

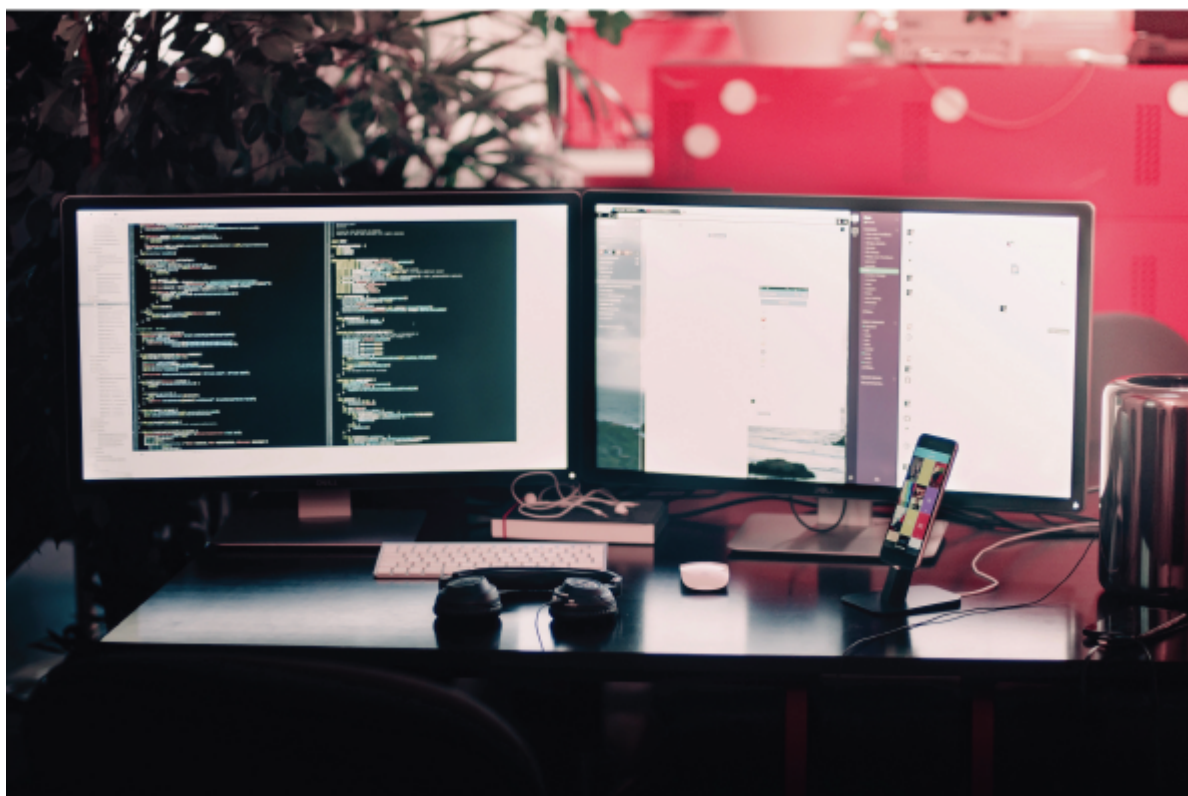
# **GAUTHIER** *Baptiste*

---

## Dossier de projet professionnel

*Création d'une boutique en ligne  
Projet Playground du Sud*

**Titre : Développeur Web et Web Mobile**



## Sommaire

Liste des compétences du référentiel.....	3
---	---

### Projet Boutique en ligne :

- Présentation du projet .....	3
- Spécifications fonctionnelles.....	4
- Fonctionnalités du projet.....	5
- Spécifications techniques.....	7
- Technologies utilisées	
- Structure du projet	
- Organisation du travail	
- Conception de la base de données	
- Design et Maquette	
- Réalisation et extraits de code.....	12
- Exemple du système de tri	
- Veille sur les vulnérabilités de sécurité.....	14

### Projet Playgrounds du Sud

- Présentation du projet.....	18
- Réalisation et extraits de code	
- Placement des ballons sur la map	
- Appel ajax pour la connexion	
- Organisation du travail	
- Recherche à partir d'un site anglophone	

Annexes.....	27
--------------	----

## Liste des compétences du référentiel couvertes par les projets :

Les projets couvrent les compétences ci-dessous :

Concernant la partie front-end : ***"Développer la partie front-end d'une application web et web mobile en intégrant les recommandations de sécurité"*** :

- Maquetter une application
- Réaliser une interface utilisateur web statique et adaptable
- Développer une interface utilisateur web dynamique
- Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce

Concernant la partie back-end : ***"Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité"*** :

- Créer une base de données
- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile
- Elaborer et mettre en œuvre des composants dans une application de gestion de contenu

*Présentation du projet boutique en ligne :*

Le projet est une boutique en ligne de vente de matériel de tennis. On peut donc y trouver des raquettes, des balles, des sacs, des cordages et des accessoires ( anti-vibrateur et grip) de différentes marques. Ce site a une partie utilisateur avec un système de connexion et d'inscription permettant de parcourir le catalogue et d'effectuer une commande.

Il y a également une partie administrateur permettant de modifier les articles et d'en insérer de nouveaux. La page index est attractive et met en avant certains articles. On trouve une sélection de produits mis en avant, un carrousel publicitaire et un texte de présentation. Enfin, une partie panier, commande et paiement est également intégrée.

Etant moi-même joueur de tennis depuis plusieurs années, je connais bien les besoins demandés par les joueurs. C'est donc tout naturellement que j'ai choisi ce thème pour ma boutique. Je peux alors facilement me mettre à la place d'un client et créer une interface web adaptée et répondre aux problématiques des futurs clients.

## Spécifications fonctionnelles :

### Périmètre du projet :

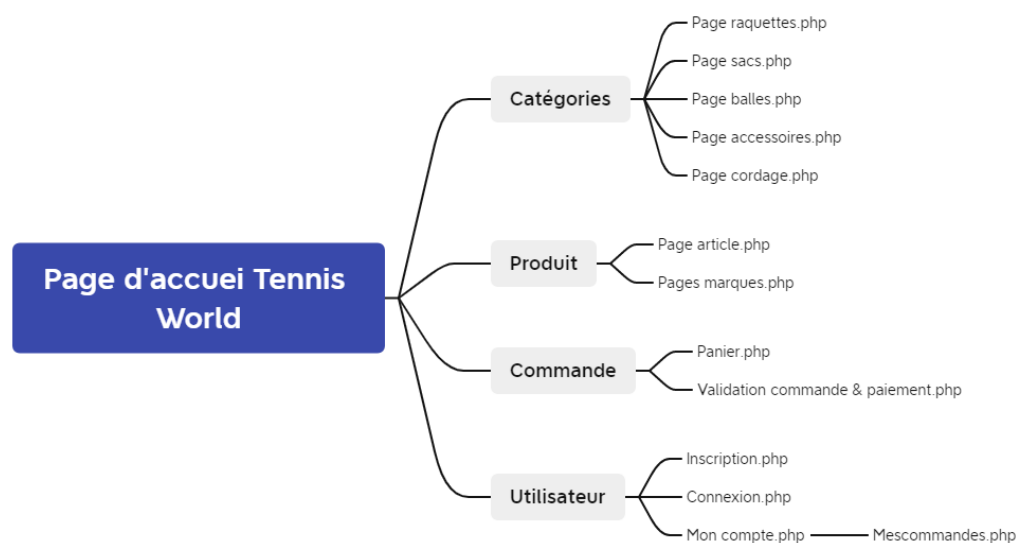
Le but de ce projet est dans un premier temps de satisfaire les acheteurs en proposant une large palette d'articles pour tous les goûts. Nous ciblons en priorité les joueurs, mais aussi toutes les personnes souhaitant des informations sur les produits. Il faut donc créer une interface claire et compréhensible de tous pour avoir la meilleure expérience utilisateur possible.

