

Romdhani Mehdi

Dossier Projet Professionnel : GameVault

Titre Développeur Web & Web Mobile



Table des matières

| | |
|--|----|
| Compétences du référentiel couvertes par le projet..... | 3 |
| Résumer..... | 4 |
| Spécifications fonctionnels..... | 5 |
| Spécifications techniques..... | 7 |
| Architecture du Projet..... | 8 |
| Maquette du Projet..... | 11 |
| Conception & Modélisation de la base de donnée..... | 14 |
| Description des fonctionnalités de l'application..... | 16 |
| Réalisations (extraits de code)..... | 20 |
| Jeu d'essai..... | 28 |
| Veille Projet & Vulnérabilités de Sécurité..... | 35 |
| Description d'un situation de travail & recherche site anglophone..... | 38 |
| Annexes..... | 41 |

Compétences du référentiel couvertes par le projet

Voici la retranscription des compétences pour les activités 1 et 2 :

Activité 1 - "Développer la partie front-end d'une application web et web mobile en intégrant les recommandations de sécurité" :

1. Maquetter une application.
2. Réaliser une interface utilisateur web ou mobile statique et adaptable.
3. Développer une interface utilisateur web dynamique.
4. Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce.

Activité 2 - "Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité" :

1. Créer une base de données.
2. Développer les composants d'accès aux données.
3. Développer la partie back-end d'une application web ou web mobile.
4. Élaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce.

Résumé

Dans le cadre de mon passage du diplôme, mon objectif était de créer une application qui réunit l'ensemble des compétences citées dans le référentiel. Ainsi, j'ai développé une plateforme de vente en ligne de jeux vidéo appelée **GameVault**.

L'application vise à offrir aux utilisateurs la **possibilité d'effectuer des achats d'articles**, de **créer et gérer** leur propre compte client, ainsi que de faciliter la recherche des articles de leur choix. Les utilisateurs peuvent trouver facilement les jeux qu'ils souhaitent acheter et profiter **d'une expérience utilisateur optimisée**.

Grâce à cette plateforme, les utilisateurs peuvent parcourir une large sélection de jeux vidéo, consulter les descriptions détaillées. Ils peuvent également ajouter des jeux à leur panier, passer des commandes sécurisées et effectuer des paiements en ligne de manière pratique.

Par ailleurs, l'application offre aux utilisateurs la possibilité de **créer leur propre compte client**. Une fois inscrits, ils peuvent enregistrer leurs informations personnelles, gérer leur historique d'achats .

GameVault propose une interface conviviale et simplifiée, permettant aux utilisateurs de trouver rapidement les jeux qui les intéressent. **Ils peuvent effectuer des recherches par titre, catégorie, sous catégorie ou plateforme, et utiliser des filtres pour affiner les résultats**. Cela leur permet de découvrir de nouveaux jeux et de trouver facilement ceux qu'ils recherchent.

En résumé, l'application GameVault que j'ai développée répond aux exigences du référentiel en intégrant les compétences liées à la création d'une plateforme de vente en ligne de jeux vidéo. Elle offre aux utilisateurs la possibilité d'effectuer des achats, de gérer leur compte, et de trouver aisément les jeux qu'ils souhaitent acquérir.

Spécifications fonctionnelles

Description de l'existant :

- Le projet a été conçu en groupe, et nous avons réalisé toutes les étapes depuis le maquettage jusqu'à la création et la modélisation de la base de données, jusqu'à la version finale du produit. Nous avons développé l'ensemble du projet à partir de zéro, en codant tout de manière autonome, **sans utiliser de projet existant préalablement.**

Périmètre du projet :

- Le site sera en français et adapté à tous les supports : **mobiles, tablettes et ordinateurs.**

Cible visée pour le site internet :

- La plateforme s'adresse spécifiquement aux joueurs, aux passionnés de jeux vidéo et aux collectionneurs qui sont à la recherche de nouveaux titres et qui souhaitent effectuer des achats.

Spécifications fonctionnelles (suite)

Arborescence :

- Frontend :

- Page Accueil
- Page Inscription
- Page Connexion
- Page Profil
- Page Produits
 - Produit individuel
 - Panier
- Page Paiement
- Page d'adresse de livraison

- Backend (Back Office) :

- Tableau de bord
- Gestion des produits
- Gestion des utilisateurs

Le choix de cette arborescence permet d'organiser de manière cohérente et intuitive les différentes pages du site web, en offrant aux utilisateurs un accès facile aux fonctionnalités principales telles que l'inscription, la connexion, la navigation parmi les produits, le panier, le paiement, etc. De plus, la mise en place d'un Back Office permet une gestion efficace des produits, des utilisateurs et des commandes, offrant ainsi une interface dédiée à l'administration et à la gestion complète du site.

Spécifications Techniques (Stack technologiques)

Les technologies choisies pour la **stack technologique** pour notre application selon les besoins spécifiques du projet sont :

- Frontend :

- **HTML/CSS** pour la structure et la présentation des pages web.
- **JavaScript (vanilla ES5/6)** pour la logique côté client et l'interactivité.

- Backend :

- **PHP (Hypertext Preprocessor)** pour la construction de la partie serveur.
- **Système de gestion de base de données (SGBD) : MariaDb** pour stocker, gérer et récupérer les données des produits, des utilisateurs, etc.
- **SQL (Structured Query Language)** pour la communication avec la base de données . Cela va permettre de créer , modifier, supprimer , mettre à jour les données de notre application **à travers différentes requête** .

- Outil de gestion de la base de données :

- **PHPMyAdmin** pour administrer et gérer la base de données MySQL.

- Gestion de l'infrastructure :

- Hébergement web avec **Plesk**.
- Utilisation de **Git pour le versioning du code source**, la collaboration et le suivi des modifications.
- Utilisation de **GitHub** pour l'hébergement du code source.

- Maquettage

- Utilisation de **Figma** pour réaliser les maquettes.

- Environnement de développement :

- **Visual Studio Code** comme environnement de développement.

