Dossier de Projet

Création d'un site web pour une communauté en ligne multi-gaming

Fait par David Buléon

Période de stage du 09/05/2023 au 21/07/2023

Table des matières

Introduction

Dans le cadre de ma reconversion professionnelle après six ans dans le secteur de l'hôtellerie-restauration, j'ai fait le choix d'entreprendre une formation AFPA pour devenir développeur web et web mobile. Cette décision découle de ma volonté de me rapprocher de mes passions et de mieux appréhender le monde qui m'entoure, dans une époque emplie de nouvelles technologies.

Mon objectif suivant est de continuer à approfondir mes connaissances dans ce vaste domaine en poursuivant mon apprentissage en alternance au sein d'un environnement professionnel riche en expérience.

La formation pour le titre de développeur web / web mobile s'est conclu par la réalisation d'un projet en autonomie, en travaillant à distance en télétravail et le présent dossier vise à présenter le projet d'un point de vue technique et fonctionnel.

Compétences du référentiel couvertes par le projet

1. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

2.Maquetter une application.

3. Réaliser une interface utilisateur web statique et adaptable.

4. Développer une interface utilisateur web dynamique.
Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce.

5. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

6.Créer une base de données.

7Développer les composants d'accès aux données.

8Développer la partie back-end d'une application web ou web mobile.

9Elaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce.

Présentation du projet

Le client

J'ai réalisé ce projet pour les "Indies", une communauté en ligne multi-gaming et e-sportive qui participe à de nombreuses activités en ligne, principalement centrées sur le gaming.

Cette communauté à but non lucratif, à laquelle j'appartiens depuis plus de 5 ans maintenant, a pour objectif premier de rassembler des joueurs principalement francophones afin de partager leurs passions communes, telles que les jeux vidéo, mais aussi le développement informatique, le matériel informatique, le monde militaire, les cryptomonnaies, et bien d'autres sujets.

Ces derniers possèdent plusieurs serveurs communautaires présents sur certains jeux tels que "Squad", "Valheim", "Project Zomboid", ainsi que sur des outils de communication comme "Discord" ou "TeamSpeak". Ils organisent régulièrement différents événements "In game", tels que des rencontres et des tournois opposant plusieurs équipes les unes aux autres dans des compétitions. De plus, ils mettent en place des événements scénarisés, encadrés par leurs soins, pour le plus grand plaisir de la communauté. Ils offrent également divers services, comme la traduction des mises à jour de "Squad" ou l'organisation de tirages au sort et de loteries permettant de gagner des jeux gratuits, au grand bonheur des milliers de personnes y participant.

Cahier des charges

Les "Indies" ont fait appel à mes services pour développer leur futur site web. Ce dernier aura deux objectifs bien définis :

Tout d'abord, il servira de site vitrine pour offrir une visibilité accrue à la communauté. Ce site permettra de présenter l'équipe dans son ensemble ainsi que de partager leurs réseaux sociaux. Il constituera également une plateforme pour publier des actualités concernant la communauté et diffuser des vidéos et des images en lien avec leurs activités.

Ensuite, le site aura pour but de développer des outils facilitant l'organisation interne de la communauté. Parmi les fonctionnalités prévues, on retrouvera un outil de création d'événements communautaires avec la possibilité pour les utilisateurs de s'y inscrire. De plus, un outil dédié à la création de matchs pour les responsables sera mis en place, offrant aux membres des "Indies" la possibilité de s'inscrire aux différentes rencontres.

Grâce à ce site web, les "Indies" pourront mieux se faire connaître tout en optimisant leur organisation interne pour le plus grand plaisir de leur communauté passionnée

Scénario

Afin d'établir un cahier des charges plus complet j'ai procéder à une analyse des besoins en simulant les différents scenarios affins d'en faire ressortir les entités pour ma future base de données (voir page suivante) :

PARTIE ADMINISTRATEUR

Inscription / désinscription sur le site / connexion au site
Accès au profile du membre connecté
CRUD sur les utilisateurs
Les rôles des membres sont attribué par les administrateurs
ACCUEIL
Peut naviguer sur la page accueil
COMPETITION
Peut naviguer sur la page Match
CRUD sur le match
NEWS
Peut naviguer sur la page news
CRUD sur les news
MEDIA
Peut naviguer dans les media
CRUD sur les media
Possibilité de liker et commenter
EVENEMENT
Peut naviguer sur la page évènement
CRUD sur les évènements
Peut s'inscrire à un évènement

