniques 7](#_Toc97818726)

[a) Langages utilisés 7](#_Toc97818727)

[b) Bibliothèques et frameworks utilisés 7](#_Toc97818728)

[c) Outils utilisés 8](#_Toc97818729)

[d) Sécurité des systèmes d’information : Recommandations et bonnes pratiques 8](#_Toc97818730)

[2. Compétences du référentiel couvertes 9](#_Toc97818731)

[2.1 Tableau récapitulatif 9](#_Toc97818732)

[2.2 Maquetter une application 10](#_Toc97818733)

[2.3 Réaliser une interface utilisateur web statique et adaptable 11](#_Toc97818734)

[2.4 Développer une interface utilisateur web dynamique 12](#_Toc97818735)

[*2.5 Créer une base de données* 12](#_Toc97818736)

[2.6 Développer des composants d’accès aux données 14](#_Toc97818737)

[2.7 Développer la partie back-end d’une application web ou web mobile 15](#_Toc97818738)

[*3. Persistance des données via une BDD* 16](#_Toc97818739)

[3.1 Création d’un évènement 16](#_Toc97818740)

[*3.2 Affichage des données de l’utilisateur* 20](#_Toc97818741)

[*3.3 Modification des évènements* 24](#_Toc97818742)

[*3.4 Suppression d’un compte utilisateur* 27](#_Toc97818743)

[4. Documentation en anglais 29](#_Toc97818744)

[5. Conclusion 30](#_Toc97818745)

# 

# 1.Cahier des charges

## 1.1 Descriptif de la demande

Développer la partie front-end et la partie back-end d’une application web et web mobile. Concevoir une base de données et développer les composants d’accès aux données. Un C.R.U.D au minimum est demandé.

Le choix de la thématique étant libre, j’ai souhaité réaliser un site marchand.

## 1.2 Architecture technique

Le site e-commerce adopte une architecture MVC dite Light sans routage d’URL, utilisant le langage de programmation PHP pour le back-end, et HTML, CSS, Bootstrap et Javascript pour le front-end. Il a été déployé sur un serveur WAMP.

## 

## 1.3 Spécifications fonctionnelles

### **a) Description**

Une catégorie est définie par :

* Un nom
* Une position
* Un slug

Une collection est définie par :

* Un nom
* Une position
* Un slug

Un produit est défini par :

* Un nom
* Un slug
* Un prix
* Une remise
* Un état
* Une ou des options
* Une description
* Un Meta titre
* Une Meta description
* Une ou des images

Un utilisateur est défini par :

* Une adresse mail
* Un mot de passe
* Un nom
* Un prénom
* Une date d’enregistrement
* Une adresse postale
* Un état d’activation du compte
* Un état de l’inscription à la Newsletter

La table newsletter est définie par :

* Une adresse mail

### 

### **b) Utilisation**

L’utilisation du site respecte une hiérarchie à 2 niveaux :

* L’Administrateur (Gestionnaire du site)
* Le client inscrit ou non.

**BACK-OFFICE**

Les fonctionnalités côté administration

**Création d’un produit**

L’administrateur a la possibilité d’ajouter un nouveau produit à ses collections.

Il a également la possibilité de créer une nouvelle catégorie ainsi qu’une nouvelle collection.

Il peut aussi personnaliser l’ordre des éléments de la Navbar à l’aide d’un glisser/déposer.

**Affichage d’un produit**

Un tableau liste tous les produits de la boutique avec l’aide d’une pagination.

Une fonction de recherche par nom de produit a été implémentée afin de cibler un produit plus rapidement.

**Modification d’un produit**

Une page d’édition est disponible afin de modifier les attributs d’un produit :

* Modification des informations générales (nom, prix, remise, options, changement de collection)
* Modification des informations liées au SEO (Meta titre, description)
* Upload / Modification / Suppression des images du produit.
* Modification de la description du produit
* Activation ou désactivation du produit

**Suppression d’un produit**

L’administrateur peut effectuer une action individuelle de suppression sur chacun des produits.

Il a la possibilité de réaliser une suppression multiple à l’aide d’une action groupée.

**Affichage et suppression d’un utilisateur**

L’administrateur de la boutique a un droit de regard sur les utilisateurs inscrits.

Il peut ainsi parcourir la liste des utilisateurs inscrits, rechercher un utilisateur par son nom de famille.

Il peut aussi supprimer un ou plusieurs utilisateurs à l’aide d’une action individuelle ou encore groupée.

**FRONT-OFFICE**

Les fonctionnalités côté utilisateurs

**Présentation des produits à l’utilisateur**

L’utilisateur de la boutique peut naviguer d’une collection vers un produit.

* Afficher un article (fiche produit)
* Afficher plusieurs articles (pages des collections)
* Trier les articles des collections par prix croissants ou décroissants (+ pagination)

**Inscription/Connexion**

L’utilisateur peut s’inscrire en tant que membre/client de la boutique et ainsi se connecter à son compte.

Il a la possibilité de modifier :

* Son identité
* Son mot de passe
* Son adresse de livraison

**Inscription à la Newsletter**

Un utilisateur peut profiter des offres privées en s’inscrivant à la Newsletter.

**Communication**

Un utilisateur a la possibilité de contacter le gestionnaire de la boutique à l’aide d’un formulaire de contact.

**Shopping**

Un utilisateur peut ajouter des articles à son panier, modifier les quantités ou encore supprimer un article de son panier.

## 1.4 Spécifications techniques

### Langages utilisés

|  |  |
| --- | --- |
| Appellation | Description |
| HTML 5 | HTML5 (HyperText Markup Language 5) est la dernière révision majeure du HTML. C’est un format de données conçu pour représenter les pages web. C’est un langage de balisage permettant de l’hypertexte. |
| CSS 3 | Les feuilles de style en cascade, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML. |
| PHP 7.4.26 | PHP: Hypertext Preprocessor, est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet. |
| JavaScript | JavaScript est un langage événementiel de programmation de scripts, principalement employé dans les pages web interactives. |
| MySQL 5.7.36 | Langage de requête fonctionnant avec le système de gestion des bases de données relationnelles. Il permet la génération de la base de données, la lecture et la modification de ses tables. |

### Bibliothèques et frameworks utilisés

|  |  |
| --- | --- |
| Appellation | Description |
| Bootstrap 5.0.2 | Bootstrap est un framework open source de développement web orienté interface graphique. Ce framework est pensé pour développer des sites avec un design responsive, qui s'adapte à tout type d'écran, et en priorité pour les smartphones. |
| SweetAlert2 11.0 |  |
| Lightbox |  |
| JQuery 3.6.0 |  |
| jsDelivr |  |

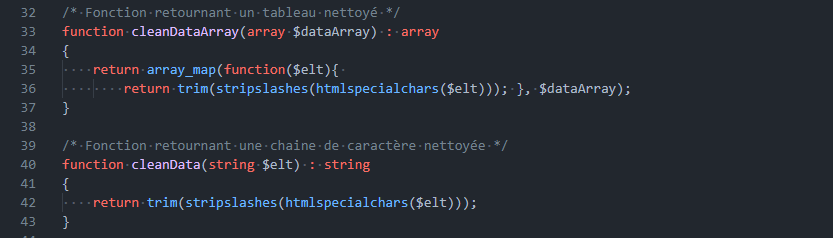
### Outils utilisés

|  |  |
| --- | --- |
| Appellation | Description |
| Virtual Studio Code | Visual Studio Code est un éditeur de code multi-plateformes, open-source et gratuit, supportant une dizaine de langages. |
| Git | Git est un logiciel de gestion de versions décentralisé. Il permet de travailler avec des services d’hébergement spécifiques tel que GitHub et BitBucket. |
| MySQL Workbench | Logiciel de gestion et d'administration de bases de données MySQL |
| phpMyAdmin | phpMyAdmin (PMA) est une application Web de gestion pour les systèmes de gestion de base de données MySQL réalisée principalement en PHP. |
| JMerise | Logiciel d’aide à la conception des bases de données (Modèle Conceptuel de Données) qui permet la génération d’un Modèle Logique de Données et d’un script SQL pour plus de facilité. |

### Sécurité des systèmes d’information : Recommandations et bonnes pratiques

Afin de me prémunir d’attaques potentielles et donc de protéger ma boutique marchande, j’ai mis en place les pratiques suivantes :