Le site devra également être adapté aux formats mobile et tablette pour réserver le meilleur accueil possible à notre client.

### Arborescence du site

Le site web est composé de plusieurs page :

- Une page d'accueil
- Une page inscription
- Une page connexion
- Une page mon compte
- Une page regroupant tous les produits d'une catégorie
- Une page produit
- Une page panier
- Une page commande
- Une page marque
- Un espace administrateur



## Description des fonctionnalités du projet

### Module Inscription / Connexion

Pour s'inscrire l'utilisateur passe par un formulaire d'inscription classique avec plusieurs informations : le nom, le prénom, adresse, téléphone, rue, code postal, mail et un mot de passe. Une fois l'inscription réussie l'utilisateur est ensuite dirigé vers la page de connexion.

Pour la connexion c'est la même chose: on demande à l'utilisateur d'entrer son email et son mot de passe dans un formulaire. Un compte est obligatoire pour pouvoir ajouter un article au panier et passer une commande.

### Espace client

L'utilisateur doit avoir un espace client pour accéder à toutes ses informations et pouvoir les modifier quand il le souhaite. Via un formulaire, il peut modifier toutes ses informations et changer son mot de passe après une vérification.

Ses commandes sont également affichées. Le client a accès à la liste de commandes et de toutes ses informations : le numéro de la commande, le nom de l'article, sa quantité, son prix et le prix total de la commande.

### Système de tri

Sur chaque page catégories il y aura un affichage de tous les produits concernant la catégorie en question. On trouvera des vignettes avec l'image principale du produit, son nom, une courte description et son prix.

L'utilisateur pourra faire un tri par marques et par prix (croissant / décroissant). Selon les catégories les tris ne seront pas les mêmes. Sur la page raquette par exemple, l'utilisateur pourra trier toutes les raquettes en fonction de la taille du manche, alors que pour un sac il pourra choisir sa capacité.

### Une page produit

Lors du clic sur une vignette l'utilisateur arrive sur la page produit.

Plusieurs informations sont affichées. Il y a le nom, la marque, la description, son prix, sa photo principale puis une à trois images secondaires.

De nouveau, selon la catégorie du produit, les informations diffèrent. La raquette aura un poids, un tamis et une taille de manche tandis qu'un grip aura une épaisseur et une couleur.

Via cette page, l'utilisateur pourra ajouter le produit à son panier. Des produits de mêmes catégories seront proposées dans une section en bas de la page.

Si l'utilisateur clique sur la marque, il pourra accéder à une page affichant tous les produits de cette marque, toutes catégories confondues.

### Barre de recherche

L'utilisateur peut effectuer une recherche via une barre de recherche intégrée dans la barre de navigation. Cela affichera les vignettes des articles correspondants à sa recherche.

### Une page panier

La page panier possède plusieurs fonctionnalités :

- Dès lors que l'utilisateur ajoute un produit via la page produit, une nouvelle ligne est affichée sur la page, reprenant l'image principale, le prix et la quantité. Si un utilisateur ajoute de nouveau ce produit à son panier la quantité augmentera de 1.
- Sur cette ligne plusieurs options sont disponibles. L'utilisateur peut augmenter ou diminuer la quantité d'un produit. Il peut également le supprimer du panier.
- Après chaque manipulation de l'utilisateur, le prix total est de nouveau calculé automatiquement.
- Au bas de la page il y a un bouton de validation pour accéder à la page paiement.
- Un compteur est également présent dans la barre de navigation sur le logo du panier.

### Une page de paiement

La page paiement regroupe toutes les informations du panier(nom, prix, quantité, prix total) et les informations du client (adresse, nom, tel, mail).

Il est possible de modifier son panier en cas d'erreur, grâce à un lien qui renvoie au panier.

Un bouton paypal est intégré à la page pour simuler le paiement. Une fois les identifiants paypal Sandbox rentrer, on effectue le paiement paypal. Une fois le paiement terminé, l'utilisateur est dirigé vers l'accueil. Sa commande apparaît désormais dans l'espace Mon compte.

### Une partie administrateur

- Gestion utilisateur : l'administrateur a la possibilité de modifier les informations d'un utilisateur, et de modifier son statut (client ou administrateur)
- Gestion du site : L'administrateur peut ajouter un nouveau produit à la boutique. A l'aide d'un formulaire adapté selon la catégorie de l'article, il peut ajouter toutes les informations et choisir les images.

Il a également accès à la liste des produits et peut en modifier un en particulier.  
Une barre de recherche est présente pour rechercher un produit.

Il peut également supprimer un produit. Les catégories et les sous catégories peuvent également être modifiées.

#### Une barre de navigation :

La barre de navigation permet à l'utilisateur de naviguer au sein du site. Elle s'adapte selon si l'utilisateur est connecté ou non. Elle se transforme en menu déroulant au format tablette et mobile.

## **Spécifications techniques du projet**

#### Technologies utilisés:

Pour la partie back-end du projet j'ai utilisé le langage PHP.  
Base de données SQL via PhpMyAdmin

Pour la partie front-end :

Tout le projet à été réalisé avec HTML / CSS. Pour le style j'ai utilisé le préprocesseur SASS permettant de compiler des fichiers css. Il ajoute plusieurs fonctionnalités et aide dans l'organisation du code.

Comme outil de versionning j'ai utilisé GIT et l'application GitHub Desktop.

Mon éditeur de texte est Visual Studio Code, accompagné des extensions : Live Sass Compiler, Live Server, Php Intelephense, php DocBlocker.

#### Utilisation de Git

Dans un premier temps, j'ai défini la base de mon projet avec tous ses fichiers. Une fois la base terminée j'ai défini un module précis à développer. Je crée alors une nouvelle branche, dupliquée depuis le master pour ma fonctionnalité. Une fois la fonctionnalité terminée, je crée une nouvelle branche qui fera office de branche de merge. Si le merge s'est bien passé, alors je peux alors merge avec le master. Pour le prochain module je vais dupliquer le nouveau master.

Je réalise alors cette opération tout au long du projet. Créer une branche de merge permet d'avoir une sécurité, avant de merge sur le master.

#### Structure du projet :

#### **Design pattern MVC :**

Dans le cadre du projet de la boutique j'ai décidé d'organiser mon code en utilisant la programmation orientée objet en php, ainsi que le design pattern MVC.

Mon projet est donc composé d'un dossier Model qui regroupe toutes mes classes Model. Ces différentes classes ont un rôle précis. Ce sont leurs méthodes qui vont communiquer avec la base de données. Elles vont principalement nous renvoyer des tableaux php en réponse de nos requêtes SQL, insérer, modifier ou supprimer des éléments en base de données.

J'ai également un dossier Vue qui va contenir toutes mes classes qui gèrent l'affichage des pages. On y trouve principalement du code HTML et des boucles php qui génèrent le code html selon ce que le modèle nous renvoie.

Et enfin le dossier Controller. Il permet de faire le lien entre le modèle et la vue. C'est lui qui va appeler le bon modèle selon certaines conditions, puis va le transmettre à la vue pour générer le bon affichage pour le client.

### **Utilisation de l'héritage :**

Grâce à la programmation orientée objet, j'ai utilisé dans quelque cas l'héritage de classes. J'ai dans mon dossier Model, une classe Model "mère", qui prend dans son constructeur la connexion à la base de données.

Ainsi toutes les autres classes Model héritent de celle-ci, ce qui leur permet d'hériter de son constructeur. Dès lors de l'instance d'un Model, la connexion à la base de données est lancée.