PARTIE VISITEUR

Inscription / désinscription sur le site / connexion au site
Accès au profile du membre connecté
ACCUEIL
Peut naviguer sur la page accueil
COMPETITION
Peut naviguer sur la page Match
NEWS
Peut naviguer sur la page new
MEDIA
Peut naviguer dans les media
Possibilité de liker et commenter
EVENEMENT
Peut naviguer sur la page évènement
Peut s'inscrire à un évènement

PARTIE INDIES

Inscription / désinscription sur le site / connexion au site
Accès au profile du membre connecté

ACCUEIL

Peut naviguer sur la page accueil

COMPETITION

Peut naviguer sur la page Match
Peut s'inscrire à un match

NEWS

Peut naviguer sur la page new

MEDIA

Peut publier un Media
Peut naviguer dans les media
Possibilité de liker et commenter

EVENEMENT

Peut naviguer sur la page évènement
Peut s'inscrire à un évènement

PARTIE PARTENAIRE CREATEUR

Inscription / désinscription sur le site / connexion au site
Accès au profile du membre connecté

ACCUEIL

Peut naviguer sur la page accueil

COMPETITION

Peut naviguer sur la page Match
Peut s'inscrire

NEWS

Peut naviguer sur la page news

MEDIA

Peut naviguer dans les media
Peut publier un Media
Possibilité de liker et commenter

EVENEMENT

Peut naviguer sur la page évènement

peut publier un évènement
Peut s'inscrire à un évènement

Les Entité

Voici les Entités qui en sont ressortie auxquelles j'ai ajouté des attributs en fonction des besoins.

Utilisateur

id_utilisateur INT AUTO_INCREMENT

nom VARCHAR (20)

email VARCHAR (50) NOT NULL

password VARCHAR (300) NOT NULL

id_role INT NOT NULL

Rôle

id_role INT AUTO_INCREMENT

nom_role VARCHAR (20) NOT NULL

News

id_news INT AUTO_INCREMENT

titre VARCHAR(50) NOT NULL

image VARCHAR(400)

article VARCHAR(500)

news_date DATE DEFAULT CURRENT_DATE()

Image (pour les media)

id_images INT AUTO_INCREMENT

titre VARCHAR(50) NOT NULL

contenu VARCHAR(400) NOT NULL

date_publication DATE DEFAULT CURRENT_DATE()

Vidéo (pour les media)

id_video INT AUTO_INCREMENT

titre VARCHAR(50) NOT NULL

contenu VARCHAR(400) NOT NULL

description VARCHAR(300)

date_publication DATE DEFAULT CURRENT_DATE()

Vidéo j’aime (pour pouvoir liker les vidéo)

id_jaime_video INT AUTO_INCREMENT

Note : J'ai créé spécialement cette table et la suivante pour plusieurs raisons que voici

1. Gestion des métadonnées spécifiques : stockage des métadonnées liées aux "j'aime" pour chaque type de média. Par exemple, enregistrer la date à laquelle un utilisateur a aimé une vidéo.

2. Performance : optimiser les performances de certaines requêtes spécifiques. Par exemple, afficher les "j'aime" d'un utilisateur sur ses vidéos et ses images, en effectuant des requêtes plus ciblées et efficaces en interrogeant directement les tables "jaime_video" et "jaime_image", plutôt que de devoir filtrer les enregistrements dans une seule table de liaison.

3. Séparation des préoccupations : séparez les préoccupations liées aux différentes fonctionnalités de votre application. Cela rend le code plus modulaire et facilite la maintenance et l'extension du système.

4. Possibilité de développement futur

Image j’aime

id_jaime_image INT AUTO_INCREMENT

Compétition

id_competition INT AUTO_INCREMENT

titre VARCHAR(50) NOT NULL

description VARCHAR(300)

date_publication DATE DEFAULT CURRENT_DATE()

date_competition DATE

Évènement

id_evenement INT AUTO_INCREMENT

titre VARCHAR(50) NOT NULL

date_publication DATE DEFAULT CURRENT_DATE()