Architecture du Projet

I - Architecture M.V.C

Pour la conception de l'application, la question s'est posée de savoir comment obtenir une architecture de code efficace, modulable, réutilisable et sécurisée. Nous avons donc opté pour l'utilisation du **design pattern MVC (Modèle-Vue-Contrôleur)**.

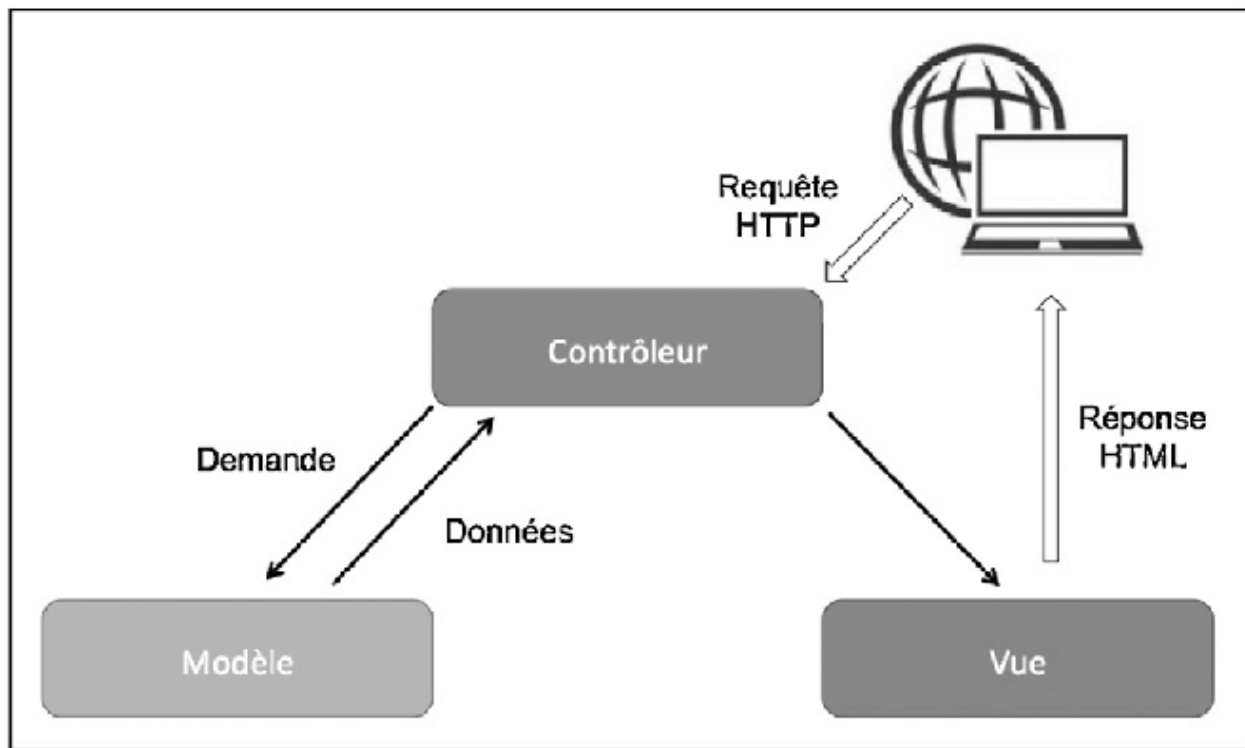
Le design pattern MVC est **une architecture** couramment utilisée dans la construction d'applications web. Il se compose **de trois modules principaux** :

1. **Le modèle (Model)** : C'est la couche qui gère les données et la logique métier de l'application. Le modèle est responsable de l'accès aux données, de leur manipulation et de leur persistance. Il représente la structure des données utilisées dans l'application.
2. **La vue (View)** : C'est la couche responsable de **l'interface utilisateur**. Elle est chargée de l'affichage des données et de l'interaction avec l'utilisateur. La vue communique avec le contrôleur pour récupérer les données à afficher.
3. **Le contrôleur (Controller)** : C'est la couche qui gère **la logique de l'application**. Il reçoit les actions de l'utilisateur depuis la vue, effectue les traitements nécessaires et met à jour le modèle en conséquence. Le contrôleur est responsable de la coordination entre la vue et le modèle.

L'architecture MVC permet une séparation claire des responsabilités et favorise la réutilisation du code. Chaque module peut être développé indépendamment, ce qui facilite la maintenance et les évolutions de l'application. De plus, l'utilisation du design pattern MVC contribue à améliorer la sécurité de l'application en évitant les mélanges entre les couches de données et d'interface utilisateur.

(Permet d'obtenir une structure **de code efficace, modulable, réutilisable et sécurisée**, favorisant ainsi le développement d'**une application web robuste et évolutive**).