### **Documentation des fonctions**

Sur ce projet j'ai beaucoup documenté les méthodes de mes classes grâce à l'extension Php DocBlocker de VScode. Cela permet d'ajouter une description à la méthode, de savoir quels sont le type de variable des paramètres et ce que retourne la fonction. Cela m'a permis d'avoir une bonne organisation au niveau de mon code et de me replonger rapidement dans le projet quelques jours plus tard.

### **Autres dossiers**

Après les dossiers pour le MVC on retrouve dans le projet les dossiers suivants :

Un dossier pages où l'on retrouve tous les fichiers php avec les instances de nos objets. Ce sont ces mêmes pages sur lesquelles navigue l'utilisateur.

Un dossier script où sont placés les fichiers javascript. Sur ce projet, il y a seulement le fichier javascript pour le paiement paypal.

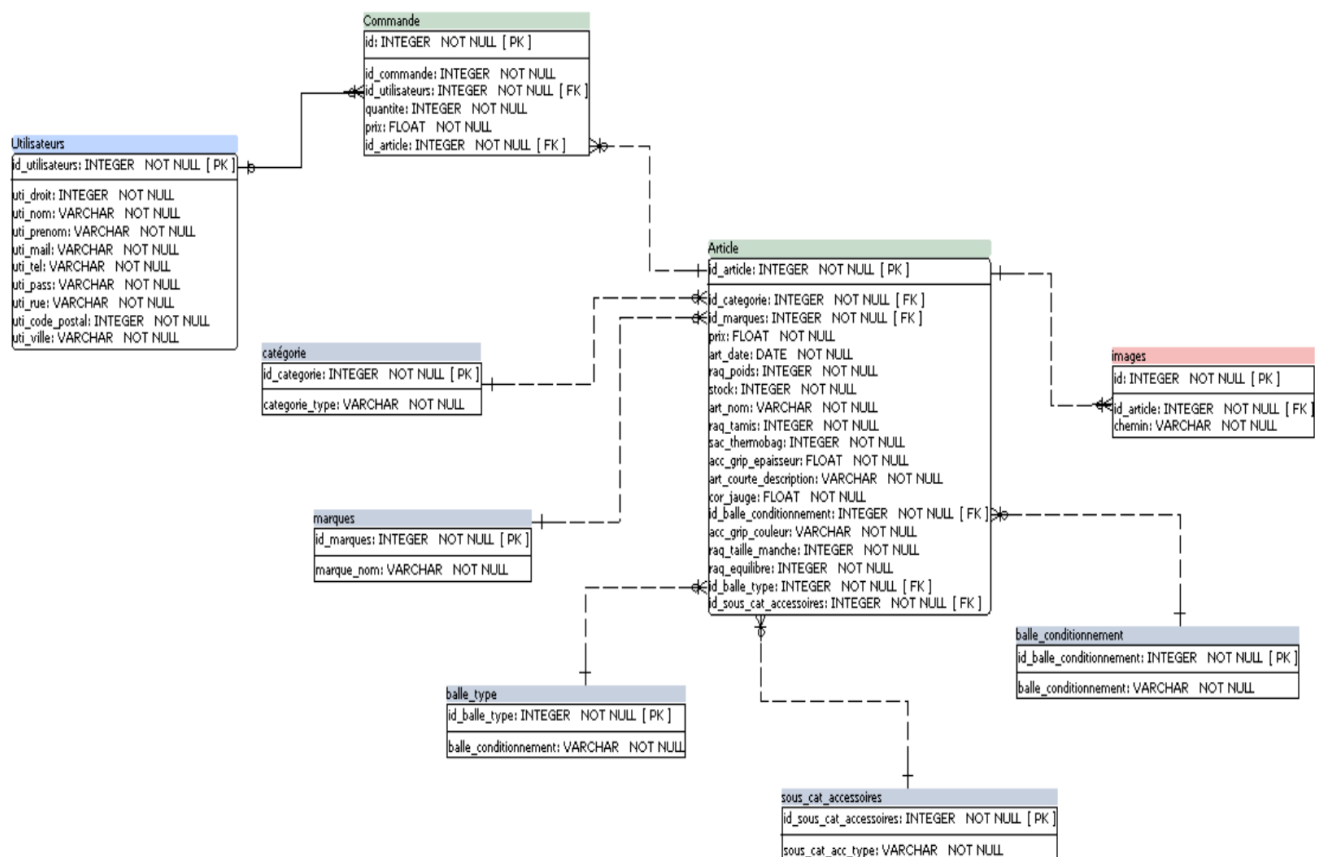
Un autre dossier nommé utils. On y retrouve à l'intérieur le fichier autoload.php.



L'autoload va directement générer des “*require\_once*” de tous les fichiers classe dont on a besoin pour chaque instance.

Un dossier sources, où l’on va stocker toutes les images et tous les fichiers CSS du projet.

### Conception de la base de donnée :



Concernant la base de données, j’ai utilisé le logiciel Looping pour créer un modèle conceptuel de données (voir en annexe) et SQL Power Architect pour le modèle physique de données.

Comme le montre l’image ci-dessus, j’ai fait le choix de réunir les caractéristiques de tous les produits dans une seule et même table. Lors de l’insertion de nouveaux produits, on aura donc certaines colonnes qui auront la valeur *NULL* en fonction de la catégorie de l’article.

J’ai également décidé de séparer les images dans une table à part. Un article peut avoir entre une et quatre images. L’image de l’article est renommée par rapport à l’id article. L’article 12 par exemple aura les images 12-0.jpg en image principale, 12-1.jpg en deuxième, 12-2.jpg en troisième etc. Il est donc plus facile par la suite d’appeler les images du produit, car elles sont toutes réunies dans la même table.

Mettre les marques et les catégories dans leurs tables respectives permet aussi d'en ajouter plus facilement à l'avenir sans poser de problèmes.

La base de données à était hébergée par la suite sur phpMyAdmin.

## Design & Maquette

Avant de commencer j'ai dans un premier temps fait une étude de l'existant. Cela m'a permis de me faire une idée du contenu du site mais aussi d'avoir un aperçu du design des différents sites.

Je me suis également également inspiré des sites tels que Behance et Dribble qui regroupent plusieurs réalisations de webdesigner.

J'ai donc décidé d'utiliser la font Poppins à l'aide google font.

En light , regular , **bold** , *italic*

Elle remplit parfaitement les critères : une typographie très lisible, avec une graisse importante, le rendu en italique donne un effet dynamique, parfait pour un site e-commerce de sport.

Pour les couleurs mon choix s'est porté sur :



Des couleurs vives pour rappeler les valeurs du sport et un vert proche du jaune qui fait écho à la balle de tennis.

A l'aide du logiciel Balsamiq Mockup, j'ai réalisé le wireframe de la boutique en ligne.

C'est un logiciel simple avec beaucoup d'éléments déjà créés. Il offre une large palette de possibilités. Il permet surtout de mettre en avant un premier aperçu du site et montrer où seront placés les éléments (*wireframe en annexe*).

Du wireframe j'ai alors créé une maquette un peu plus détaillée avec les pages principales du site. J'ai utilisé le logiciel Figma pour réaliser la maquette. Cela m'a permis de mettre en place le prototype de mon projet. (*maquette en annexe*).

## **Intégration de l'application**

L'intégration du site web a été réalisée entièrement en HTML / CSS. Concernant le responsive j'ai utilisé les medias queries pour adapter la boutique à tous les supports.

Tout mon CSS a été écrit avec le préprocesseur SASS. Il m'a permis de mettre en place plusieurs fonctionnalités supplémentaires dans mon code.

Principalement cela m'a permis d'imbriquer un élément dans un autre, de rajouter des variables et d'utiliser l'héritage. J'ai par exemple créé un élément *dflex* qui a comme propriété : `display: flex; justify-content: center; align-items: center`.

Au lieu de réutiliser ces mêmes propriétés un peu partout dans mon code, un élément peut hériter des propriétés de *dflex*, grâce à la fonction *@extend dflex*.