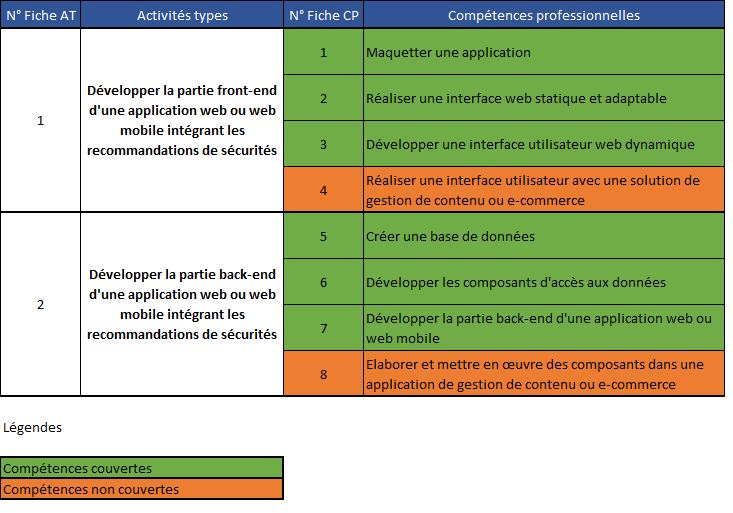
* Contre la faille XSS, chaque donnée saisie par un utilisateur doit être passée dans la fonction PHP **htmlspecialchars()**



* Contre les injections SQL, j’ai utilisé les requêtes préparées de PDO (Méthodes PDO : **prepare()** , **bindValue()** , **execute()** )
* Des expressions régulières ont été utilisées comme pattern afin de contrôler le bon format d’une saisie utilisateur, elles sont utilisées notamment avec la fonction PHP **preg\_match()**.  
  La fonction **filter\_var()** a été utilisée pour le contrôle des adresses e-mails.
* Les bonnes pratiquent imposent de préfixer les tables de sa base de données.
* Lors de l’enregistrement d’un utilisateur dans la table des « users », il est impératif de créer une clé de hachage pour le champ « password ».  
  On utilise la fonction PHP **password\_hash()**.   
  Les mots de passe ne doivent jamais être stockés en clair.  
  Lorsqu’une connexion échoue, le message d’erreur ne doit pas préciser si c’est l’adresse mail qui est incorrecte ou bien le mot de passe.
* L’arborescence des dossiers doit être cachée aux utilisateurs à l’aide du fichier **.htaccess**.
* Vérifier si l’utilisateur connecté ou non, a les droits nécessaires pour accéder à une page
* Créer des méthodes utiles afin de vérifier l’existence d’une donnée en BDD.

# 2. Compétences du référentiel couvertes

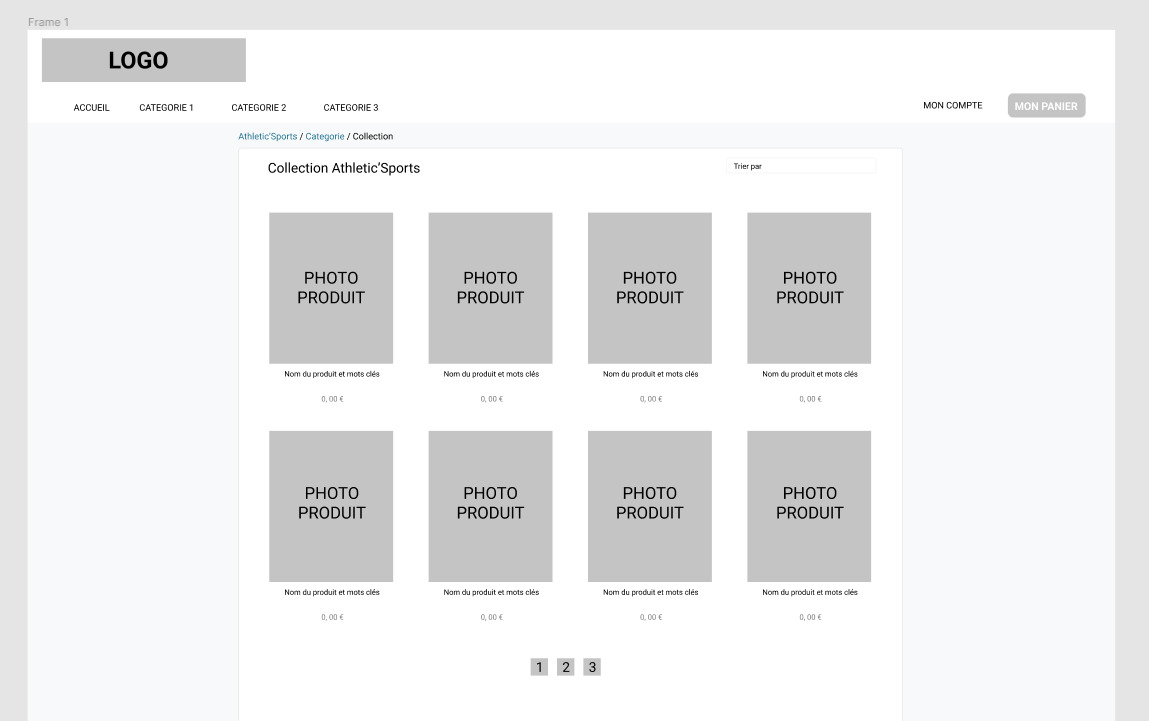
## 2.1 Tableau récapitulatif



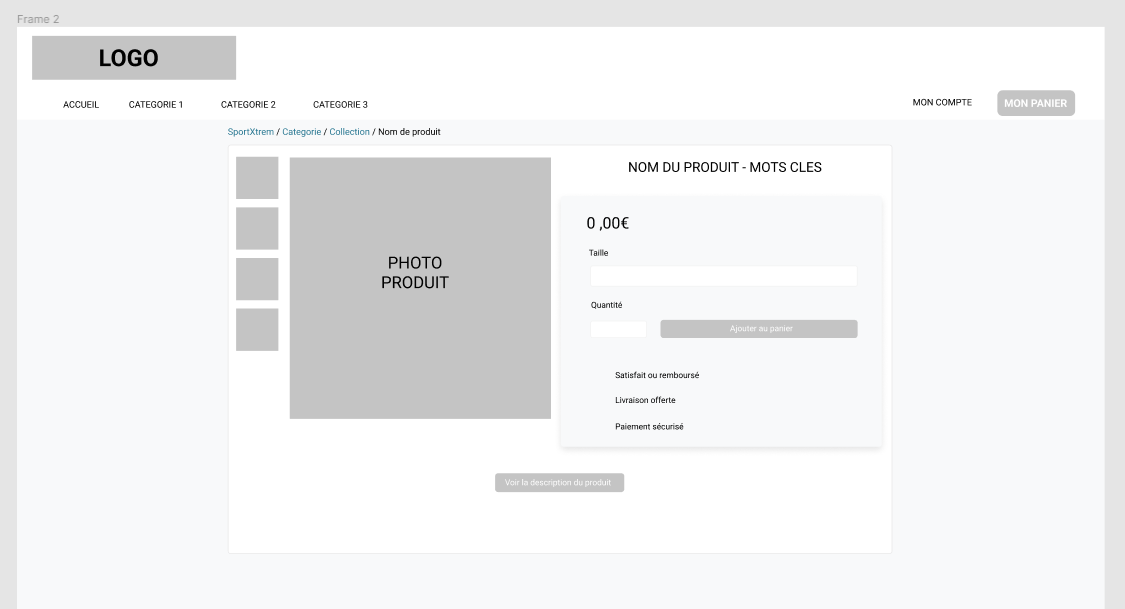
## 2.2 Maquetter une application

Le maquettage est une étape primordiale dans la conception de son projet, il permet à la fois d’avoir un visuel et une vue d’ensemble.