II - Schéma d'architecture M.V.C



Maquette du projet

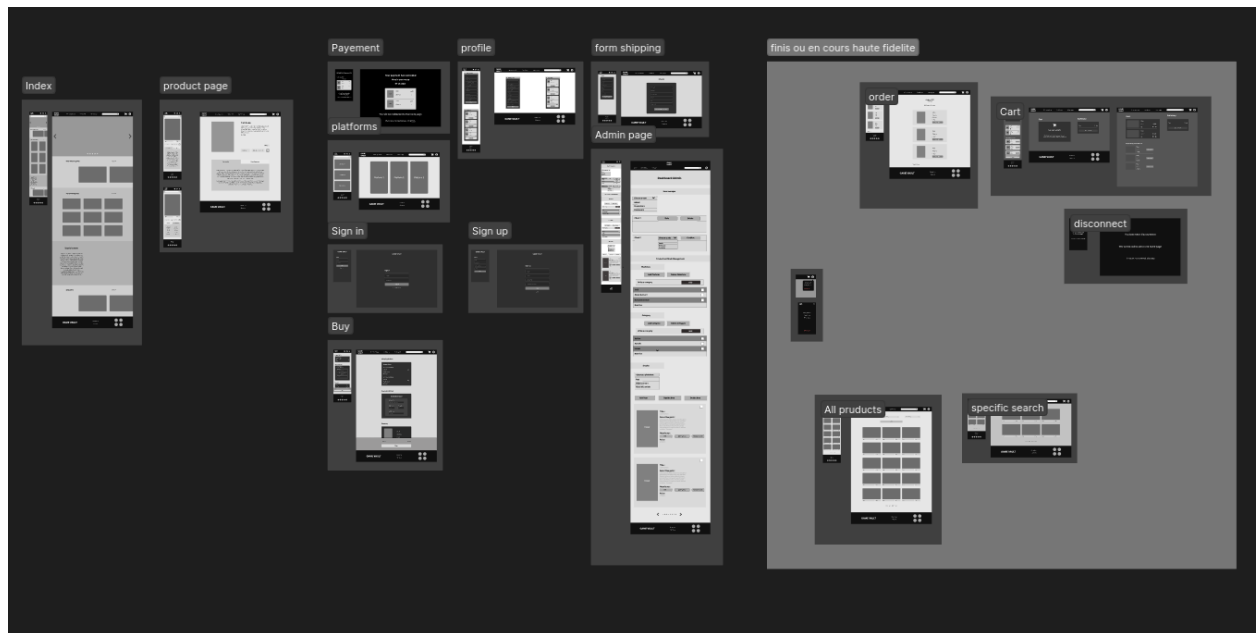
I - Conception Maquette

Avant de commencer le développement du projet, il est essentiel de procéder étape par étape. **Une première étape** consiste à créer une maquette de l'interface graphique de notre application, afin d'avoir une idée visuelle de son apparence et de son agencement.

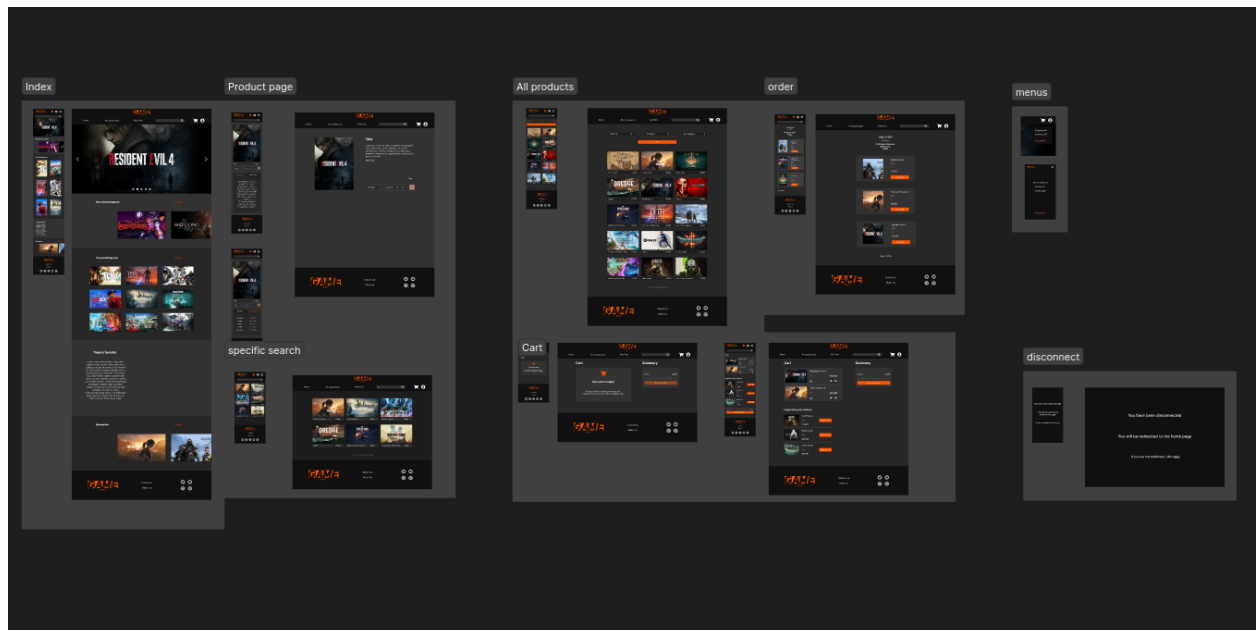
Pour réaliser cette tâche, j'ai utilisé **Figma**, une application dédiée à la création d'interfaces graphiques pour des applications web, mobiles et d'autres plateformes. Figma offre une interface conviviale qui permet de **concevoir et de prototyper** des designs interactifs.

Grâce à Figma, j'ai pu créer **des wireframes et des maquettes** visuelles pour chaque page de l'application, en disposant les différents éléments tels que **les boutons, les formulaires, les menus**, etc. Cela m'a permis de visualiser l'aspect global de l'interface, d'affiner la disposition des éléments et d'ajuster les interactions.

WireFrame Basse-Fidélité



WireFrame Haute-Fidélité



CONCEPTION & MODÉLISATION DE LA BASE DE DONNÉE

I - Méthode Merise

Une fois la maquette réalisée, nous pourrons procéder à la conception et à la modélisation de notre base de données pour l'ensemble de notre application. Pour cette tâche, nous utiliserons **la méthode Merise**.

La méthode Merise est une approche de modélisation de bases de données qui se base sur trois modèles :

- **MCD (Modèle Conceptuel des Données)** : Il représente les concepts et les **relations entre les entités** .
- **MLD (Modèle Logique des Données)** : Structure des données à un niveau plus détaillé, en utilisant **des schémas relationnels, des contraintes d'intégrité et des clés étrangères**.
- **MPD (Modèle Physique des Données)** : Il représente la façon dont les données sont réellement stockées **sur un système de gestion de bases de données (PhpMyAdmin)**.

Cela à pour but de concevoir des bases de données efficaces et bien structurées.

Nous avons choisi d'utiliser la méthode Merise pour notre élaboration de base de données pour plusieurs raisons. Tout d'abord, cette méthode permet de prendre en compte les différents aspects d'un système d'information, en identifiant les entités, les relations, les attributs et les contraintes nécessaires. Elle favorise une modélisation claire et cohérente de la structure des données.

