Documentation technique du projet d’Infrastructure

par Hénan NOËL

Développement d’un site Web dynamique et responsive en PHP et *HTML et avec MariaDB*

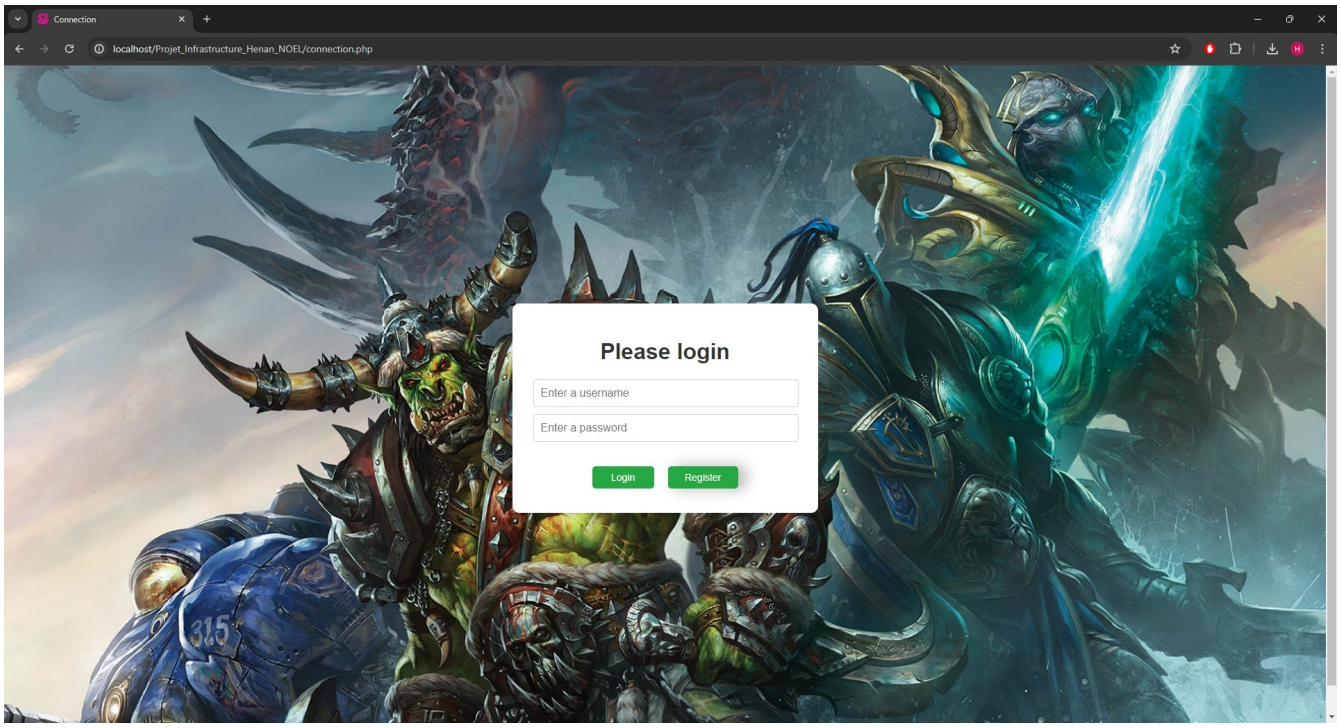


Table des Matières

I – Introduction

Bienvenue dans la documentation technique de ce projet d'Infrastructure. Cette documentation vise à fournir une compréhension complète de l'architecture, de la mise en place et de l'utilisation de mon site web dynamique développé en PHP et HTML, avec une base de données MariaDB hébergée sur PHPMyAdmin via XAMPP.

II – Infrastructure du projet

1. Serveur local

Afin de tester ce projet et de le réaliser, je me suis appuyé sur le serveur *Apache* et le module *MySQL* intégrés dans XAMPP. Le serveur *Apache* me permet de run mon site en localhost et ainsi faciliter le développement de celui-ci.

2. La base de données

Afin de gérer ma base de données, j'ai utilisé *MariaDB* via *PHPMyAdmin*. *PHPMyAdmin* étant une interface web graphique me permettant de gérer ma base de données, cela m'a offert la possibilité d'exécuter mes requêtes SQL plus facilement, sans avoir avoir à tout coder de A à Z.

3. Le front end

Pour gérer la structure visuelle du site, ainsi que l'*UX* (*User Experience*) et l'*UI* (*User Interface*), je me suis appuyé sur du *HTML* et du *CSS*. Ces deux langages m'ont permis de proposer un site web propre et agréable à utiliser, sans créer de frustrations ou de conflits lors de son utilisation.

4. Le back end

Afin de créer mon site, de récupérer les données ou bien encore de gérer les requêtes, j'ai fais le choix d'utiliser *PHP*. C'est le langage principale utilisé pour le développement de ce projet afin de générer des pages web dynamiques, de gérer les formulaires, accéder à la base données, gérer les requêtes entre le serveur et la base de données…

5. XAMPP

XAMPP est une suite de logiciel fournissant un environnement de développement local complet pour *PHP*, *Apache* et *MariaDB*. Il m'a permis de tester mon site localement afin pouvoir le déployer sur un serveur plus tard.