date_evenement DATE

Commentaire

id_commentaire INT AUTO_INCREMENT

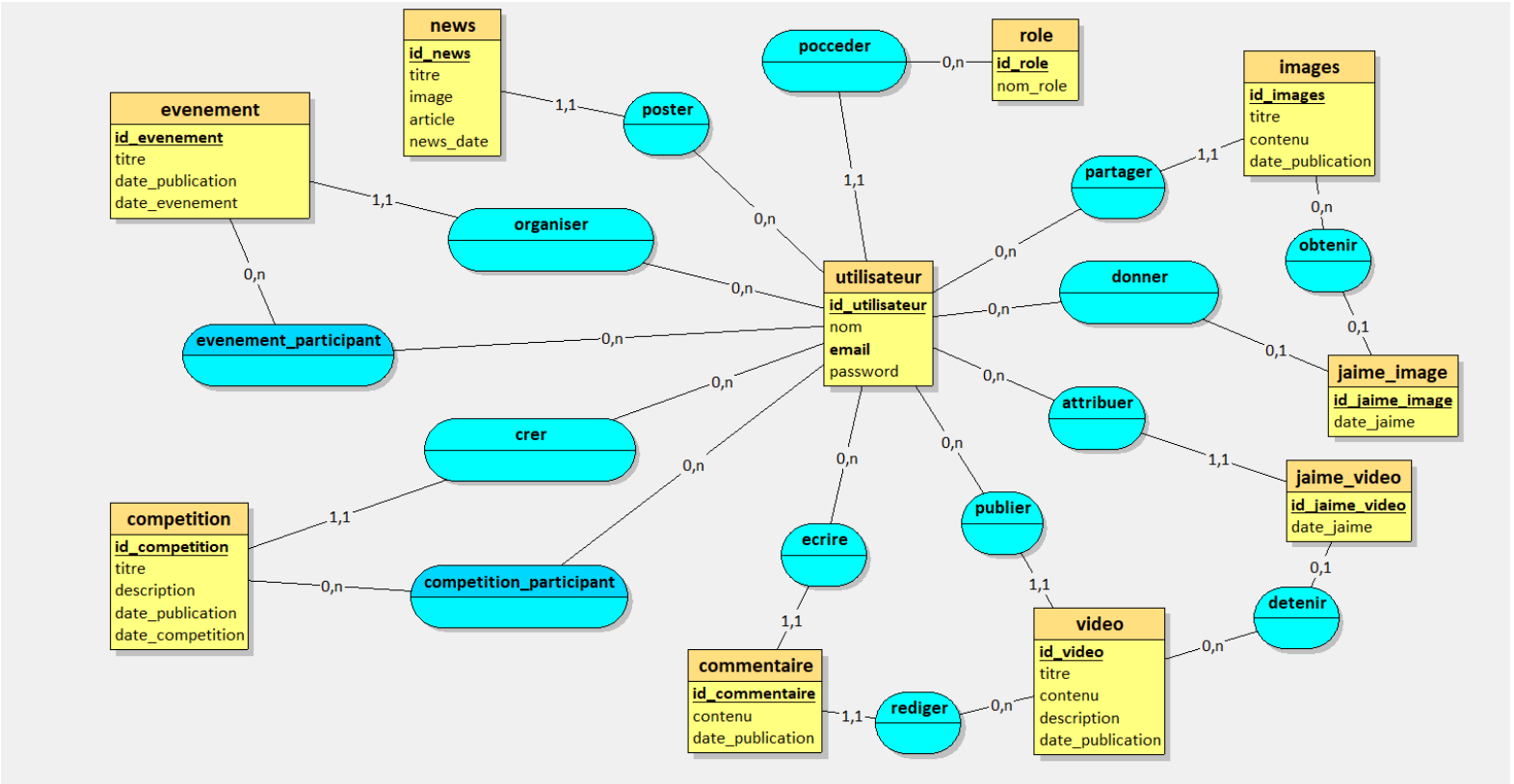
contenu VARCHAR(300)

date_publication DATE DEFAULT CURRENT_DATE()

Conception de la base de données

Après avoir rassemblé toutes les informations nécessaires, j'ai utilisé le logiciel de modélisation conceptuelle de données appelé "Looping" pour faciliter cette tâche. Grâce à cet outil, j'ai pu définir de manière simple et efficace les entités, leurs attributs et les relations entre elles.

Voici le modèle conceptuel de données (MCD) une fois les relations et cardinalités correctement définies :



« Capture d'écran du MCD »

Relations et cardinalités

Table "role":

- Relation : One-to-Many avec la table "utilisateur".

Table "utilisateur":

- Relation : Many-to-One avec la table "role".

- Relation : Many-to-Many avec les tables "images", "video", "news", "evenement", "competition", "jaime_image", "jaime_video", "commentaire", "evenement_participant" et "competition_participant".

Table "images":

- Relation : Many-to-One avec la table "utilisateur".