***Voir annexe pour consulter le M.C.D / M.L.D / M.P.D**

DESCRIPTIONS DES FONCTIONNALITÉS DE L'APPLICATION

I - Inscription & Connexion

Lors de la navigation sur le site, l'utilisateur aura la possibilité d'accéder à **une page d'inscription**. Sur cette page, il lui sera demandé de fournir un nom d'utilisateur, un mot de passe et une confirmation de mot de passe. Ces informations seront utilisées pour créer un compte client dans la base de données. Une fois inscrit, l'utilisateur pourra **se connecter à via une page de connexion** sur son espace personnel pour effectuer des achats, modifier son profil etc .

II - Catalogue Produits / Filtre

Une page a été créée spécifiquement pour afficher tous les produits, avec des filtres dédiés par **plateforme de jeux, catégorie et sous-catégories**, afin de permettre aux utilisateurs d'effectuer des recherches précises.

III - Fiche Produit

L'utilisateur sera en mesure d'accéder à une fiche produit cet page comprend :

- Une photo du produit
- Nom du produit
- Description du produit
- Prix du produit
- Renseignements produits (Développeur, Date de Publication, etc..)
- Choix de la plateforme
- Choix quantité du produit

IV - Panier Produit

Une page est dédiée à l'affichage du panier de l'utilisateur, où sont affichés tous les produits ajoutés par celui-ci. L'utilisateur a la possibilité d'ajuster **la quantité directement sur chaque élément du panier**. De plus, un bouton est prévu pour rediriger vers la page de paiement, où il pourra effectuer ses achats.

V - Rechercher un produit

L'utilisateur devrait avoir la possibilité de rechercher un jeu à partir de n'importe quelle page de l'application. Pour faciliter cette fonctionnalité, **un champ d'autocomplétion** sera mis en place, permettant à l'utilisateur de saisir des termes de recherche et d'obtenir des suggestions correspondantes en temps réel.

VI - Solution de Paiement

Une fois le panier validé par l'utilisateur, il sera redirigé vers notre solution de paiement intégrée. Sur la page de paiement, il devra fournir son adresse d'envoi et ses coordonnées pour une transaction sécurisée. Des vérifications seront effectuées au niveau des formulaires d'adressage et de paiement afin d'assurer la validité et la sécurité des informations fournies.

VII - Espace Profil

Un utilisateur inscrit aura un espace de profil où il pourra compléter un formulaire avec des informations telles que nom, prénom, adresse d'envoi et date de naissance. Il pourra également mettre à jour son profil, y compris le changement de mot de passe.

VIII - Partie Back-Office

La partie Back-Office de notre application est l'une de ses fonctionnalités les plus importantes. Son rôle principal est de gérer l'ensemble des données de notre site, ce qui comprend :

- **1. Gestion des utilisateurs :** Cette fonctionnalité permettra d'administrer les utilisateurs enregistrés sur notre site, y compris la création de nouveaux comptes, la modification des informations utilisateur et la suppression de comptes si nécessaire.
- **2. Gestion de la marchandise :** Cette fonctionnalité permettra de gérer tous les aspects liés aux produits proposés sur notre site. Cela inclut l'ajout de nouveaux articles, la mise à jour des informations sur les produits existants et la suppression des articles qui ne sont plus disponibles.

VIII - Partie Back-Office (suite)

- **3. Game Manager** : Cette fonctionnalité permettra d'ajouter de nouveaux articles à notre catalogue, de mettre à jour les détails des articles existants tels que le prix, la description, les images, etc., et de supprimer des articles qui ne sont plus pertinents.

En résumé :

Toutes les fonctionnalités mentionnées précédemment seront implémentées dans notre application, garantissant ainsi son fonctionnement complet et son aspect d'une plateforme de vente de jeux vidéo en ligne. Nous nous assurerons de respecter **le cahier des charges prévu à cet effet**, afin de créer une application opérationnelle répondant aux exigences spécifiées.

RÉALISATIONS

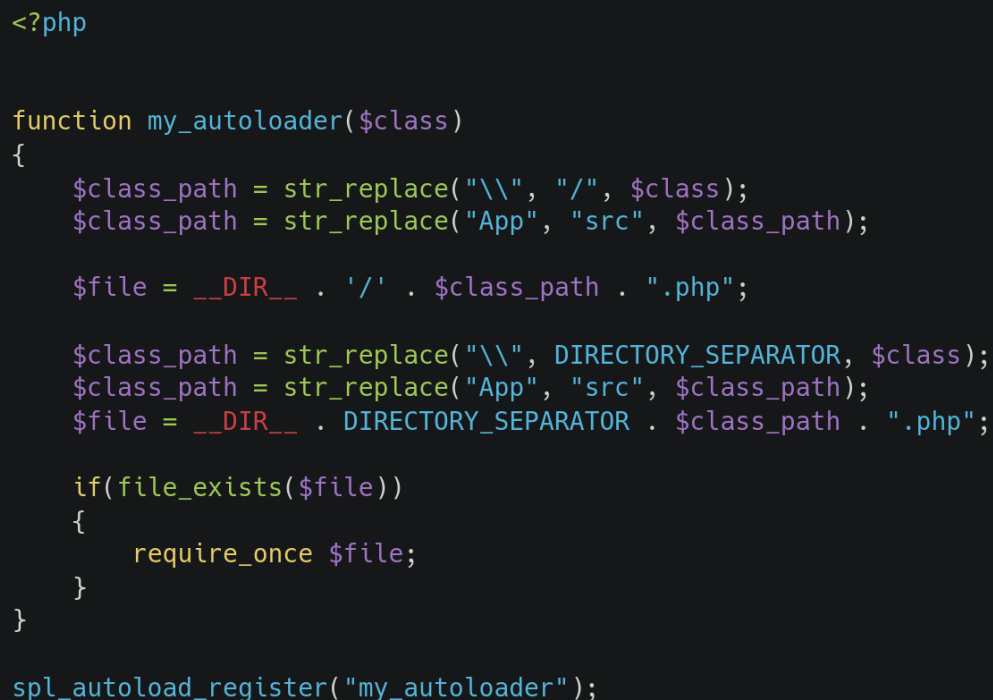
(Extraits de code)

À savoir :

Pour la réalisation de notre projet de boutique en ligne **GameVault**, nous avons utilisé la **Programmation Orientée Objet** .