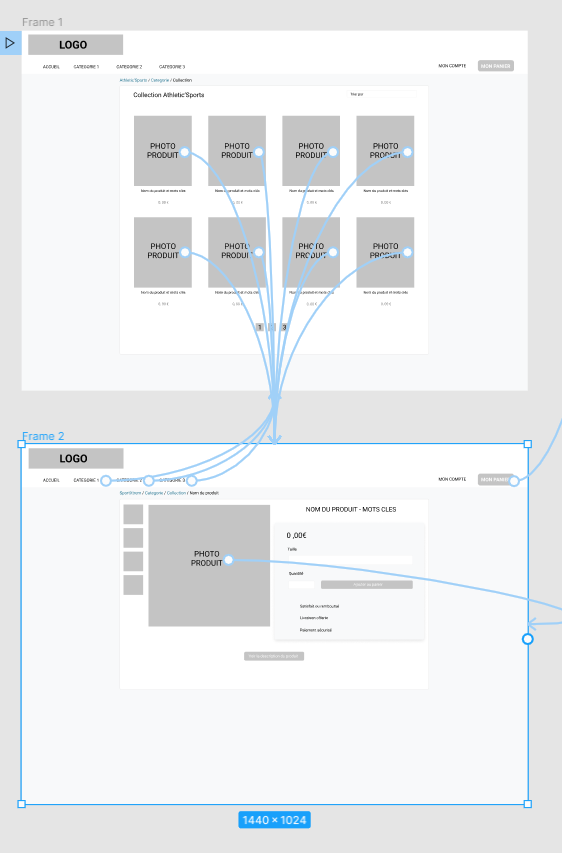
Mais surtout, d’en déduire les fonctionnalités dont on aura besoin pour développer son projet. Cette maquette a été réalisée avec l’outil en ligne **Figma**.



Page des collections - Figma



Page de la fiche produit - Figma



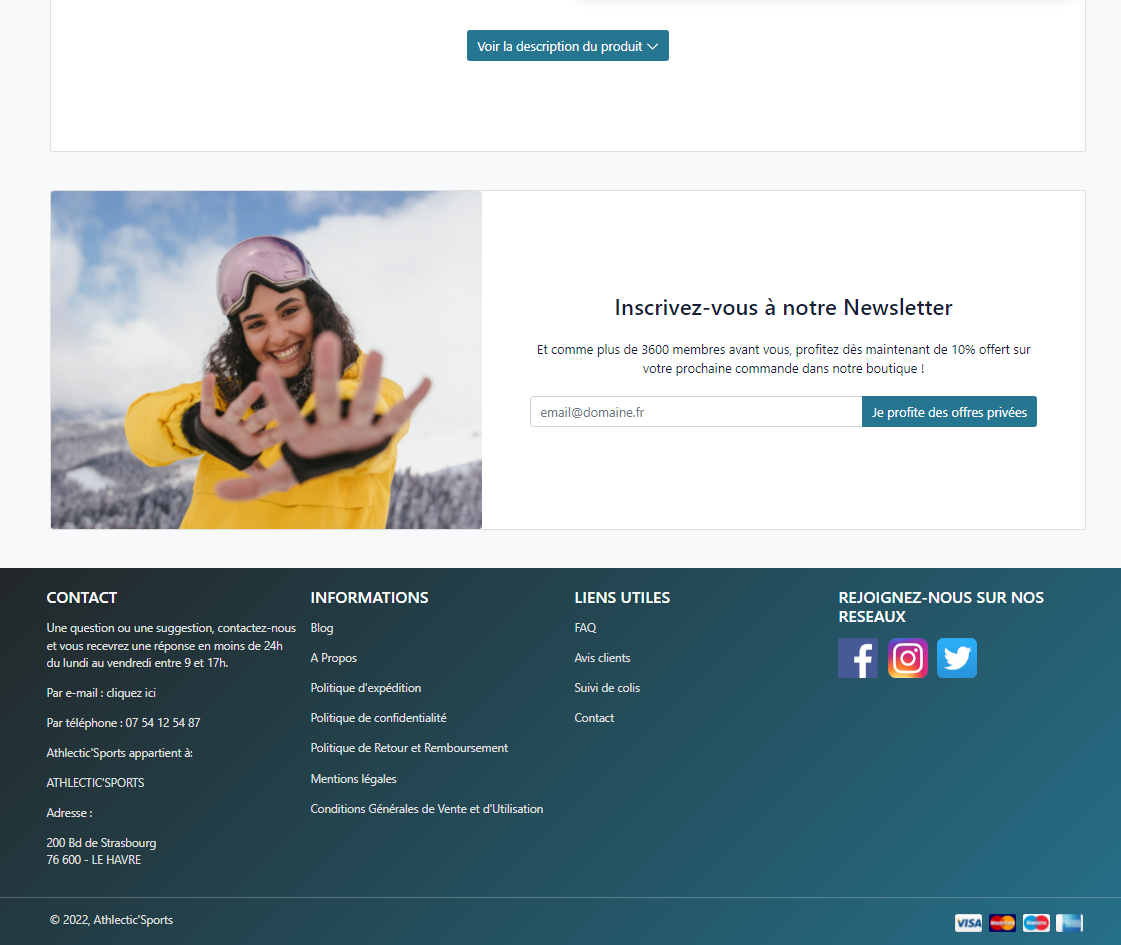
Prototypage - Figma

## 2.3 Réaliser une interface utilisateur web statique et adaptable

L’utilisation des téléphones a pris une place tellement importante dans nos vies qu’il a fallu s’adapter et concevoir des sites internet adaptés aux différents supports numériques de la vie quotidienne.

C’est pourquoi, concevoir son site en ***Mobile First*** est devenu une étape nécessaire pour continuer de recevoir des internautes sur nos sites internet.

L’utilisation de Bootstrap est un avantage indéniable dans ce processus, en utilisant les breakpoints que nous propose le Framework, nous pouvons rendre très rapidement nos applications web en mode responsive.





## 2.4 Développer une interface utilisateur web dynamique

Pour une meilleure interaction entre l’utilisateur et la plateforme marchande, mon choix d’utiliser le Javascript était une évidence, de plus, c’est un stack qui a le vent en poupe et mérite que l’on s’attarde sur ce dernier.

Cependant, il a été utilisé avec parcimonie dans le projet et notamment dans les cas suivants :

* Conception d’un panier avec Javascript.
* Contrôle du nombre de caractères entrés (formulaire contact & SEO Méta)
* Redirection lors d’un choix de tri dans la page des collections
* Zoom sur une image produit avec LightBox
* Navigation d’image en image d’un produit
* Affichage d’un toast boostrap lors de l’ajout au panier
* Affichage d’un toast SweetAlert pour la gestion d’erreur du back-office
* Prévisualisation du rendu de la SERP Google
* Prévisualisation de l’image produit en cours d’upload
* Utilisation de Jquery pour faire de l’AJAX
* Utilisation de Jquery pour la fonction « draggable » de son « User Interface » et utilisée pour le positionnement des catégories et des collections

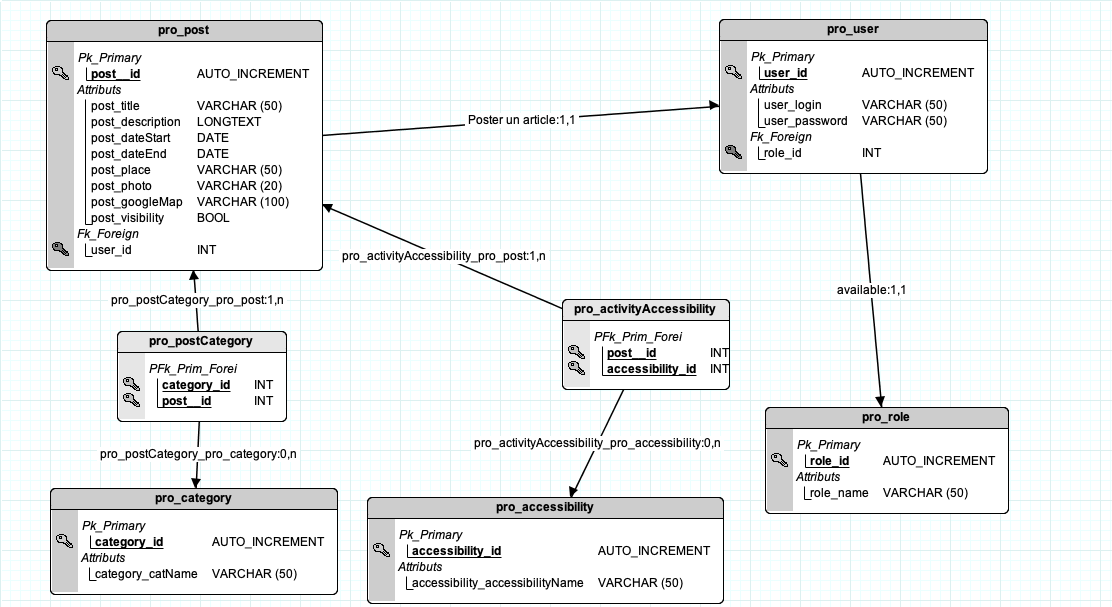
## 2.5 Créer une base de données

Une fois les différentes fonctionnalités définies et le maquettage terminé, il a été nécessaire d’établir la base de données. Elle stockera toutes les données du site web.