On peut également importer un fichier sass dans un autre. Pratique pour créer un fichier réservé aux media-queries dans lequel les variables sont les tailles des écrans en pixels. L'appel aux media queries est donc moins long et plus lisible.

```

@media #{$light-down}{
  #paiement{
    flex-wrap: wrap;

    & > div{
      width: 100%;
      font-size: 2vw;
    }

    & #recap_paiement
    {
      order: 2 ;
    }

    h1{
      font-size: 3vw;
    }
  }
}

```

```

$medium-down: "only screen and (max-width: 1024px)";
$light-down: "only screen and (max-width : 768px)";

```

## Réalisations

**Exemple de mon modèle MVC utilisé pour trier les sacs de tennis en fonction du thermobag (capacité du nombre de raquettes) :**

La méthode *findAllArticles* est une méthode générique de mon Model Article. Elle va chercher en base de données les informations d'un article ainsi que son image principale. Elle est utilisée pour l'affichage des vignettes. Elle prend deux paramètres : *\$order* qui est optionnel et *\$group by*, qui est une chaîne de caractère obligatoire qui vient s'ajouter à la fin de ma requête. L'utilisation de *group by* est nécessaire pour fonctionner avec MIN(chemin).

```

/**
 * Renvoie un tableau avec les infos d'un article + son image principale
 *
 * @param string|null $order
 * @param [type] $groupby
 * @return array
 */
public function findAllArticles(?string $order, $groupby)
{
    $sql = "SELECT articles.id_articles,art_nom,art_courte_description,prix,id_marques,id_categorie,MIN(chemin)
            FROM articles
            NATURAL JOIN images_articles"
    ;

    if($order){
        $sql .= $order;
        $sql .= $groupby ;
        // var_dump($sql);
    }
    else{
        $sql .= $groupby;
    }

    $requete = $this->bdd->prepare($sql) ;
    $requete->execute();

    return $requete->fetchAll();
}

```

La méthode Vue *displayAllArticles* va générer le html, en récupérant le tableau de mon model. Les vignettes vont alors être affichées pour le client.

```

/**
 * Affiche les articles sur la page boutique
 *
 * @param array $result
 * @return void
 */
public function displayAllArticles(array $result): void{
    ?>
    <section class="galerie_article">
        <?php
        foreach($result as $value)
        {
            ?>

            <div class="vignette_article">
                <div class="img">
                    <a href="article.php?id=<?=$value['id_articles'] ; ?>"></a>
                </div>
                <h1> <?=$value['art_nom'] ; ?> </h1>
                <p id="courte_descr_aff_article"> <?=$value['art_courte_description'] ; ?> </p>
                <p> <?=$value['prix'] ; ?> € </p>
            </div>

            <?php
        }
        ?>
    </section>
    <?php
}

```

L'image ci-dessus est une des conditions de la méthode *TrierPar* de mon Controller Article. L'input de tri est un input type **checkbox**. L'utilisateur peut donc choisir entre **une et quatre possibilités** pour la capacité du sac.

Lors du clic, je récupère tous les \$\_POST correspondant.

La fonction *array\_pop* me permet de supprimer le dernier index de mon tableau. Cet index correspond au bouton.

Il me reste donc les input des checkbox.

La fonction *implode* me permet de transformer la valeur de mes index en chaîne de caractères séparé par une virgule.

C'est cette variable *\$valeurs* que l'on passe en paramètre dans le modèle.

On a donc une fonction qui s'adapte selon les choix de l'utilisateur.

On voit donc le contrôleur qui passe les paramètres vers le modèle, puis le modèle qui est transféré à la vue.

L'affichage est donc effectué lors du clic sur le bouton.

```
elseif(isset($_POST['tri_thermo'])){\n    $delete = array_pop($_POST);\n    $valeurs = implode(",", $_POST);\n\n    $thermo = $article->findAllArticles("{ $cat } AND sac_thermobag IN ({ $valeurs })",\n    " GROUP BY articles.id_articles,art_nom,art_courte_description,prix,id_marques,id_categorie" ) ;\n    $view_article->displayAllArticles($thermo);\n}
```

## Veille sur les vulnérabilités de sécurité :

### Faible XSS

Pour décrire rapidement la faille XSS on peut simplement dire qu'elle consiste à injecter du code malicieux dans une page Web, grâce à une variable faillible (c'est-à-dire non sécurisée ou mal sécurisée) qui sera affichée.

Si l'utilisateur rentre une balise dans un input, elle sera alors interprétée par le navigateur (ex : `<script> Je fais ce que je veux ici ! </script>`).

Pour pallier ce problème j'ai utilisé *htmlspecialchars()* qui convertit les caractères spéciaux en entités HTML.

### Injection SQL :

L'injection SQL est un type d'attaque sur une application web qui permet à un attaquant d'insérer des instructions SQL malveillantes dans l'application web, pouvant potentiellement accéder à des données sensibles dans la base de données ou détruire ces données.

Lors d'un formulaire de connexion par exemple, l'utilisateur pourra modifier la requête SQL via l'input de la connexion.

Pour éviter une injection SQL on peut utiliser PDO et la fonction `bindParam()`.

Sur toutes les requêtes de mon projet, lorsque l'utilisateur est amené à entrer des informations, on échappe les variables de la requête en utilisant une requête préparée.

Voici un cas d'utilisation :

```
public function inscription_user()
{
    $uti_droits = 0;
    $nom = htmlspecialchars($_POST['nom']);
    $prenom = htmlspecialchars($_POST['prenom']);
    $mail = htmlspecialchars($_POST['mail']);
    $telephone = htmlspecialchars($_POST['telephone']);
    $mdp = password_hash($_POST['mdp'], PASSWORD_DEFAULT);
    $rue = htmlspecialchars($_POST['rue']);
    $code_postal = htmlspecialchars($_POST['code_postal']);
    $ville = htmlspecialchars($_POST['ville']);

    $requete = $this->bdd->prepare("INSERT INTO utilisateurs(uti_droits,uti_nom,uti_prenom,uti_mail,uti_tel,uti_motdepasse,uti_rue,uti_code_postal,
VALUES (:uti_droits,:nom,:prenom,:mail,:telephone,:mdp,:rue,:code_postal,:ville)");

    $requete->bindParam(':uti_droits', $uti_droits);
    $requete->bindParam(':nom', $nom);
    $requete->bindParam(':prenom', $prenom);
    $requete->bindParam(':mail', $mail);
    $requete->bindParam(':telephone', $telephone);
    $requete->bindParam(':mdp', $mdp);
    $requete->bindParam(':rue', $rue);
    $requete->bindValue(':code_postal', $code_postal);
    $requete->bindValue(':ville', $ville);

    $requete->execute();
}
```

### Faille upload :

Le principe de l'attaque est très simple. Le pirate essaie d'uploader un fichier qui contient du code malveillant ou un code PHP de sa création. Si la faille est là, alors le fichier finira par atterrir sur le serveur. Il suffit ensuite au pirate d'appeler son fichier pour que celui-ci s'exécute.

C'est un risque lorsque l'utilisateur est amené à upload une image, une vidéo ou un CV.

Dans le cas de notre projet, nous devons prévoir la sécurité lors de l'insertion d'image de produit.

Pour cela j'ai créé plusieurs conditions :

- Une pour définir le type d'extensions que l'on va recevoir (jpg, jpeg,png).

- Une autre pour limiter le poids du fichier.