Pourquoi ? En raison de ses nombreux avantages. La **P.O.O** permet de structurer et d'organiser efficacement le code en le regroupant en objets autonomes, qui encapsulent à la fois des données et des fonctionnalités liées à ces données . Le code devient plus modulaire et facilite la maintenance .

I - Autoloader



```
<?php

function my_autoloader($class)
{
    $class_path = str_replace("\\", "/", $class);
    $class_path = str_replace("App", "src", $class_path);

    $file = __DIR__ . '/' . $class_path . ".php";

    $class_path = str_replace("\\", DIRECTORY_SEPARATOR, $class);
    $class_path = str_replace("App", "src", $class_path);
    $file = __DIR__ . DIRECTORY_SEPARATOR . $class_path . ".php";

    if(file_exists($file))
    {
        require_once $file;
    }
}

spl_autoload_register("my_autoloader");
```

I - Autoloader (suite)

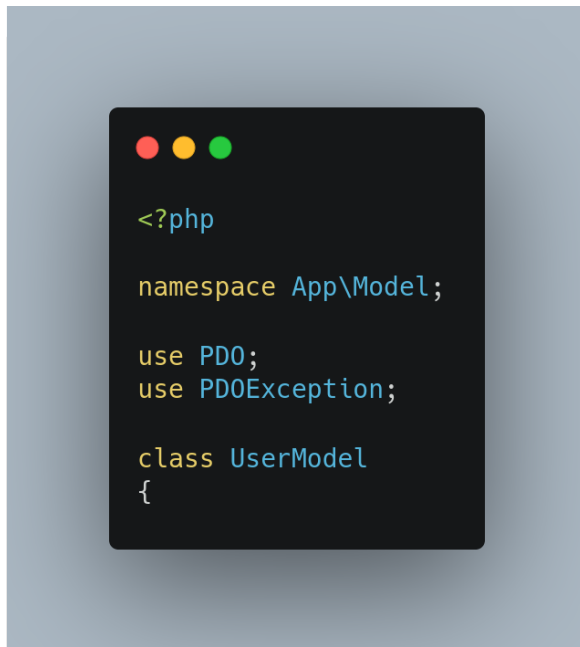
L'autoloader en PHP est utilisé pour charger automatiquement les classes lorsqu'elles sont utilisées, sans avoir à les inclure manuellement **avec la fonction** `require_once`. Cela facilite la gestion des dépendances et permet d'éviter des erreurs d'inclusion de fichiers.

Donc j'ai pu créer la fonction **my_autoloader** pour réaliser **l'autoloading** des classes.

- La fonction **my_autoloader** est déclarée avec un seul paramètre `$class`, qui correspond au nom de la classe que vous essayez d'utiliser dans votre code.
- **str_replace** pour remplacer tous les caractères `"\"` par `"/`, ce qui est nécessaire pour obtenir le chemin correct du fichier de classe dans le système de fichiers.
- On remplace la chaîne `"App"` par `"src"` dans le chemin de classe. Cela est fait pour respecter une structure de répertoires typique où les classes sont placées dans le répertoire `"src"` plutôt que dans le répertoire `"App"`.
- La variable **\$file** est construite en utilisant le chemin du répertoire courant (`__DIR__`) et le chemin de la classe modifié. Ainsi, **\$file** contient le chemin absolu du fichier de classe correspondant.
- On utilise à nouveau **str_replace** pour remplacer les caractères `"\"` par le séparateur de répertoires approprié (`DIRECTORY_SEPARATOR`).
- Enfin, la condition **if(file_exists(\$file))** vérifie si le fichier de classe existe (**return true**) réellement dans le système de fichiers. Si c'est le cas, la **fonction require_once** est utilisée pour inclure le fichier de classe.
- Enfin **spl_autoload_register("my_autoloader")** enregistre la fonction **my_autoloader** en tant que fonction d'autoloading, ce qui signifie que cette fonction sera appelée automatiquement chaque fois qu'une classe sera utilisée mais n'aura pas encore été incluse.

II - Classe UserModel / Connexion à la base de données

Pour interagir avec la base de données précédemment créée et effectuer des insertions de données, j'ai mis en place une classe appelée "UserModel" en respectant l'architecture MVC. Grâce à l'autoloader, j'ai créé un espace de noms (**namespace**) nommé "App\Model" afin de pouvoir utiliser la classe UserModel dans mon code en l'appelant correctement.

A code editor window with a dark background and light blue text. It contains PHP code for the UserModel class. The code starts with a PHP opening tag, followed by a namespace declaration 'App\Model', and then two 'use' statements for 'PDO' and 'PDOException'. Finally, it declares the 'UserModel' class and starts its opening curly brace.

```
<?php

namespace App\Model;

use PDO;
use PDOException;

class UserModel
{
```

À l'intérieur de notre classe, nous utilisons le constructeur qui est appelé chaque fois que notre classe est instanciée. Dans le constructeur, **nous instancions un nouvel objet PDO** afin d'établir la connexion à la base de données.

Dont l'utilisation d'un namespace PDO & PDOException est nécessaire .

II - Classe UserModel / Connexion à la base de données (suite)



```
<?php
private PDO $conn;

public function __construct()
{
    $db_username = 'root';
    $db_password = '';

    try {

        $this->conn = new PDO('mysql:host=localhost;dbname=boutique_en_ligne;charset=utf8',
            $db_username, $db_password);

        $this->conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    } catch (PDOException $e) {

        echo "Error : " . $e->getMessage();
    }
}
```

La variable ``$conn`` déclarée comme ``private PDO $conn;`` est une propriété privée de la classe ``UserModel`` qui représente la connexion à la base de données. Elle est de type ``PDO``, ce qui signifie qu'elle contiendra une instance de la classe PDO utilisée pour interagir avec la base de données.