4. Table "video":

- Relation : Many-to-One avec la table "utilisateur".

- Relation : Many-to-Many avec la table "commentaire".

5. Table "news":

- Relation : Many-to-One avec la table "utilisateur".

6. Table "evenement":

- Relation : Many-to-One avec la table "utilisateur".

- Relation : One-to-Many avec la table "evenement_participant".

7. Table "competition":

- Relation : Many-to-One avec la table "utilisateur".
- Relation : One-to-Many avec la table "competition_participant".

8. Table "jaime_image":

- Relation : Many-to-One avec la table "utilisateur".
- Relation : Many-to-One avec la table "images".

9. Table "jaime_video":

- Relation : Many-to-One avec la table "utilisateur".
- Relation : Many-to-One avec la table "video".

10. Table "commentaire":

- Relation : Many-to-One avec la table "utilisateur".
- Relation : Many-to-One avec la table "video".

11. Table "evenement_participant":

- Relation : Many-to-One avec la table "utilisateur".
- Relation : One-to-Many avec la table "evenement".

12. Table "competition_participant":

- Relation : Many-to-One avec la table "utilisateur".
- Relation : One-to-Many avec la table "competition".

Une fois la modélisation terminée, j'ai récupéré le code SQL pour y effectuer quelques modifications manuelles. J'ai modifié les attributs "date_publication" en y ajoutant une

valeur par défaut, en utilisant la date du jour de la publication grâce à l'instruction DEFAULT CURRENT_DATE().

Ensuite, j'ai créé une nouvelle base de données sur l'interface PHPmyAdmin et utilisé ce code SQL pour générer ma base de données.

Maquettage

Dans le cadre de mon projet et après avoir consulté les différents responsables de la communauté "Indies" concernant le cahier des charges du site, j'ai procédé à la réalisation des maquettes de plusieurs pages, à la fois statiques et dynamiques. Pour faciliter cette tâche, j'ai utilisé un outil numérique de conception graphique appelé "Canva".

Voici des exemples des différentes maquettes réalisée :

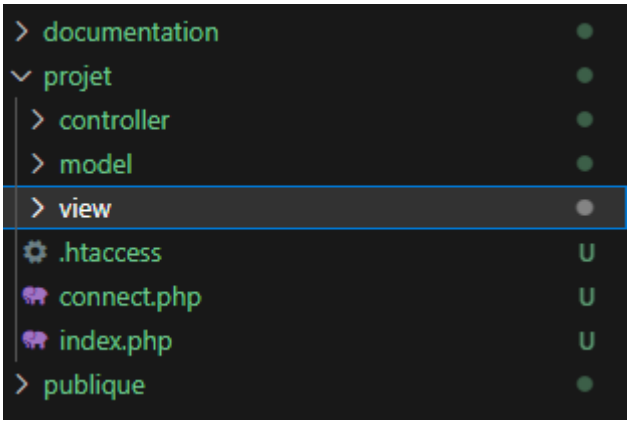
« Capture d'écran, maquette de la page d'accueil statique réalisé avec Canva »

« Capture d'écran, maquette de la page news dynamique réalisé avec Canva »

Architecture

Méthode MVC

Pour réaliser ce projet, avec une base de données SQL j'ai choisi d'utiliser PHP en méthode MVC. J'ai choisi PHP avec l'architecture MVC pour mon projet en raison de sa séparation claire des responsabilités, de sa structure organisée et évolutive, de sa large adoption avec une documentation riche, de sa compatibilité avec les serveurs web courants et de son coût avantageux. Cette combinaison ma permis de développer efficacement une application web robuste, facile à maintenir et évolutive tout en bénéficiant du soutien d'une grande communauté de développeurs.



(Capture d'écran architecture du projet)

Comme l’indique la capture d’écran précédente voici l’architecture que j’ai mis en place. Tout d’abord on peut observer le dossier Controller qui traite la logique métier, le Model qui gère l’accès aux données, puis Vue qui s’occupe de l’affichage.

Index.php

On peut également observer un fichier nommer index.php : L'index agit comme un routeur pour gérer les demandes de mon application. Lorsqu'un utilisateur accède à une page du site, l'URL est analysée pour extraire l'action spécifiée. L'action correspond à une opération que l'utilisateur souhaite effectuer, telle que "afficher un produit" ou "modifier un profil".

Une fois l'action extraite, le routeur la découpe en parties en utilisant le caractère '/' comme délimiteur. Le premier morceau est considéré comme le nom du contrôleur à inclure, et le deuxième morceau, s'il existe, est considéré comme la fonction à appeler dans ce contrôleur.