J’ai donc utilisé le logiciel JMerise pour modéliser mes entités, mes champs ainsi que mes cardinalités. J’ai obtenu un MCD (Modèle Conceptuel de Données).

Puis une fois mon MCD modélisé, j’ai utilisé une fonctionnalité de JMerise pour générer automatiquement mon MLD : Modèle Logique de Données. L’outil se base sur les clefs primaires et les cardinalités renseignées pour créer les clefs étrangères dans les différentes tables.

MLD :

**

Une fois mon MCD et mon MLD de créés, je me suis appuyé sur une fonction de JMerise pour générer automatiquement le script SQL correspondant à mon Merise.

x#------------------------------------------------------------

# Script MySQL.

#------------------------------------------------------------

#------------------------------------------------------------

# Table: pro\_category

#------------------------------------------------------------

CREATE TABLE pro\_category(

category\_id Int Auto\_increment NOT NULL ,

category\_catName Varchar (50) NOT NULL

,CONSTRAINT pro\_category\_PK PRIMARY KEY (category\_id)

)ENGINE=InnoDB;

CREATE TABLE pro\_accessibility (

accessibility\_id INT AUTO\_INCREMENT NOT NULL,

accessibility\_accessibilityName VARCHAR(50) NOT NULL,

CONSTRAINT pro\_accessibility\_PK PRIMARY KEY (accessibility\_id)

) ENGINE=INNODB;

CREATE TABLE pro\_role (

role\_id INT AUTO\_INCREMENT NOT NULL,

role\_name VARCHAR(50) NOT NULL,

CONSTRAINT pro\_role\_PK PRIMARY KEY (role\_id)

) ENGINE=INNODB;

CREATE TABLE pro\_user (

user\_id INT AUTO\_INCREMENT NOT NULL,

user\_login VARCHAR(50) NOT NULL,

user\_password VARCHAR(50) NOT NULL,

role\_id INT NOT NULL,

CONSTRAINT pro\_user\_PK PRIMARY KEY (user\_id),

CONSTRAINT pro\_user\_pro\_role\_FK FOREIGN KEY (role\_id)

REFERENCES pro\_role (role\_id)

) ENGINE=INNODB;

CREATE TABLE pro\_post (

post\_\_id INT AUTO\_INCREMENT NOT NULL,

post\_title VARCHAR(50) NOT NULL,

post\_description LONGTEXT NOT NULL,

post\_dateStart DATE NOT NULL,

post\_dateEnd DATE NOT NULL,

post\_place VARCHAR(50) NOT NULL,

post\_photo VARCHAR(20) NOT NULL,

post\_googleMap VARCHAR(100) NOT NULL,

post\_visibility BOOL NOT NULL,

user\_id INT NOT NULL,

CONSTRAINT pro\_post\_PK PRIMARY KEY (post\_\_id),

CONSTRAINT pro\_post\_pro\_user\_FK FOREIGN KEY (user\_id)

REFERENCES pro\_user (user\_id)

) ENGINE=INNODB;

CREATE TABLE pro\_postCategory (

category\_id INT NOT NULL,

post\_\_id INT NOT NULL,

CONSTRAINT pro\_postCategory\_PK PRIMARY KEY (category\_id , post\_\_id),

CONSTRAINT pro\_postCategory\_pro\_category\_FK FOREIGN KEY (category\_id)

REFERENCES pro\_category (category\_id),

CONSTRAINT pro\_postCategory\_pro\_post0\_FK FOREIGN KEY (post\_\_id)

REFERENCES pro\_post (post\_\_id)

) ENGINE=INNODB;

CREATE TABLE pro\_activityAccessibility (

post\_\_id INT NOT NULL,

accessibility\_id INT NOT NULL,

CONSTRAINT pro\_activityAccessibility\_PK PRIMARY KEY (post\_\_id , accessibility\_id),

CONSTRAINT pro\_activityAccessibility\_pro\_post\_FK FOREIGN KEY (post\_\_id)

REFERENCES pro\_post (post\_\_id),

CONSTRAINT pro\_activityAccessibility\_pro\_accessibility0\_FK FOREIGN KEY (accessibility\_id)

REFERENCES pro\_accessibility (accessibility\_id)

) ENGINE=INNODB;

## 2.6 Développer des composants d’accès aux données

Dans une architecture MVC, chaque modèle représente une table de la base de données. Ils doivent pouvoir s’y connecter à chaque exécution d’une méthode afin d’accéder aux différentes données.

J’ai donc créé une classe spécifique regroupant les éléments nécessaires pour se connecter à la base de données.



*La classe  database  est commune à toutes les autres classes nécessitant une connexion à la base de données*



## 2.7 Développer la partie back-end d’une application web ou web mobile

Comme indiqué précédemment, dans une architecture MVC, un Modèle correspond à une table de la base de données.

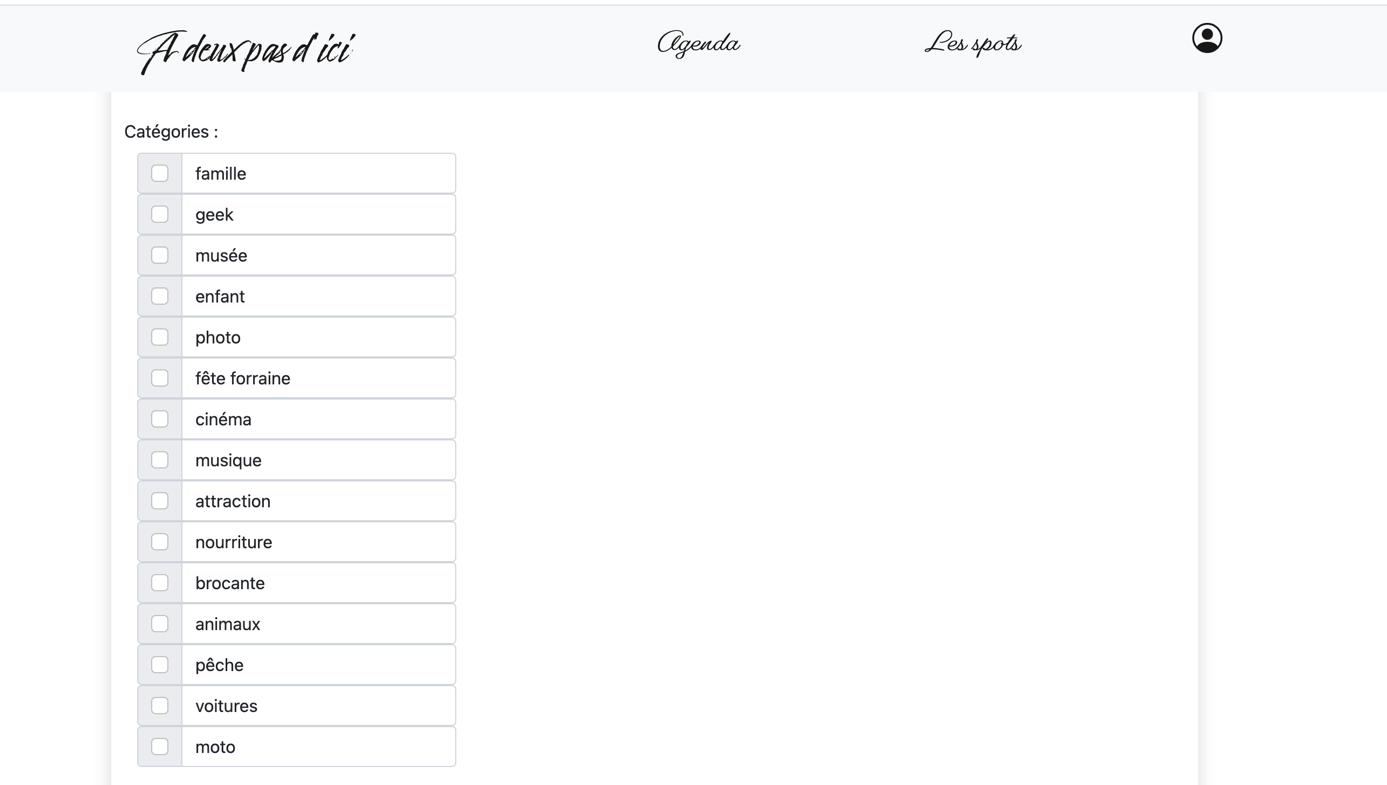
Dans notre site, les Modèles utilisent la classe « database » pour se connecter et disposent de diverses méthodes pour créer, lire, modifier et supprimer les données respectives.