L'utilisation de ``$this->conn`` fait référence à la propriété ``$conn`` de l'objet courant (instance de la classe). Lorsque le constructeur est appelé et qu'un objet ``UserModel`` est instancié, la connexion PDO est créée et stockée dans la propriété ``$conn`` de cet objet spécifique. Cette utilisation de ``$this`` permet d'accéder aux propriétés et méthodes de l'objet en cours d'utilisation.

III - Inscription (Model)

Il a été nécessaire de créer une méthode appelée `insertUserDb` pour insérer des utilisateurs dans la base de données à partir d'un formulaire de connexion préalablement créé. Cette méthode permet de prendre les données fournies par le formulaire et de les insérer dans la base de données afin de créer de nouveaux utilisateurs.

```
public function InsertUserDb($login, $email, $hash)
{
    $sql = "INSERT INTO user (`login`, `password`, `email`, `id_role`) VALUES (:login, :pass, :email, :id_role)";
    $req = $this->conn->prepare($sql);
    $req->execute(array(':login' => $login,
        ':pass' => $hash,
        ':email' => $email,
        ':id_role' => 1
    ));

    $sqlIdUser = "SELECT id FROM user WHERE login = :login";
    $reqIdUser = $this->conn->prepare($sqlIdUser);
    $reqIdUser->execute([':login' => $login]);
    $idUser = $reqIdUser->fetchAll(PDO::FETCH_ASSOC);

    $date = date('Y-m-d H:i:s');

    return 'Go Signup';
}
```

Méthode Insertion User

Requête préparée contre injections SQL

Requête SQL

Retourne un tableau associatif

- On définit **une requête SQL** qui utilise des paramètres nommés (**:login, :pass, :email, :id_role**) pour l'insertion des données dans la table "user".
- **La méthode `prepare`** de l'objet PDO (`$this->conn`) et stocke la préparation dans la variable `$req`.
- **La méthode `execute`** est appelée sur l'objet `$req` en passant un tableau associatif contenant les valeurs des paramètres (:login, :pass, :email, :id_role) pour effectuer l'insertion dans la base de données.
- La méthode retourne la chaîne de caractères **'okSignup'** pour indiquer que l'insertion de l'utilisateur a été effectuée avec succès.

III Inscription (Controller)

À l'aide de notre contrôleur, nous avons mis en place une méthode nommée `Register`. Cette méthode est responsable de la validation des données avant leur insertion dans la base de données. Pour ce faire, nous utilisons l'instance `\$this->model`, qui représente la classe `UserModel`. Nous faisons appel à la méthode `insertUserDb` de cette classe pour effectuer l'insertion effective des données de l'utilisateur dans la base de données.

```
public function Register(?string $login, ?string $email, ?string $password, ?string $passwordConfirm)
{
    $login = htmlspecialchars(trim($login));
    $email = htmlspecialchars(trim($email));
    $password = htmlspecialchars(trim($password));
    $passwordConfirm = htmlspecialchars(trim($passwordConfirm));

    $messages = [];

    $row = $this->model->RowCount('user', 'login', $login);

    if ($row <= 0 && strlen($login) >= 4 && !preg_match("[\W]", $login) && preg_match("/@/", $email) &&
    preg_match("/\./", $email) && strlen($password) >= 5 && $password == $passwordConfirm) {

        $hash = password_hash($password, PASSWORD_DEFAULT);

        $this->model->InsertUserDb($login, $email, $hash) == 'okSignup' ? $messages['okReg'] = 'Your account is
        now created and you can login' : $messages['errorRegDb'] = 'There was an error with the database insertion, please
        try again later';
    }
}
```

On peut observer l'utilisation des fonctions `htmlspecialchars` et `trim` pour **prévenir les failles de sécurité XSS et SQL**. De plus, il est essentiel de mettre en place le hachage du mot de passe en utilisant la fonction `password_hash`. Ces mesures de sécurité sont nécessaires pour protéger les données sensibles des utilisateurs contre les attaques potentielles.

III - Inscription (Controller)

```
    } else {  
        if ($row > 0) {  
            $messages['errorLoginExist'] = 'The login already exist. Please choose another one';  
        }  
        if (strlen($login) <= 3 || preg_match("[\W]", $login)) {  
            $messages['errorLogin'] = 'Your login must contain at least 4 characters and no specials characters';  
        }  
        if (!preg_match("/@/", $email) || !preg_match("/\./", $email)) {  
            $messages['errorEmail'] = 'Your email must contain a \'@\' and a \'.\'';  
        }  
        if (strlen($password) <= 4) {  
            $messages['errorPassLong'] = 'Your password must contain at least 5 characters';  
        }  
        if ($password != $passwordConfirm) {  
            $messages['errorPassMatch'] = 'The passwords do not match';  
        }  
    }  
  
    $json = json_encode($messages, JSON_PRETTY_PRINT);  
    echo $json;  
}
```

Une fois que toutes ces vérifications ont été effectuées, notamment en vérifiant l'existence de l'utilisateur dans la base de données à travers **la variable ` \$row `**, ainsi que les vérifications concernant la longueur du login et du mot de passe en **utilisant les fonctions `strlen()` et `preg_match`** pour vérifier les caractères, nous créons un tableau avec plusieurs index et valeurs pour renvoyer les messages appropriés.

Ce tableau est ensuite encodé **en JSON à l'aide de la fonction `json_encode`** afin de pouvoir l'exploiter dans la vue. En encodant les données en JSON, nous facilitons leur manipulation et leur utilisation dans le cadre de l'affichage et du traitement des messages dans la vue.

JEU D'ESSAIE

(Panier Utilisateur)

I - Panier (données d'entrée , données attendues)

La méthode ``AddProduct`` permet d'ajouter un produit au panier.