En utilisant ces différents langages et logiciels, j'ai pu ainsi créer un site web dynamique et interactif répondant aux besoins de ses utilisateurs et aux contraintes imposées.

III – Communication avec la base de données

1. Configuration

Afin de garantir la communication avec la base de données, j'ai utilisé *PHP* pour créer le fichier permettant ce lien. J'ai donc décidé de créer un fichier db_config.php qui permet de récupérer les « *credentials* », ce qui correspond à l'ensemble des informations de l'utilisateur sur sa base de données (via des variables : nom de la base de données, nom d'utilisateur, mot de passe et serveur/adresse IP). Ce fichier étant un fichier personnel, le séparer de db_connect.php est nécessaire afin de garantir la sécurité de nos données personnelles.

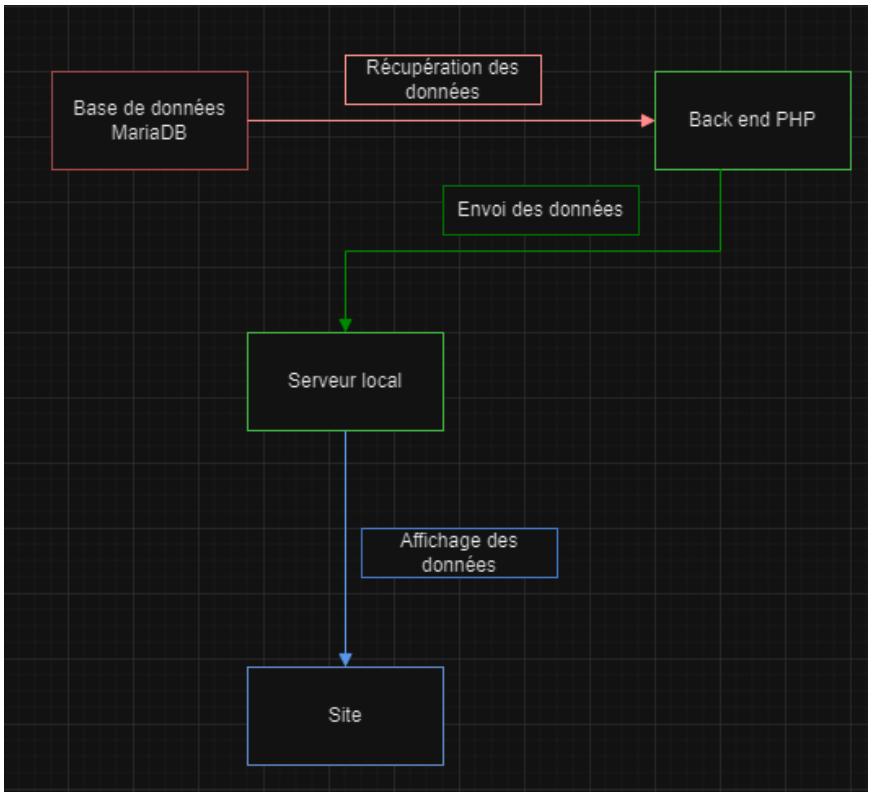
Une fois ce fichier prêt, j'ai configurer la connexion à la base de données via le fichier db_connect.php, qui vient récupérer les variables définies afin de permettre l'accès à la base de données en toute sécurité. Afin de garantir cette connexion sur chaque page, j'ai utilisé l'attribut *required_once* en début de chaque code, permettant ainsi de venir récupérer les données nécessaires.

2. Gestion des erreurs

Lors de la connexion ou durant la communication, les erreurs sont gérées via des messages d'alertes et renvoi d'erreurs permettant de les identifier. Si un ID de jeu n'est pas reconnu ou si une donnée manquante est présente, un message d'erreur s'affichera afin de détecter d'où vient le problème, et ce à chaque action sur le site.

Ce principe fonctionnement essentiellement sur le principe de boucle try-catch, qui me servent à détecter les erreurs de *PDO* (PHP Data Object) lors de la connexion entre la base de données et le serveur. Cette boucle essaye de se connecter, et si une erreur est rencontrée ou qu'une donnée devient nulle, le code compris dans le catch va renvoyer le message d'erreur correspondant.

3. Illustration du fonctionnement



IV – Ports et sécurité

1. Ports utilisés

Tout d'abord, XAMPP me permet d'avoir un serveur *Apache* utilisant le port 80 afin de me connecter en local. Ce port étant celui alloué par défaut, je ne l'ai pas changé.

Concernant *MySQL*, son port par défaut est le 3306, que je n'ai pas changé par soucis de risques de bugs. Ce port me permet de me connecter à ma base de données librement, sans créer de conflits avec d'autres ports.

2. Sécurité

La sécurité que j'ai appliqué sur mon site consiste en un système d'authentification, via la base de données. Si un utilisateur malveillant cherche à utiliser le site sans avoir de compte préalablement créé et enregistré dans la base de données, celui-ci se verra interdire l'accès au site.