Pour pouvoir les utiliser, les Modèles sont directement instanciés dans les Contrôleurs des différentes Vue.

En effet, les Vues permettent d’afficher des données et interagir avec l’utilisateur.

Pour cela, chaque Vue dispose d’un contrôleur respectif.

*Affichage d’une liste « Catégories » du formulaire selon les données de la table « Catégorie ».*

**

On inclut les Modèles et le Contrôleur via un « require »

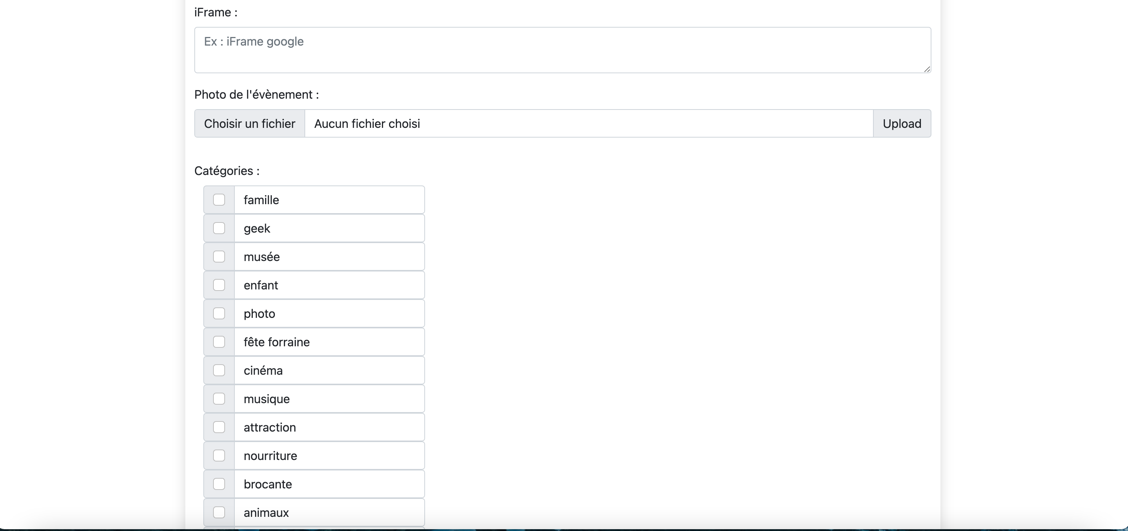
**

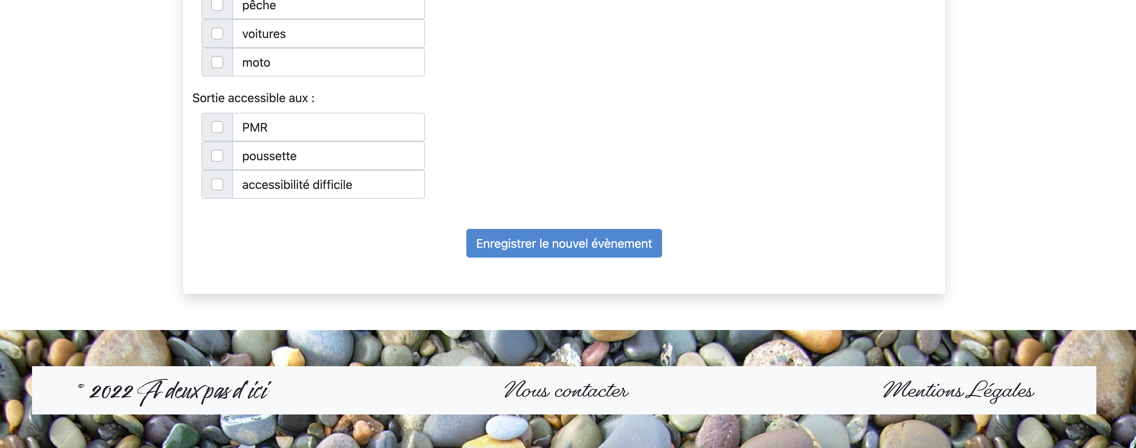
# 3. Persistance des données via une BDD

## 3.1 Création d’un évènement

Création d’un formulaire pour poster un nouvel évènement.

**

**

**

Nous faisons des contrôles via des Regex en Javascript pour vérifier que l’utilisateur respecte bien les formats des réponses du formulaire. Le fait d’utiliser Javascript permet d’avoir un affichage dynamique en cas d’erreur sur le formulaire. Toutefois, en parallèle, nous réalisons les mêmes contrôles de regex en PHP pour plus de sécurité.

Screen des regex :

**

Conditions requises lors de la validation du formulaire pour lancer la fonction du rajout du nouvel évènement :

// vérification du formulaire

if (!empty($\_POST)) {

if (isset($\_POST["title"])) {

if (empty($\_POST["title"])) {

$arrayError["title"] = "Veuillez saisir le titre de l'évènement";

}

}

if (isset($\_POST["description"])) {

if (empty($\_POST["description"])) {

$arrayError["description"] = "Veuillez saisir la description de l'évènement";

}

}

if (isset($\_POST["timeStart"])) {

if (empty($\_POST["timeStart"])) {

$arrayError["timeStart"] = "Veuillez saisir le début de l'évènement";

}

}

if (isset($\_POST["timeEnd"])) {

if (empty($\_POST["timeEnd"])) {

$arrayError["timeEnd"] = "Veuillez saisir l'horaire de fin d'évènement";

}

}

if (isset($\_POST["dateStart"])) {

if (empty($\_POST["dateStart"])) {

$arrayError["dateStart"] = "Veuillez saisir la date du début d'évènement";

} elseif (!preg\_match($regexDate, $\_POST["dateStart"])) {

$arrayError["dateStart"] = "Format invalide";

}

}

if (isset($\_POST["dateEnd"])) {

if (empty($\_POST["dateEnd"])) {

$arrayError["dateEnd"] = "Veuillez saisir la date de fin d'évènement";

} elseif (!preg\_match($regexDate, $\_POST["dateEnd"])) {

$arrayError["dateEnd"] = "Format invalide";

}

}

if (isset($\_POST["place"])) {

if (empty($\_POST["place"])) {

$arrayError["place"] = "Veuillez saisir le lieu de l'évènement";

}

}

if (isset($\_POST["iFrame"])) {

if (empty($\_POST["iFrame"])) {

$arrayError["iFrame"] = "Veuillez saisir l'iFrame google de l'évènement";

}

}

if (!isset($\_POST["category"])) {

$arrayError["category"] = "Veuillez selectionner le(s) thème(s) de l'évènement";

}

if (!isset($\_POST["accessibility"])) {

$arrayError["accessibility"] = "Veuillez selectionner l'accessibilité de l'évènement";

}

// upload image

$arrayExt = ['jpg', 'jpeg', 'png', 'gif'];

$arrayMsg = [];

$maxSizeFile = 1000000;

$id = uniqid();

if (isset($\_FILES['fileToUpload'])) {

$tmpName = $\_FILES['fileToUpload']['tmp\_name'];

$name = $\_FILES['fileToUpload']['name'];

$size = $\_FILES['fileToUpload']['size'];

$type = $\_FILES['fileToUpload']['type'];

$error = $\_FILES['fileToUpload']['error'];

$ext = substr(strrchr($type, '/'), 1);

if ($error == 0) {

$typeImg = getimagesize($tmpName);

if ($typeImg !== false && in\_array($ext, $arrayExt)) {

if ($size < $maxSizeFile) {

$namePicture = $id . '-' . $name;

move\_uploaded\_file($tmpName, "../assets/imgAgenda/$namePicture");

// $arrayError["picture"] = 'Fichier ajouté';