Le routeur inclut alors le fichier du contrôleur approprié en fonction de l'action. Si la fonction correspondante est trouvée dans le contrôleur, elle est appelée, éventuellement avec un argument passé dans l'URL. Si l'action ou la fonction n'existe pas, le routeur affiche un message d'erreur "404 - Page not found".

Cette approche facilite la gestion des différentes actions de l'application web, permettant ainsi de rediriger les utilisateurs vers les fonctionnalités appropriées en fonction des URL

qu'ils visitent.

```php
<?php

// Définition de la constante _BASE avec le chemin de base du projet
define("_BASE", "/Dev/Indies/projet");

// Fonction pour afficher le contenu d'une variable de manière formatée
function dbg($d) {
    echo "<pre>";
    print_r($d);
    echo "</pre>";
}

// Vérification de l'action spécifiée dans l'URL (GET)
$action = isset($_GET['action']) ? $_GET['action'] : null;

// Vérification si une action est définie
if ($action != null) {
    // Séparation de l'action en morceaux en utilisant le délimiteur "/"
    $tab = explode('/', $action);

    // Le premier élément de $tab représente le nom du contrôleur à inclure
    $controller = $tab[0];

    // Le deuxième élément, s'il existe, représente la fonction à appeler dans le contrôleur
    $commande = isset($tab[1]) ? $tab[1] : 'index'; // Si aucune fonction spécifiée, utilise 'index'

    // Inclusion du fichier du contrôleur
    require('controller/'.$controller.'.php');

    // Vérification si la fonction spécifiée dans $commande existe dans le contrôleur inclus
    if (function_exists($commande)) {
        // Si un argument est spécifié dans l'URL, on le récupère, sinon on le met à NULL
        $arg = isset($tab[2]) ? $tab[2] : NULL;

        // Appel de la fonction avec ou sans argument
        if (isset($arg)) {
            $commande($arg);
        } else {
            $commande();
        }
    } else {
        // La fonction spécifiée n'existe pas dans le contrôleur
        echo '404 - Page not found';
    }
} else {
    // Aucune action spécifiée dans l'URL
    echo '404 - Page not found';
}
?>
```

.htaccess :

Le fichier .htaccess est utilisé pour configurer les règles de réécriture d'URL dans le serveur Apache. Ces règles permettent de rediriger les requêtes entrantes vers le fichier index.php tout en modifiant l'apparence de l'URL pour une meilleure convivialité et une lisibilité améliorée.

```
1   RewriteEngine on
2   RewriteRule ^([a-zA-Z0-9\-\_\/]*)$ index.php?action=$1
```

(Capture d'écran fichier htacces)

Connect.php

Ce code PHP établit une connexion à la base de données MySQL en utilisant l'objet PDO. Il définit les informations nécessaires pour la connexion (nom de la base de données, hôte, nom d'utilisateur, mot de passe), puis configure PDO pour générer des exceptions en cas d'erreurs. Cette connexion PDO établie peut ensuite être utilisée pour exécuter des requêtes SQL et interagir avec la base de données dans le reste de l'application.

Production

Une fois toute cette architecture mise en place et ma base de données opérationnel pour démarrer, j'ai commencé la réalisation de l'application.

Voici quelques exemples de production coter back-end et front-end.

Exemple 1

Publier des news

J'ai créé une page en PHP qui contient un formulaire HTML avec une méthode POST. Afin de garantir la sécurité de cette page, j'utilise la variable superglobale PHP « $_SESSION » pour stocker des informations sur les utilisateurs et ainsi pouvoir identifier leur rôle. Je tiens à préciser que seuls les utilisateurs ayant le rôle d'administrateur auront accès au contenu de cette page. L'administrateur peut ainsi remplir le formulaire

Lorsque le formulaire est soumis, ses données sont envoyées à l'adresse suivante : <?php echo _BASE; ?>/admin_news/publier_article/. Cette adresse correspond à mon contrôleur "admin_news", qui exécute la fonction "publier_article". Le contrôleur effectue plusieurs vérifications.

Tout d'abord, il vérifie si le formulaire a bien été soumis. Ensuite, il récupère les données saisies dans les champs et procède au traitement de l'image fournie, vérifiant sa taille et son extension, puis lui attribue un nom généré. Cette image est ensuite enregistrée côté serveur dans un dossier spécifique par téléversement (upload).