Voici le code pour assurer la sécurité :

```
$tailleMax = 2097152;
$extensionsValides = array('jpg', 'jpeg', 'gif', 'png');
if($_FILES['image']['size'][$i] <= $tailleMax)
{
    $extensionUpload = strtolower(substr(strrchr($_FILES['image']['name'][$i], '.'),1));
    if(in_array($extensionUpload, $extensionsValides))
    {
        $chemin = "../medias/img_articles/".$_result['id_articles']."-".$i.".".$extensionUpload;
        $mouvement = move_uploaded_file($_FILES['image']['tmp_name'][$i], $chemin );
        if($mouvement)
        {
            $admin->insertImage($extensionUpload, $i);
        }
        else{
            return "Erreur durant l'importation du fichier";
        }
    }
    else{
        return "Votre image doit etre au format jpg, jpeg, gif ou png" ;
    }
}
else{
    return "L'image ne dois pas dépasser 2mo" ;
}
```

### **Attaque de force brute :**

Consiste via un programme de tester toutes les combinaisons possibles pour trouver un mot de passe et accéder à un compte admin ou utilisateur.

Pour l'utilisateur, il est nécessaire d'utiliser des mots de passe différents sur chaque site afin d'assurer plus de sécurité.

Utiliser des mots de passe complexes. Forcer durant l'inscription l'utilisateur a remplir des mot de passes selon des conditions précises (nombres de caractères, majuscules, chiffres, caractères spéciaux...).

Pour ça on peut utiliser les expressions régulières et la fonction preg\_match pour vérifier les mot de passes selon certaines conditions.



```

if( preg_match('#^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*\W){8,}$#', $new_mdp))
{
    $user->update_mot_de_passe($new_mdp);
    header('Location: messages_et_redirections/profil_modifie.php');
    exit();
}
else
{
    return 'Le mot de passe doit contenir au moins 8 caractères, une majuscule, une minuscule,
}

```

## Projet Les Playgrounds du Sud :

Il s'agit d'un site réalisé pour le graphiste freelance Milan Jambou. Le but est de réunir les basketteurs à Marseille, faire « décoller » la culture basket dans la ville. Le cœur du site sera au niveau d'une carte, créé par un graphiste, où chaque terrain sera répertorié, permettant à l'utilisateur de signaler sa présence à une heure précise. Tous les terrains seront des ballons positionnés sur la carte. L'utilisateur pourra, une fois inscrit, cliquer sur un ballon pour afficher les informations du terrain et signaler sa présence.

Chaque utilisateur aura un pseudo, un avatar, un poste ainsi que ses préférences au niveau des terrains. Il aura également un top 5 des terrains les plus aimés et un top 5 des terrains les plus fréquentés en temps réel.

Tous les terrains auront des informations détaillées : nom, coordonnées gps, point d'eau, éclairage, nombre de panier, type de planche, présence de filet, hauteur de l'anneau...

La maquette a été entièrement réalisée par le graphiste. Je me suis en revanche occupé du développement du site et de la base de données.

*(Le modèle conceptuel de données de la base est en annexe.)*

### Technologie utilisée :

#### **Partie Back-end :**

Php en langage back, sql pour la base de données.

#### **Partie Front-end :**

HTML / CSS et le langage Javascript.

J'ai également utilisé quelques librairies : font awesome pour certains pictogrammes, animate.css pour les animations et JQuery.

## Réalisations et extraits de code

### Exemple du placement des ballons sur la map

La page d'accueil du site est une carte de la ville de Marseille. Sur cette map sont positionnés des ballons indiquant la position des terrains. Ces ballons sont des liens vers la page terrains avec l'id du terrain en GET.



Pour placer les ballons, la première étape était sur photoshop. Ayant l'image avec tous les ballons, je l'ai importé dans le logiciel et relevé grâce à la grille, à quel pourcentage de l'image chaque ballon est-il situé à partir de la gauche et du haut.

J'ai donc enregistré toutes ces coordonnées puis je l'ai intégré à la base de données, dans la table réservée aux terrains.

Chaque terrain a donc dans la base de données, des coordonnées X et Y.

Au niveau du code, j'ai pu placer ces ballons grâce à ces données. L'image de la carte est en *position: relative*, tous les ballons sont en absolute, avec deux valeurs : top: coordonnées Y et left: coordonnées X.

J'ai donc utilisé Javascript pour générer tous les ballons. Grâce à une requête ajax, je lui indique le chemin du fichier php, qui va me renvoyer un fichier json avec toutes les coordonnées de chaque terrain.

Je crée alors une boucle for qui parcourt mon fichier json. A chaque tour de boucle un `<span>` est créé avec le lien du terrain et le svg du ballon.

L'avantage d'utiliser les pourcentages pour le placement, c'est pour le responsive. Lorsque l'on réduit la fenêtre, les terrains ne bougent pas et restent parfaitement positionnés.

*Requête ajax en javascript :*

```
// placement ballon

var bloc_map = document.getElementById("img") ;

$.ajax({
  type: "GET",
  url: "script_ajax/position_terrain.php",
  dataType: "json",
  async: false,
  success: function (response) {

    console.log(response);
    for(var i = 0 ; i < response.length ; i++)
    {
      console.log("dans la boucle") ;
      var new_span = document.createElement("span") ;

      // new_span.className = "link_terrain";

      new_span.innerHTML = "<a href='./pages/terrain.php?id="+ response[i].id + "' class='link_terrain'><img src='./src/img/ballon.svg'></a>" ;

      new_span.setAttribute("style" , "top: "+ response[i].position_map_y +"%; left: "+ response[i].position_map_x +"%;") ;

      bloc_map.append(new_span) ;
    }
  }
});
```

*La page position\_terrain.php qui affiche le resultat au format JSON.*

```
<?php

require_once("../libraries/autoload.php");

$terrain = new \Models\Terrain();

$result = $terrain->getPostionTerrain();

echo json_encode($result) ;
```

La méthode `getPosition Terrain` qui va chercher les informations pour pouvoir placer les terrains.

```
public function getPostionTerrain()
{
    $requete = $this->bdd->prepare("SELECT id,position_map_x,position_map_y
                                   FROM terrains
                                   ");
    $requete->execute();

    $result = $requete->fetchAll(PDO::FETCH_CLASS, __NAMESPACE__ . '\\Terrain');

    return $result;
}
```

### **Affichage des joueurs présents sur un terrain**

Lorsque l'utilisateur est connecté, il a la possibilité de faire plusieurs choses sur la page terrains. Il peut signaler sa présence à une heure précise et voir quels sont les joueurs présents sur ce terrain toute la semaine à partir de la date sélectionnée. Il peut également cliquer sur une étoile pour ajouter un terrain à ses favoris.

Si l'utilisateur clique sur ces fonctionnalités sans être connecté, alors un encart de connexion apparaît avec un fondu, forçant l'utilisateur à se connecter.

Dans le cas où l'utilisateur n'est **pas connecté** :

Une requête Ajax est lancée, qui renvoie le code html de la page connexion.php.