} else {

$arrayError["picture"] = 'Fichier trop volumineux. Veuillez en choisir un autre.';

}

} else {

$arrayError["picture"] = 'Format de fichier non valide. Veuillez en choisir un autre.';

}

} else {

$arrayError["picture"] = "Veuillez inclure la photo de l'évènement";

}

}

Méthode permettant de rajouter le nouvel évènement dans la base de données :

/\*\*

\* Permet d'ajouter un évènement

\* @param string $title : titre de l'évènement

\* @param string $description : description de l'évènement

\* @param string $timeStart : heure du début de l'évènement

\* @param string $dateStart : date de depart de l'évènement

\* @param string $dateEnd : date de fin de l'évènement

\* @param string $place : lieu de l'évènement

\* @param string $iFrame : googleMap de l'évènement

\* @param string $picture : photo de l'évènement

\* @param string $idUser : id de l'utilisteur

\* @param string $category : catégorie de l'évènement

\* @param string $accessibility : accessibilité de l'évènement

\*/

public function addPost(string $title, string $description, string $timeStart, string $dateStart, string $dateEnd, string $place, string $iFrame, string $picture, int $idUser): int

{

$db = $this->connectDb();

$sql = "INSERT INTO `pro\_post` (`post\_title`, `post\_description`, `post\_timeStart`, `post\_dateStart`, `post\_dateEnd`, `post\_place`,`post\_googleMap`, `post\_photo`,

`post\_visibility`, `user\_id`)

VALUES ( :title, :description, :timeStart, :dateStart, :dateEnd, :place, :iFrame, :picture, :visibility, :user\_id )";

$query = $db->prepare($sql);

// je lie les valeurs des paramètres récupérés (ex. $title) à mes marqueurs nominatifs (ex. :title).

$query->bindValue(':title', $title, PDO::PARAM\_STR);

$query->bindValue(':description', $description, PDO::PARAM\_STR);

$query->bindValue(':timeStart', $timeStart, PDO::PARAM\_STR);

$query->bindValue(':dateStart', $dateStart, PDO::PARAM\_STR);

$query->bindValue(':dateEnd', $dateEnd, PDO::PARAM\_STR);

$query->bindValue(':place', $place, PDO::PARAM\_STR);

$query->bindValue(':iFrame', $iFrame, PDO::PARAM\_STR);

$query->bindValue(':picture', $picture, PDO::PARAM\_STR);

// visibility = 1 car uniquement posté par Admin

$query->bindValue(':visibility', 1, PDO::PARAM\_INT);

$query->bindValue(':user\_id', $idUser, PDO::PARAM\_INT);

$query->execute();

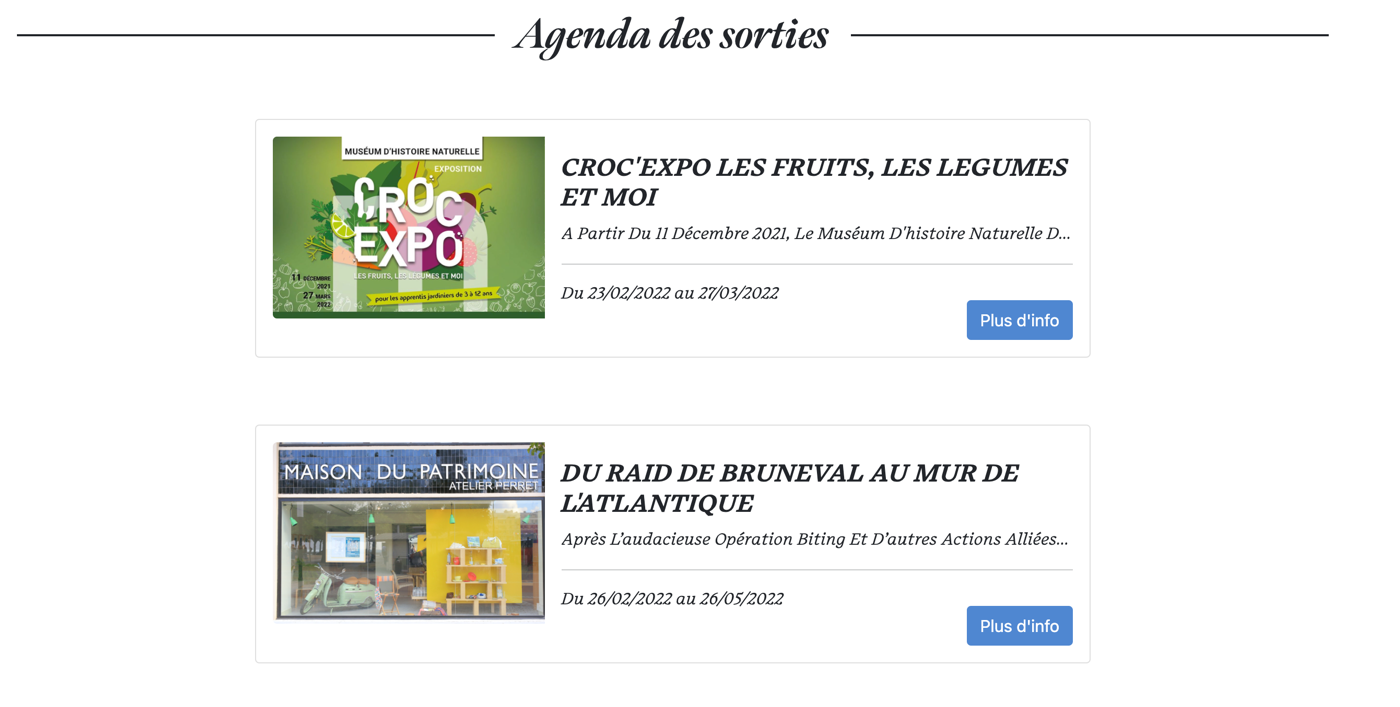
return $db->lastInsertId();

}

## 3.2 Affichage des données de l’utilisateur

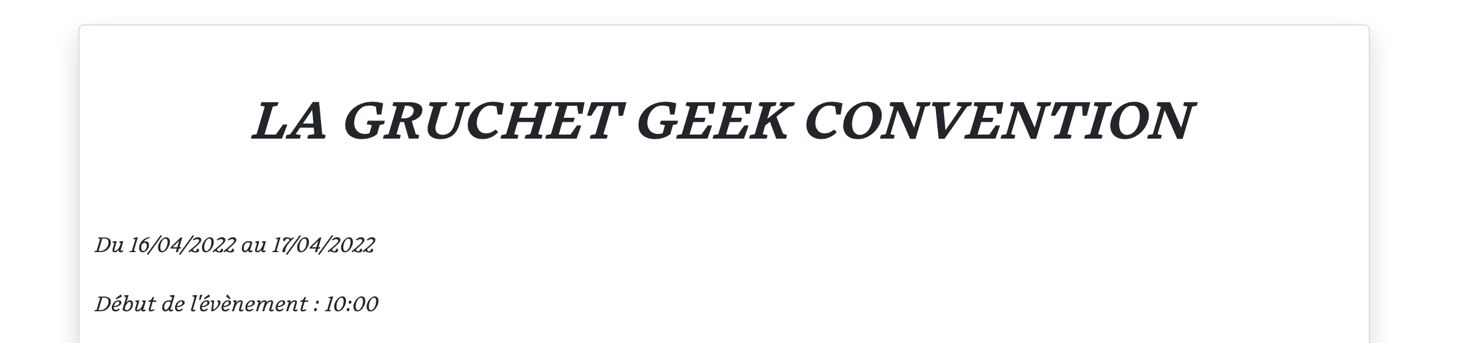
Une fois l’évènement créé, nous allons pouvoir retrouver les données liées à l’évènement dans la section « Agenda ». Cette page permet à la fois de consulter un aperçu des informations et de cliquer sur « plus d’info » et afficher l’évènement en détail.