- La méthode commence par vérifier si le produit est déjà dans le panier en effectuant une requête SQL avec un compteur (``rowCount()``).
- Ensuite, elle récupère le prix d'un produit et le multiplie par la quantité pour obtenir le prix total des articles (``$priceOneItem * $quantity``).
- Si le produit n'est pas déjà dans le panier, elle l'ajoute à la table ``item_cart`` avec la quantité, le prix total et la plateforme.

```
if($row == 0) { ← Condition si la ligne retournée est strictement égale à 0 / return true
    // IF THE PRODUCT ISN'T IN THE CART, ADD IT //
    $sql = "INSERT INTO item_cart (id_cart, id_game, quantity, price, platform) VALUES (:cart, :idProduct,
:quantity, :price, :platform)";
    $req = $this->conn->prepare($sql);
    $req->execute([':cart' => $cartId,
        ':idProduct' => $idProduct,
        ':quantity' => $quantity,
        ':price' => $priceItems,
        ':platform' => $platform
    ]);

    // GET CART PRICE //
    $sqlGetPriceCart = "SELECT total_price FROM cart WHERE id = :cartId";
    $reqGetPriceCart = $this->conn->prepare($sqlGetPriceCart);
    $reqGetPriceCart->execute([':cartId' => $cartId]);

    $priceOneCart = $reqGetPriceCart->fetch(PDO::FETCH_ASSOC)['total_price'];

    $newCartPrice = $priceOneCart + $priceItems;

    // CHANGE CART PRICE //
    $sqlChangeCartPrice = "UPDATE cart SET total_price = :newPrice WHERE id = :cartId";
    $reqChangeCartPrice = $this->conn->prepare($sqlChangeCartPrice);
    $reqChangeCartPrice->execute([':newPrice' => $newCartPrice,
        ':cartId' => $cartId
    ]);

    $message = "The article was successfully added to the cart";
}
```

- Elle met également à jour le prix total du panier en ajoutant le prix des articles ajoutés (**`$newCartPrice = $priceOneCart + $priceItems`**).
- Enfin, elle renvoie un message indiquant que l'article a été ajouté avec succès.

```

} else { ← !true

    // IF THE PRODUCT IS ALREADY IN THE CART, GET DATA //
    $sqlId = "SELECT quantity, price, id FROM item_cart WHERE id_cart = :cart AND id_game = :idProduct AND
platform = :platform";

    $reqId = $this->conn->prepare($sqlId);
    $reqId->execute([':cart' => $cartId,
                    ':idProduct' => $idProduct,
                    ':platform' => $platform
    ]);

    $tab = $reqId->fetch(PDO::FETCH_ASSOC);

    // AND INCREASE QUANTITY AND PRICE //
    $quantityNew = $tab['quantity'] + $quantity;
    $priceNew = $tab['price'] + $priceItems;

    $sqlUp = "UPDATE item_cart SET quantity = :quantity, price = :newPrice WHERE id = :id";

    $reqUp = $this->conn->prepare($sqlUp);
    $reqUp->execute([':quantity' => $quantityNew,
                    ':newPrice' => $priceNew,
                    ':id' => $tab['id']
    ]);

    $message = "The article was successfully added to the cart";
}

return $message;

```

- Si le produit est déjà dans le panier, elle récupère les données existantes (quantité, prix) et les met à jour en ajoutant la nouvelle quantité et le nouveau prix.
- Enfin, elle renvoie un message indiquant que l'article a été ajouté avec succès.

II - Panier (donnée obtenue)

Une fois toutes les données entrées, nous souhaitons afficher la liste des articles **dans le panier côté client**. Pour cela, l'utilisation de **JavaScript** est nécessaire, en particulier l'utilisation de l'asynchronisme et de l'API Fetch pour effectuer des appels à nos pages.

Pour mettre en œuvre cette fonctionnalité, nous allons utiliser la fonction **`displayGames`** qui sera chargée d'afficher les articles du panier. Voici comment nous pouvons continuer le texte :



```
const getCartContentHTML = async() => {  
    const response = await fetch('cart.php?getCart=1');  
    const cartContentJSON = await response.text();  
    return cartContentJSON;  
}  
  
const displayGames = async() => {  
    const content = JSON.parse(await getCartContentHTML());  
    const main = document.getElementsByTagName('main')[0];  
    const divCartContent = document.getElementById('displayCartContent');  
    const divAllGames = document.createElement('div');  
    divAllGames.className = 'divAllGames';  
    divAllGames.innerHTML = "";  
    divCartContent.appendChild(divAllGames);  
}
```

Function async

Requête pour appeler de la page CART

Convertir en format Texte

Retourne un résultat sous format de données JSON

Convertir un JSON en chaîne de caractère


```
if(content['isEmpty'] === false) {
```

← Si condition retourne FALSE

```
  for (const game in content) {
```

← Itérations sur les propriétés de Content

```
    if (Object.hasOwnProperty.call(content, game)) {
```

← Si la propriété game appartient à content

```
      const element = content[game];
```

```
      if(game === "displayGame") {
```

```
        for (const gameHTML in element) {
```

```
          if (Object.hasOwnProperty.call(element, gameHTML)) {
```

```
            const htmlGame = element[gameHTML];
```

```
            divAllGames.innerHTML = divAllGames.innerHTML + htmlGame;
```

```
          }
```

```
        }
```

```
      }
```

```
    }
```

```
  }
```