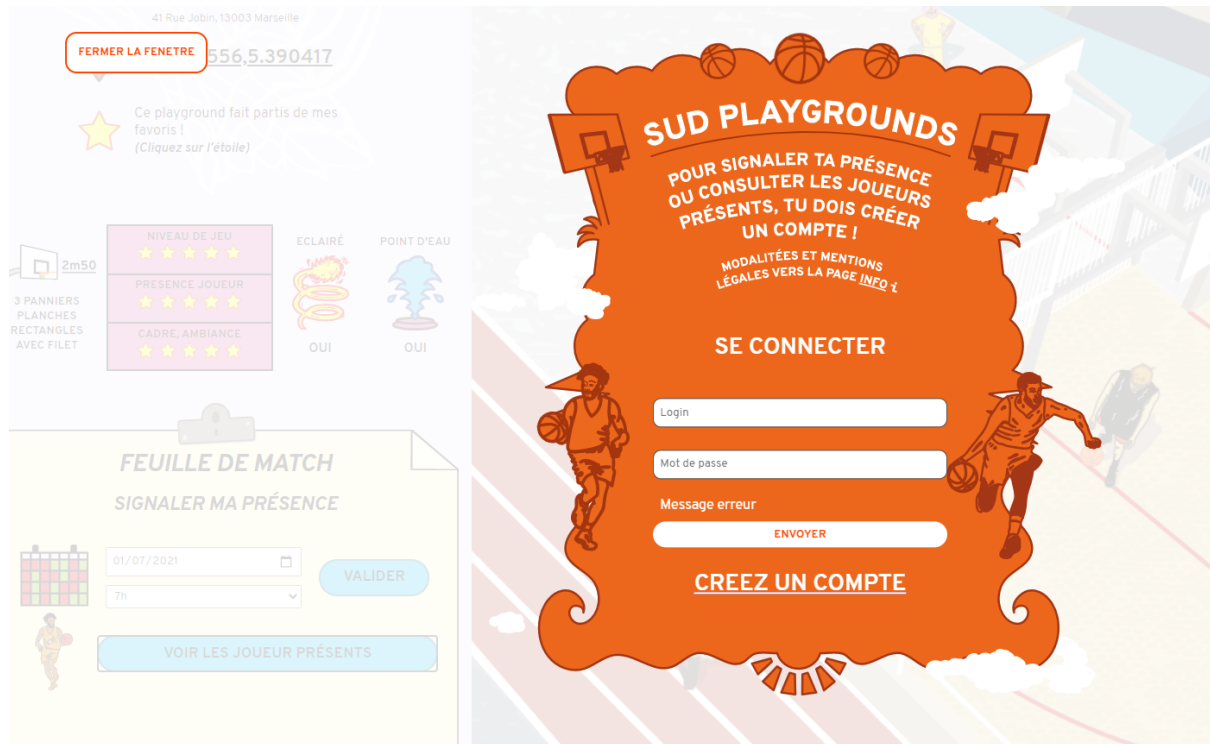
```
else{

    $.ajax({
        type: "GET",
        url: "connexion.php",
        data: "data",
        dataType: "text",
        success: function (response) {
            $('#page_connexion').append(response);
            $('#page_connexion').addClass("animate__animated animate__fadeIn")
            setTimeout(function(){
                $('#page_connexion').removeClass("animate__animated animate__fadeIn")
            },1000);
        }
    });
}
```

Ici, la fonction *append* me permet d'ajouter le contenu retourner par la requête dans le bloc page connexion. Je lui ajoute également deux classes qui appartiennent à une librairie, pour pouvoir ajouter une entrée en fondue.

Au bout d'une seconde je supprime ces classes, car j'ai un autre événement sur un bouton pour fermer l'encart et continuer à visiter le site sans être connecté. Dans ce cas-là ce sera une animation *fadeOut*.

Encart de connexion avec un *background-color* : *white* et une *opacité* de 50%.



Dans le cas où l'utilisateur est **connecté**, il a la possibilité de voir la liste des joueurs présents sur le terrain dans la semaine.

Lors de l'événement click du bouton, on lance donc la requête ajax vers un fichier php qui renvoie un fichier json avec les bonnes informations. On fait passer deux variables dans l'attribut *data* pour les transmettre au fichier php et qu'il puisse exécuter à bien sa requête.

```
var day = document.getElementById("day").value ;

$.ajax({
  type: "GET",
  url: "../script_ajax/voir_joueur_present.php",
  data: {day: day , id_terrain: get_id},
  dataType: "json",
  success: function (response) {
    console.log(response)
  }
})
```

Dans le fichier json récupérer on a donc : l'avatar de la personne, son login et sa date de présence.

On va donc maintenant créer une boucle qui va nous afficher une liste de joueurs. Pour réaliser cette liste, on va devoir créer nos éléments à l'aide de la méthode `document.createElement()`. Un élément "p" dans lequel il y aura une image, un lien (le nom de la personne renvoie sur sa page profil.php) et la date.

```
for(let i = 0; i < response.length ; i++)
{
  //On crée une date
  let date = new Date(response[i].heure_presence);

  let dateLocale = date.toLocaleString('fr-FR',{
    weekday: 'long',
    year: 'numeric',
    month: 'long',
    day: 'numeric',
    hour: 'numeric'});

  // On créé une image
  var img = new Image() ;
  img.src = '../src/img/'+response[i].avatar ;
  img.classList.add("avatar");

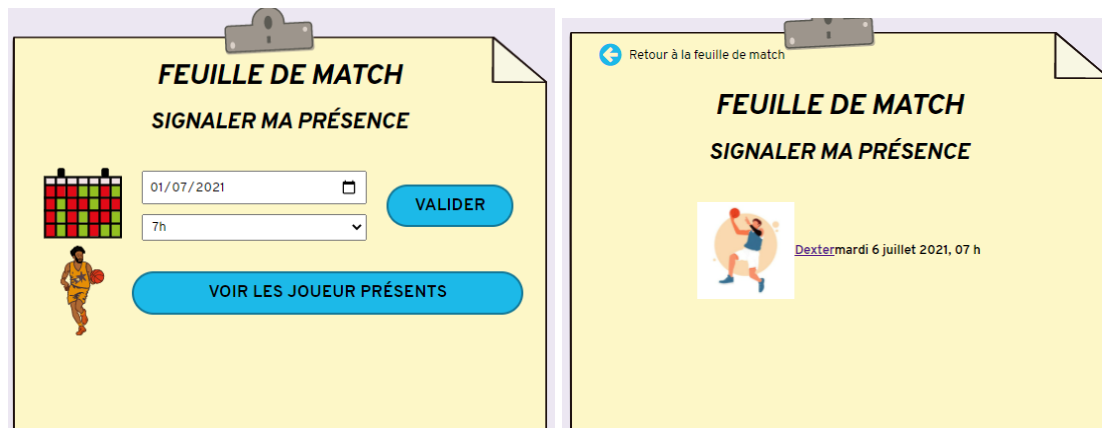
  //on créé le link login
  var a = document.createElement('a');
  var link = document.createTextNode(response[i].login);
  a.appendChild(link);
  a.title = response[i].login;
  a.href = "profil.php?id="+ response[i].id_joueur;

  // on créé le p
  var p = document.createElement("p") ;
  p.classList.add("joueur_present_date");
  p.append(img,a,dateLocale) ;

  bloc_feuille_match.append(p) ;
}
```

On transforme également les dates grâce à la fonction *ToLocaleString()*. Dans le cas ci-dessus la variable date locale sera de type (vendredi 2 juillet 2021, 9 h) alors que le format récupéré était "02-07-2021 09:00:00".

Pour finir on utilise la fonction *append()*, pour envoyer tous les éléments dans la balise *p*, puis le *p* dans le *bloc\_feuille\_match*. On obtient donc ce résultat.



Via une petite flèche on peut retourner sur la feuille de match. J'utilise la propriété *innerHTML* du bloc feuille match pour vider le contenu de la section ( `element.innerHTML = ""` ).

Puis j'effectue une requête ajax qui me renvoie le contenu HTML des deux input et les deux boutons de mon formulaire.

## Organisation du travail

Avec ce projet, c'est la première fois que je travaille pour un professionnel en tant que développeur.

Cela m'a appris à identifier les besoins d'un client et de trouver des solutions techniques pour répondre à sa demande.

Nous avons beaucoup échangé, nous travaillons ensemble deux jours par semaine. Le métier de designer et de développeur se complète et nous avons mis en place une véritable alchimie de travail. J'ai donc aussi beaucoup appris en termes d'interface et d'expérience utilisateur.

J'ai pu aussi de mon côté lui apporter des connaissances dans le développement.