*Aperçu des informations*

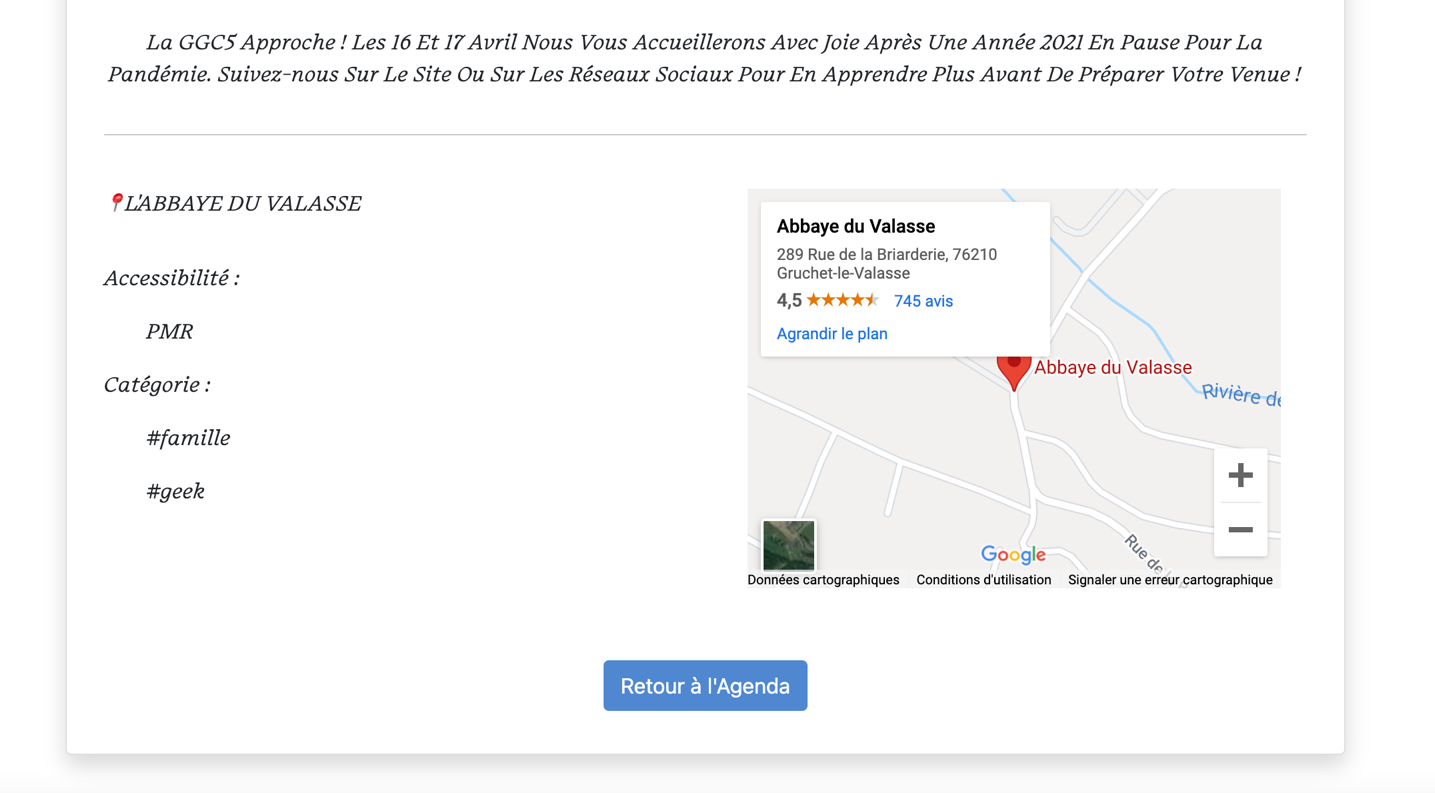




Évènement complet

**

**

**

Fonction permettant d’afficher l’aperçu des données de l’évènement :

/\*\*

\* Permet de poster dans la page Agenda

\* @return array : tableau associatif contenant les infos des posts

\*/

public function getAllPost(): array

{

$base = $this->connectDb();

$sql = "SELECT `post\_id`, `post\_title`, `post\_description`, `post\_timeStart`, DATE\_FORMAT(`post\_dateStart`, '%d/%m/%Y') AS `post\_dateStartAll`, DATE\_FORMAT(`post\_dateEnd`, '%d/%m/%Y') AS `post\_dateEnd`, `post\_place`, `post\_photo`, `post\_googleMap`, `post\_visibility`, `user\_id` FROM `pro\_post` ORDER BY `post\_dateStart`;";

$resultQuery = $base->query($sql);

return $resultQuery->fetchAll();

}

Fonction permettant d’afficher les données complète de l’évènement :

/\*\*

\* Permet de recuperer toutes les informations de l'évènement selon un id

\*

\*/

public function getOnePost($id)

{

$base = $this->connectDb();

$sql = "SELECT `post\_id`, `post\_title`, `post\_description`, `post\_timeStart`, DATE\_FORMAT(`post\_dateStart`, '%d/%m/%Y') AS post\_dateStart, DATE\_FORMAT(`post\_dateEnd`, '%d/%m/%Y') AS `post\_dateEnd`, `post\_place`, `post\_photo`, `post\_googleMap`, `post\_visibility`, user\_id FROM `pro\_post` WHERE `post\_id`= :id";

$resultQuery = $base->prepare($sql);

$resultQuery->bindValue(':id', $id, PDO::PARAM\_INT);

$resultQuery->execute();

return $resultQuery->fetch();

}

Screen de la page HTML affichant les résultats de la fonction d’affichage. Sur cette page nous affichons à la fois les données présentent en base

<div class="container mt-5 mb-3 pt-3 pb-3 shadow">

<div class="row justify-content-center mb-5">

<div class="card col-lg-0 widthBigCard mt-5 shadow">

<h1 class="card-title mt-5 mb-5 text-center titleCard"><?= $postArray['post\_title'] ?></h1>

<p class="card-text mt-2 textIntro">Du <?= $postArray['post\_dateStart'] ?> au <?= $postArray['post\_dateEnd'] ?></p>

<p class="card-text mt-1 textIntro">Début de l'évènement : <?= $postArray['post\_timeStart'] ?></p>

<img src="../assets/imgAgenda/<?= $postArray['post\_photo'] ?>" class="card-img-top" alt="...">

<div class="card-body">

<p class="card-text mt-5 pb-3 text-center textIntro"><?= $postArray['post\_description'] ?></p>

<HR class="mt-2 mb-4">

<div>

<div class="row">

<div class="col">

<p class="card-text mt-3 pb-3 texte-center textIntro">📍<?= $postArray['post\_place'] ?></p>

<div>

<p class="card-text mt-3 textIntro">Accessibilité : </p>

<?php foreach ($arrayAccessibility as $access) { ?>

<ul class="card-text mt-3 textIntro"><?= $access ?></ul>

<?php } ?>

<p class="card-text mt-3 textIntro">Catégorie : </p>

<?php foreach ($arrayCategory as $cat) { ?>

<ul class="card-text mt-3 textIntro">#<?= $cat ?></ul>

<?php } ?>

</div>

</div>

<div class=col>

<p class="card-text mt-3 text-center"><?= $postArray['post\_googleMap'] ?></p>

</div>

</div>

<div class="text-center">

<a type="submit" href="agenda.php" value="Retour" name="btn-submit" class="btn blueButton text-white mb-3 mt-5">Retour à l'Agenda</a>

</div>

</div>

</div>

</div>

</div>

</div>

## 3.3 Modification des évènements

Sur la page « PanelAdmin », nous allons pouvoir changer les données relatives à l’évènement par le biais du bouton « Plus d’info » puis « modifier l’évènement ».