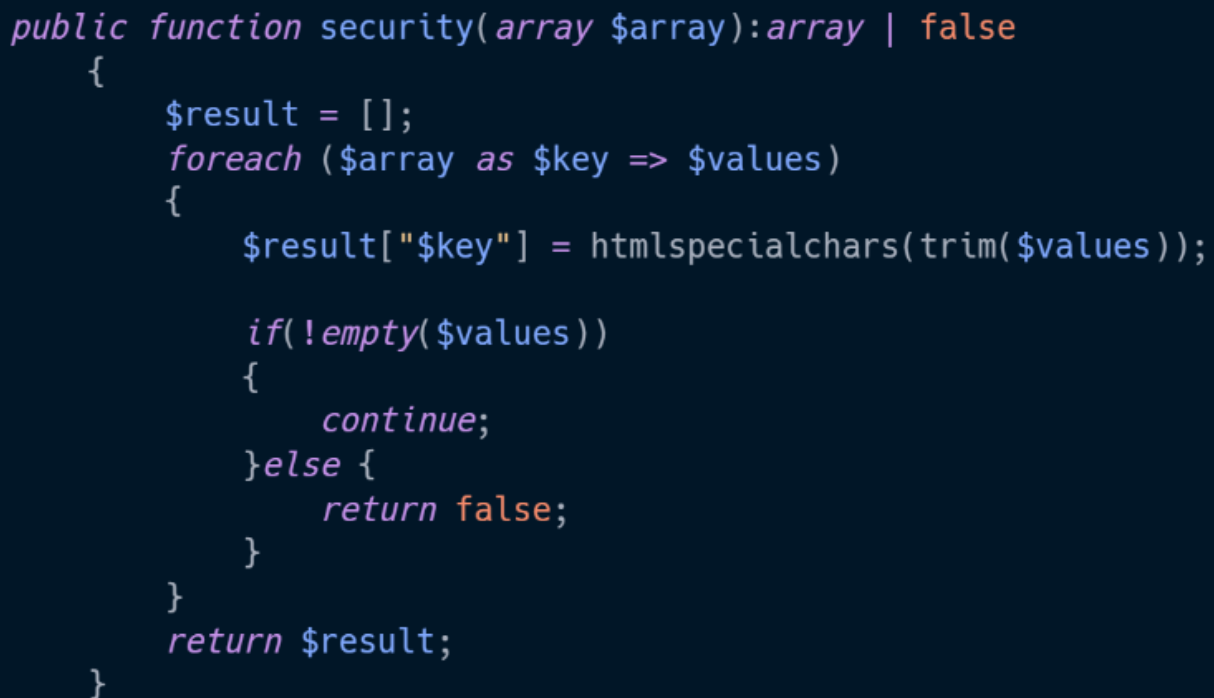
Veille durant le projet & Vulnérabilités de Sécurité

Durant le projet de conception de notre boutique en ligne, il était essentiel de rester à jour avec les dernières avancées technologiques. Pendant cette période, j'ai effectué une veille technologique continue pour découvrir de nouvelles fonctionnalités que je souhaitais intégrer à notre application. **Stack Overflow** a été une ressource précieuse, car j'ai pu y trouver des réponses et des solutions à des problèmes spécifiques rencontrés lors du développement.

L'une des préoccupations majeures était de garantir la sécurité de notre application. J'ai consacré du temps à rechercher et comprendre les meilleures pratiques pour prévenir les attaques telles que les injections SQL, les failles XSS et les attaques de force brute. J'ai mis en place des mesures de protection adaptées à notre boutique en ligne, notamment :

- **Protection contre les injections SQL** : J'ai utilisé des requêtes préparées avec des paramètres liés pour empêcher les attaques par injection SQL. Cela garantit que les données sont correctement échappées et que seuls les paramètres sécurisés sont inclus dans les requêtes SQL.
- **Prévention des failles XSS** : J'ai utilisé des fonctions de filtrage et d'échappement des données pour éviter les attaques XSS. Par exemple, j'ai utilisé **la fonction ``htmlspecialchars``** pour convertir les caractères spéciaux en entités HTML et ainsi éviter toute exécution de code potentiellement dangereux.
- **Protection contre les attaques de force brute** : J'ai mis en place des mesures de sécurité pour limiter le nombre de tentatives de connexion et pour détecter et bloquer les activités suspectes. J'ai également encouragé l'utilisation de mots de passe forts et recommandé aux utilisateurs de mettre en place des politiques de sécurité strictes.

En somme, grâce à une veille technologique proactive et à des recherches approfondies, j'ai pu mettre en œuvre des mesures de sécurité adéquates dans notre boutique en ligne, tout en intégrant les dernières fonctionnalités pour améliorer l'expérience utilisateur. La sécurité des données et la protection contre les vulnérabilités sont des éléments cruciaux dans le développement d'une boutique en ligne, et j'ai pris les mesures nécessaires pour garantir la fiabilité et la sécurité de notre application.



```
public function security(array $array):array | false
{
    $result = [];
    foreach ($array as $key => $values)
    {
        $result["$key"] = htmlspecialchars(trim($values));

        if(!empty($values))
        {
            continue;
        }else {
            return false;
        }
    }
    return $result;
}
```

Description d'une situation de travail

(Avec une recherche sur un site anglophone)

Durant le projet de conception de notre boutique en ligne, il était crucial de prendre en compte les failles de sécurité, notamment les failles XSS. Dans ce contexte, j'ai réalisé des recherches approfondies pour comprendre les meilleures pratiques et les mesures de prévention nécessaires. Une partie de mes recherches s'est effectuée sur des sites anglophones, et cela s'est avéré extrêmement bénéfique.

En effet, les sites anglophones offrent une richesse d'informations beaucoup plus étendue que les sites francophones dans le domaine de la sécurité web. J'ai pu y trouver des ressources, des études de cas, des exemples concrets et des discussions approfondies sur les vulnérabilités XSS et les méthodes de protection. La communauté anglophone, notamment sur des plateformes comme **StackOverflow**, propose des réponses détaillées et des solutions pratiques pour résoudre les problèmes de sécurité.

3 Answers

Sorted by: Highest score (default) ▾



8

You should not encode HTML-Specialchars when inserting into database, that way data is manipulated (and maybe different when editing the dataset). You should rather encode them when displaying it.



But yes, `htmlspecialchars()` is enough to prevent XSS as long as you don't forget to use it. The way YOU use it however is as secure as before. XSS is prevented through the encoded version, the database does not care about it.



Share Improve this answer Follow

edited Jan 19, 2012 at 20:04

answered Jan 19, 2012 at 19:58

Merci

Mehdi Romdhani

***Annexes**

Voir PDF (car les dimensions ne sont pas adéquats)