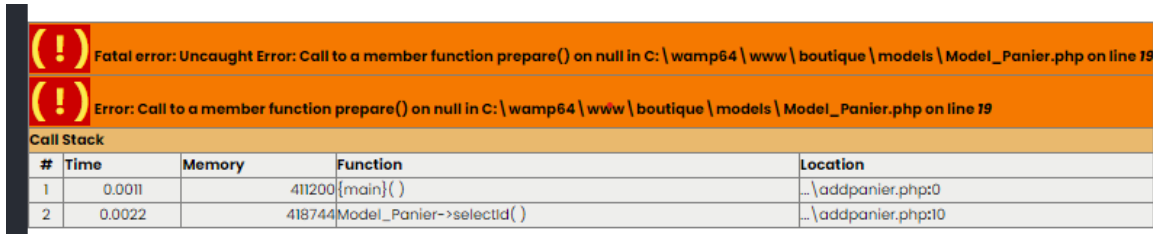
## Exemple d'une recherche en anglais

Rechercher une erreur sur un navigateur fait partie intégrante du métier de développeur. Une bonne recherche permet de résoudre rapidement un problème. Une recherche en anglais permet de cibler beaucoup plus de forum et de site avec une communauté du monde entier.

C'est pourquoi au cours de mon apprentissage, sur tous mes projets je m'efforce à chercher en anglais. C'est aujourd'hui devenu la totalité de mes recherches, même si il m'arrive de trouver des réponses sur quelques forums français comme openclassrooms ou developpez.com.

Durant le développement de ma boutique, toutes mes classes modèles héritent d'une classe mère appelée Model, qui dans son constructeur lance la connexion à la base de données.

Durant le développement du panier, j'obtiens alors cette erreur :



The screenshot shows a PHP error message in an orange box. The error is a 'Fatal error: Uncaught Error: Call to a member function prepare() on null in C:\wamp64\www\boutique\models\Model\_Panier.php on line 19'. Below the error message is a 'Call Stack' table with two rows. The first row shows the main function call, and the second row shows the call to the Model\_Panier constructor.

#	Time	Memory	Function	Location
1	0.0011	411200	{main}()	...\addpanier.php:0
2	0.0022	418744	Model_Panier->selectId()	...\addpanier.php:10

Je comprends donc que mon `$this->bdd` est null, ce qui veut dire qu'il y a un problème sur la connexion à la base de données. Toutes mes classes Model jusqu'à présent n'avaient eu aucun problème de connexion, seulement pour le panier ce n'était pas bon.

Après quelques minutes je remarque que mon Panier est la seule classe fille à avoir un constructeur.

Je tape alors sur google : *"class not inherit constructor parent php"*

Ma recherche tombe en premier sur le forum de stackoverflow avec la question ci-dessous :



▲ Is it mandatory to call the parent's constructor from the constructor in the child class constructor?

5 To explain consider the following example:

▼

```
class Parent{  
    function __construct(){  
        //something is done here.  
    }  
}  
  
class Child extends Parent{  
    function __construct(){  
        parent::__construct();  
        //do something here.  
    }  
}
```

This is quite normal to do it as above. But consider the following constructors of the class `Child` :

```
function __construct(){  
    //Do something here  
    parent::__construct();  
}
```

Is the above code correct? Can we do something before you call the parent's constructor? Also if we do not call the parent's constructor in the child's constructor like below is it legal?

```
class Child extends Parent{  
    function __construct(){  
        //do something here.  
    }  
}
```

I am from JAVA, and the types of constructor I have shown are not possible in Java. But can these be done in PHP?

Il est ici demandé s'il est obligatoire d'appeler le constructeur parent depuis le constructeur de la classe fille. Il explique ceci avec l'exemple qui suit. Il dit que c'est tout à fait normal de le faire comme ci-dessus, mais considérons le constructeur suivant de la classe fille.

Le code ci-dessus est-il correct ? Peut faire quelque chose avant d'appeler le constructeur parent ? Est-ce également possible de ne pas appeler le constructeur parent comme dans le code ci-dessous ?

Ce à quoi une personne lui répond :

2 Answers

Active

Oldest

Votes



10



Is it mandatory?

No



Is the above code correct?



Yes

Can we do something before you call the parent's constructor?

Yes. You can do it in any order you please.

Also if we do not call the parent's constructor in the child's constructor like below is it legal?

Yes. But it's not **implicit** either. If the child constructor doesn't call the parent constructor then it will never be called because the child constructor overrides the parent. If your child has no constructor then the parent constructor will be used

Ce n'est donc pas obligatoire. Il est possible d'appeler le constructeur parent dans n'importe quel ordre. C'est tout à fait légal de ne pas appeler le constructeur parent, mais ce n'est pas implicite pour autant. Il explique que le constructeur enfant à toujours le dessus sur celui du parent, dans le cas où l'on n'appelle pas le constructeur parent. En revanche, si la classe fille n'a pas de constructeur, elle hérite automatiquement du parent.

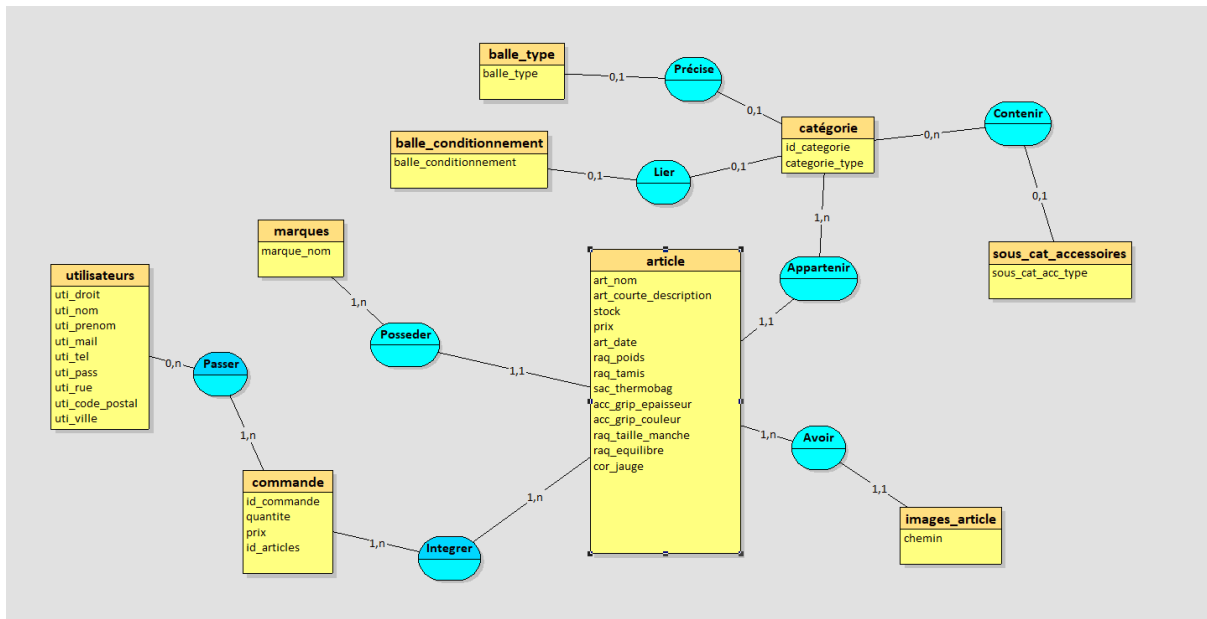
C'était exactement le problème concernant mon modèle Panier. Je n'appelle pas le constructeur parent, ce qui fait que le constructeur de la classe prend le dessus.

J'ai donc rajouté le `parent::__construct()`; à l'intérieur du constructeur.