*Bouton « plus d’info » dans le panel Admin*



*Bouton « modifier l’évènement »*

**

*Méthode permettant de modifier un évènement :*

// this = réfere a l'objet actuel

public function modifyEvenement(int $id, string $title, string $description, string $timeStart, string $dateStart, string $dateEnd, string $place, string $iFrame, string $picture, int $idUser, array $category, array $accessibility): void

{

$base = $this->connectDb();

$sql =

"UPDATE `pro\_post` SET post\_title = :title, post\_description = :description, post\_timeStart=:timeStart, post\_dateStart = :dateStart, post\_dateEnd = :dateEnd, post\_place = :place, post\_googleMap = :iFrame, post\_photo = :picture, post\_visibility = :visibility, user\_id= :user\_id

WHERE post\_id = :id;";

$resultQuery = $base->prepare($sql);

$resultQuery->bindValue(':id', $id, PDO::PARAM\_INT);

$resultQuery->bindValue(':title', $title, PDO::PARAM\_STR);

$resultQuery->bindValue(':description', $description, PDO::PARAM\_STR);

$resultQuery->bindValue(':timeStart', $timeStart, PDO::PARAM\_STR);

$resultQuery->bindValue(':dateStart', $dateStart, PDO::PARAM\_STR);

$resultQuery->bindValue(':dateEnd', $dateEnd, PDO::PARAM\_STR);

$resultQuery->bindValue(':place', $place, PDO::PARAM\_STR);

$resultQuery->bindValue(':iFrame', $iFrame, PDO::PARAM\_STR);

$resultQuery->bindValue(':picture', $picture, PDO::PARAM\_STR);

// visibility = 1 car uniquement posté par Admin

$resultQuery->bindValue(':visibility', 1, PDO::PARAM\_INT);

$resultQuery->bindValue(':user\_id', $idUser, PDO::PARAM\_INT);

$resultQuery->execute();

}

*Controller permettant de modifier un évènement :*

// si aucune erreur envoie le formulaire vers la BDD

if (count($arrayError) == 0) {

// strtoupper = en majuscule / ucwords = 1ere lettre en majuscule

$id = htmlspecialchars(trim($\_POST['idEvent']));

$title = htmlspecialchars(strtoupper(trim($\_POST['title'])));

$description = htmlspecialchars(ucwords(trim($\_POST['description'])));

$timeStart = htmlspecialchars(trim($\_POST['timeStart']));

$dateStart = htmlspecialchars(trim($\_POST['dateStart']));

$dateEnd = htmlspecialchars(trim($\_POST['dateEnd']));

$place = htmlspecialchars(trim($\_POST['place']));

$iFrame = $\_POST['iFrame'];

$picture = htmlspecialchars(trim($namePicture));

$category = $\_POST['category'];

$accessibility = $\_POST['accessibility'];

// actuellement seul l'admin peut poster et dispose de l'id 1

$idUser = 1;

$postObj = new Post();

$postObj->modifyEvenement($id, $title, $description, $timeStart, $dateStart, $dateEnd, $place, $iFrame, $picture, $idUser, $category, $accessibility);

$postCatObj = new PostCategory();

$postCatArray = $postCatObj->deleteEventCategory($id);

$postCatObj = new ActivityAccessibility();

$postCatArray = $postCatObj->deleteEventAccessibility($id);

// Utilisation d'une boucle pour enregistrer toutes les catégories de l'événements récupérée dans le formulaire : Méthode setCategory avec l'id de événement et la catégory

$categoryObj = new PostCategory;

foreach ($category as $value) {

$categoryObj->setCategory($value, $id);

}

// Utilisation d'une boucle pour enregistrer toutes les accessibilités de l'événements récupérée dans le formulaire : Méthode setAccessibility avec l'id de événement et l'accessibilité'

$accessibilityObj = new ActivityAccessibility;

foreach ($accessibility as $value) {

$accessibilityObj->setAccessibility($value, $id);

}

// redirige l'utilisateur vers gestion agenda après la modification d'un évènement par le bias du formulaire

$\_SESSION['toast']['modify'] = 1;

header("Location: gestionAgenda.php");

}

## 3.4 Suppression d’un compte utilisateur

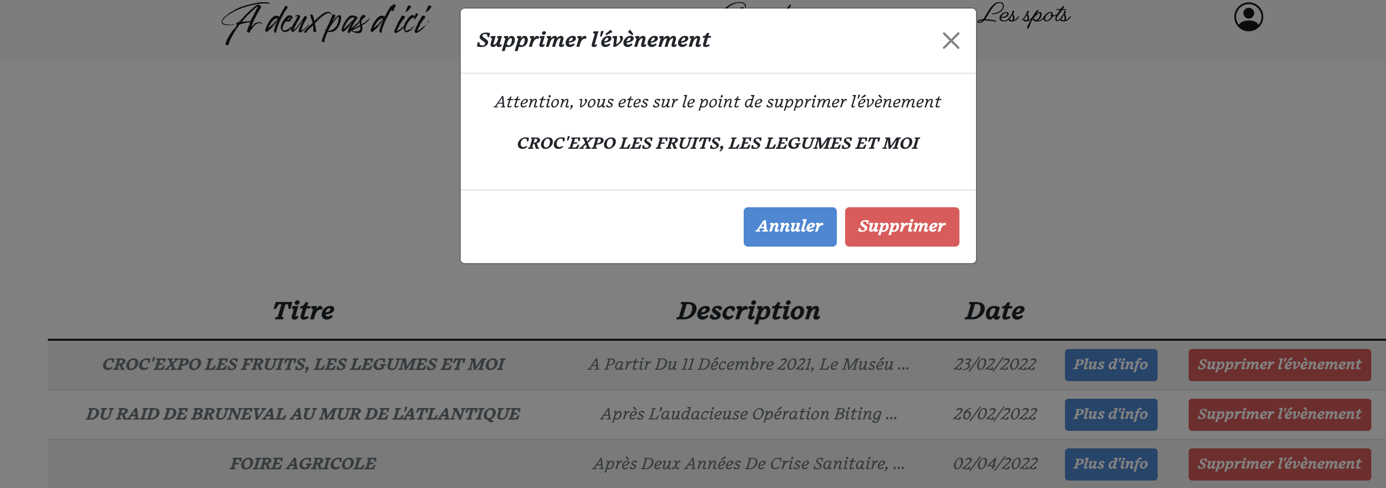
Pour effectuer la suppression de l’évènement, l’administrateur doit cliquer dans « panelAdmin » sur le bouton « supprimer l’évènement ».

Une modal apparait pour demander de confirmer la suppression.

Vue du panel Admin :



Vue de la modal :



Méthode permettant de supprimer l’évènement :

/\*\*

\* Permet de supprimer un évènement par le bias de la page gestionAgenda

\* @param $id : ex.5

\*

\*/

public function deleteEvenement($id): void

{

$base = $this->connectDb();

$sql = "DELETE FROM `pro\_post` WHERE `post\_id` = :id";

$resultQuery = $base->prepare($sql);

$resultQuery->bindValue(':id', $id, PDO::PARAM\_INT);

$resultQuery->execute();

}

*Controller permettant de supprimer un évènement :*

// suppression d'un évènement

$deleteEvenementOk = 0;

if (isset($\_POST["deleteEvenement"])) {

$id = htmlspecialchars(trim($\_POST["idDeleteEvenement"]));

$postObj = new Post();

$postDelete = $postObj->deleteEvenement($id);

$deleteEvenementOk = 1;

}

# 4. Documentation en anglais

Durant mon projet pro, j'ai dû rechercher des informations dans la documentation ou sur Stack

Overflow ainsi que des forums comme Stack Overflow , j'ai donc été amené à traduire de moi-même

les textes en anglais, comme le texte reproduit ci-dessous :

Responsive breakpoints

Since Bootstrap is developed to be mobile first, we use a handful of media queries to create sensible breakpoints for our layouts and interfaces.

These breakpoints are mostly based on minimum viewport widths and allow us to scale up elements as the viewport changes.

Points de ruptures responsive

Puisque Bootstrap est développé pour être « mobile first », nous utilisons une poignée de « media queries » pour créer des points de ruptures sensibles pour nos mises en page et nos interfaces.

Ces points d'arrêt sont principalement basés sur des largeurs minimales de fenêtre et nous permettent d’agrandir les éléments à mesure que la fenêtre change.

# 5. Conclusion

Le projet « A deux pas d’ici » est la première étape en adéquation avec ma formation

Le projet, bien que fonctionnel, est encore à ses débuts.

De nouvelles fonctionnalités sont en cours de réflexion afin d'améliorer la navigation des utilisateurs sur le site et seront ajoutées au fur et à mesure.

J’envisage également de créer un filtre permettant de sélectionner des évènements par catégorie.

Dans un second temps, le développement d'une interface pour permettre aux utilisateurs de créer un compte, de pouvoir se connecter et d’avoir la possibilité de poster un évènement que l’administrateur ou le modérateur validera ou non si l’évènement respect la charte du site.

Cela permettra aux utilisateurs d’avoir encore plus de choix pour des sorties.

Au cours de ce projet, j'ai pu me rendre compte du temps que demande la construction d'un site de cette ampleur et de l'investissement nécessaire pour la bonne mise en exécution, tant de la phase de conception, que de la réalisation et du débogage.

J'ai beaucoup aimé travailler sur ce projet qui reste encore ouvert à de nombreuses perspectives d'évolution et d'enrichissement.