Ceci permet de garantir la sécurité des données contenues sur le site. En effet, chaque page redirigera automatiquement vers la page de connexion (*connection.php*) si aucune session en cours n'est détectée. Ce principe est possible via la fonction *PHP* intégrée *session_start()*, qui vérifie si un utilisateur est actuellement connecté ou non.

index.php
```php
<?php
session_start();
require_once "./db_connect.php";
```

De plus, la sécurité des données utilisateurs est garantie par le principe de *hash* de leurs mots de passe. Le *hash* permet de chiffrer les mots de passe dans la base de données, une fois que l'utilisateur a été créé. Ce chiffrement, qui est différent à chaque saisie du mot de passe, garanti la sécurité des données de l'utilisateur, car je ne connaîtrais pas le mot de passe saisi et enregistré, bien que j'ai accès à la base de données. De ce fait, je saurais seulement quels utilisateurs se sont enregistrées, sans en connaître plus. Afin de vérifier si le *hash* généré correspond à celui enregistré, j'ai utilisé la méthode de *password_verify()* qui permet cette vérification en temps réel.

Le principe de *hash* le mot de passe créé :

```php
$passwordHash = password_hash($password, PASSWORD_BCRYPT);
$sql = "INSERT INTO auth(username, password) VALUES(:username, :password);";
```

Le principe de vérifier le *hash* généré par la saisie :

```php
if ($user) {
    // User found, check the password
    if (password_verify($password, $user['password'])) {
        // Correct password, start session and redirect to index.php
        $_SESSION['username'] = $username;
        header("Location: index.php");
        exit();
```

Enfin, afin de garantir la sécurité du site web, si une saisie d'utilisateur ou de mot de passe est incorrecte, l'alerte renvoyée est la suivante : « Username or password is wrong. Please try again ».

```php
<?php
    if (isset($_GET['error']) && $_GET['error'] == 'login_failed') {
        echo "<script>alert('Username or password is wrong. Please try again.'); window.location = './connection.php';</script>";
    }
?>
```

Ce principe ralentit la pénétration d'un utilisateur malveillant en augmentant le temps de *brute force* des informations.

V. Détails de l'utilisation des outils proposés

1. MariaDB

Ce système de gestion de base de données m'a permis d'organiser les données de mes utilisateurs ainsi que des jeux enregistrés.

MariaDB m'a également permis de gérer ma base de données en sauvegardant toutes données nécessaires au bon fonctionnement de mon site, que ça soit les données de jeux ou les données utilisateurs pour se connecter au site.

Il m'a aussi facilité l'accès et l'utilisation des requêtes *SQL* afin de récupérer les données sur mon serveur via mon code *PHP* afin de gérer les connexions.

2. PHP

PHP a été le langage de code principalement utilisé pour gérer les interactions entre la base de données et le serveur, ainsi que pour gérer l'affichage dynamique du site et des erreurs pouvant être rencontrées.

Ce langage me permet également d'intégrer des fonctions que je n'ai pas a développer moi-même telles que *password_hash*, *password_verify* ou encore *start_session*(), qui sont toutes des fonctions intégrées permettant de garantir la sécurité du site.

Enfin, *PHP* m'a servi a gérer mes alertes HTML garantissant le bon fonctionnement ou non de certaines actions sur le site, en fonction des envois de données sur la base de données.

3. HMTL/CSS

J'ai utilisé ces deux langages pour structurer et styliser mon site web. Afin de garantir une expérience agréable pour les utilisateurs, *HTML* m'a permis de structurer le site, proposer un site compréhensible aux personnes malvoyantes ou non-voyantes grâce à une structure de code propre, pouvant être lue par synthèse vocale. Ce principe garanti l'accessibilité du site à tous.

Le *CSS* m'a permis de styliser le site, le rendre responsive, et proposer une *UX* (*User Experience*) et une *UI* (*User Interface*) optimale aux utilisateurs. Je m'en suis servi afin que mon site soit le plus optimisé possible pour tous, sur tout type de plateforme.

4. XAMPP

XAMPP m'a donné un accès à mon serveur local afin de travailler et de tester les différentes fonctionnalités et les différents liens entre la base de données et mon back end. J'ai pu me servir de ce logiciel pour développer mon site en toute sérénité, avant de le déployer sur un serveur en ligne.

VI. Conclusion

Ce projet de site web garanti donc l'expérience utilisateur dans son intégralité ainsi que son accessibilité. Grâce aux différentes méthodes et outils mis en place tout au long de ce projet, j'ai pu garantir une sécurité des données utilisateurs basique ainsi qu'un accès à une base de données qui peut être remplie sans passer par des requêtes *SQL* faites nous-même.

L'utilisation de *PHP* avec *HTML* permet également de proposer un site fluide à son utilisateur afin de réduire l'impression d'attente de celui-ci.

J'ai pu apprendre beaucoup grâce à ce projet : un nouveau langage (*PHP*), l'utilisation d'une base de données et sa manipulation, ainsi que la gestion des utilisateurs et de leur sécurité.

Le prochain objectif est désormais de mettre en place un système de sauvegarde automatique et un système de récupération en cas de perte des données ou de bug dans la base de données.