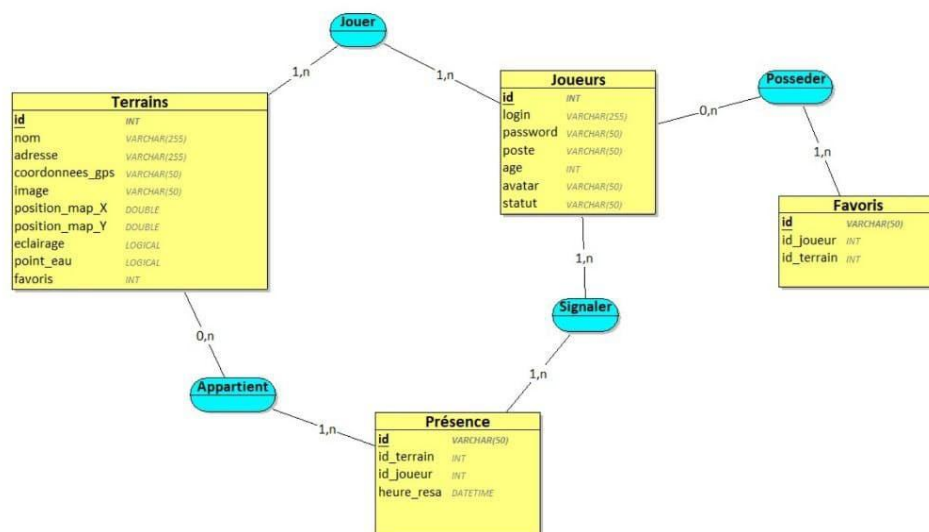
```
public function __construct(){
    parent::__construct();
    if(!isset($_SESSION))
    {
        session_start();
    }
    if(!isset($_SESSION['panier'])){
        $_SESSION['panier'] = array();
    }
}
```

Annexes :

### Mvc Boutique en ligne



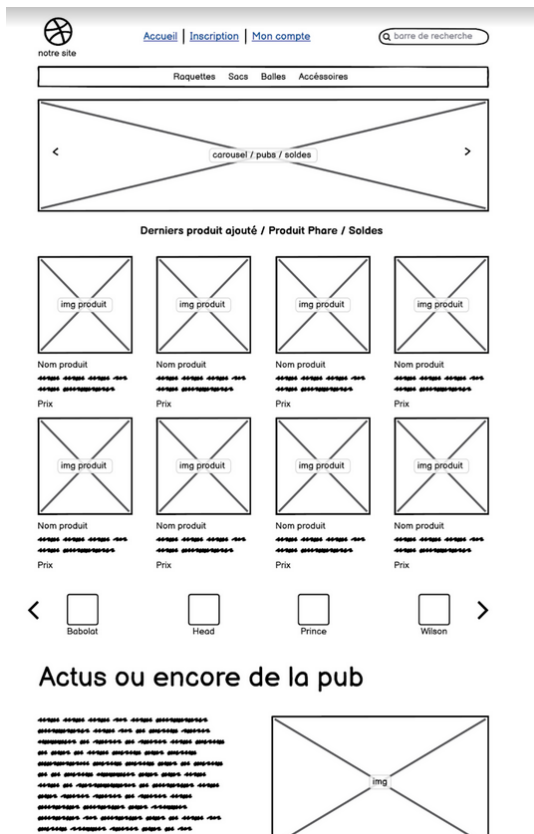
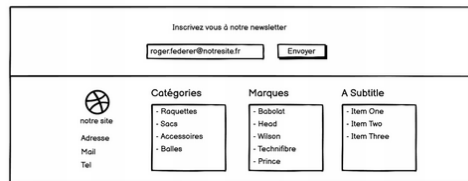
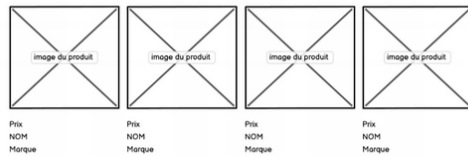
### Mvc Playgrounds du Sud

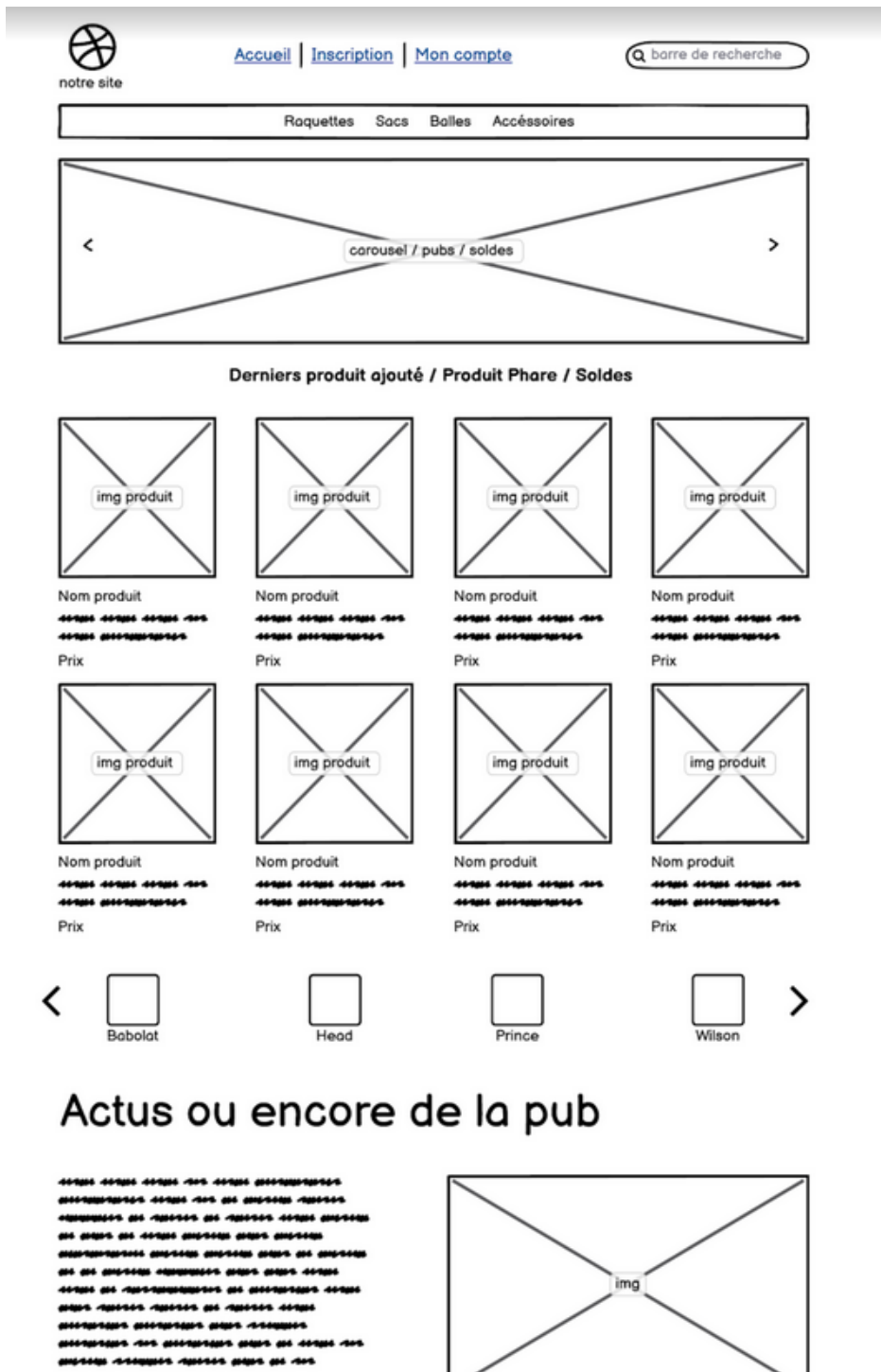


## Wireframe boutique en ligne ( Balsamiq )



### Propositions articles de la même catégorie





Maquette Figma Boutique en ligne