Enfin, le contrôleur appelle un modèle qui se charge d'enregistrer les données dans la base de données en utilisant une requête SQL.

En suivant cette approche, les administrateurs pourront publier des actualités pour leur communauté en toute sécurité, en veillant à ce que seules les données valides soient enregistrées dans la base de données.

```php
<?php
require('view/header.php');
if (session_status() !== PHP_SESSION_ACTIVE) session_start();
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Creation-article-Indies</title>
</head>

<body>
    <?php
    if (!isset($_SESSION['id'])) {
        exit("Veuillez vous authentifier");
    } elseif ($_SESSION['role'] !== 1) {
        exit("Accès refusé");
    } else {
    ?><div id="news">
            <h2 id="admin_titre">Publier un article</h2><br>

            <form action="<?php echo _BASE; ?>/admin_news/publier_article/" method="POST" enctype="multipart/form-data">
                <label for="titre" class="nom_modif">Titre :</label>
                <input type="text" name="titre" id="news_titre" required><br><br>

                <label for="image" class="nom_modif">Insérer une image :</label>
                <input type="file" name="image" id="image"><br><br>

                <label for="article" class="nom_modif" id="text_area">Rédiger votre article :</label>
                <textarea id="texte_article" name="article"></textarea>

                <input type="hidden" name="id_utilisateur" value="<?= $_SESSION['id'] ?>">

                <input type="submit" name="sub" value="Publier">
            </form>
        </div>
    <?php } ?>

</body>

</html>
```

(Capture d'écran formulaire publication news)

```php
function publier_article($id) {
   require('model/admin_news.php');
// Vérifier si le formulaire a été soumis (le bouton "sub" est présent dans le formulaire)
    if (isset($_POST['sub'])) {
        // Récupérer les données du formulaire appliquer htmlspecialchars pour éviter les attaques XSS

        // Récupérer l'ID de l'utilisateur (éventuellement déjà fourni en paramètre de la fonction, mais ici on va le remplacer par cel
        $id = htmlspecialchars($_POST['id_utilisateur']);

        // Récupérer le titre de l'article
        $titre = htmlspecialchars($_POST['titre']);

        // Récupérer le contenu de l'article
        $article = htmlspecialchars($_POST['article']);

        $image = null;
         // Vérifier si une image a été téléchargée
        if (isset($_FILES['image']) && $_FILES['image']['error'] == 0) {
            // Dossier de destination où l'image sera sauvegardée
            $dossier = '../publique/img_article/';

            // Nom temporaire de l'image après son téléchargement
            $temp_name = $_FILES['image']['tmp_name'];

            // Vérifier que l'image a été téléchargée avec succès
            if (!is_uploaded_file($temp_name)) {
                exit('Erreur, l\'image est introuvable');
            }

            // Vérifier la taille de l'image pour éviter des fichiers trop volumineux
            if ($_FILES['image']['size'] >= 1000000) {
                exit('Erreur, l\'image est trop volumineuse');
            }

            // Obtenir l'extension de l'image téléchargée
            $infoimage = pathinfo($_FILES['image']['name']);
            $extensions_upload = $infoimage['extension'];

            // Convertir l'extension en minuscules pour la cohérence
            $extensions_upload = strtolower($extensions_upload);

            // Liste des extensions autorisées pour les images
            $extensions_autoriser = array('png', 'jpg', 'jpeg');

            // Vérifier si l'extension de l'image est autorisée
            if (!in_array($extensions_upload, $extensions_autoriser)) {
                exit('Erreur, veuillez insérer une image au bon format. Formats autorisés : png, jpg, jpeg');
            }

            // Construire le nom de l'image à partir de l'ID de l'utilisateur, du titre de l'article et de son extension
            $nom_image = $id . "_" . str_replace(' ', '_', $titre) . "." . $extensions_upload;

            // Déplacer l'image téléchargée vers le dossier de destination avec le nouveau nom
            if (!move_uploaded_file($temp_name, $dossier . $nom_image)) {
                exit('Problème dans le téléchargement de l\'image, veuillez réessayer');
            }

            // Attribuer le nom de l'image à la variable $image pour l'utiliser ultérieurement dans la fonction
            $image = $nom_image;
        } else {
            // Si aucune image n'a été téléchargée, utiliser une image par défaut
            $image = 'indies_logo.png';
        }

        // Appeler la fonction pour publier l'article avec les données récupérées
        publication_article($id, $titre, $article, $image);
```

```php
        // Afficher les données du formulaire (peut être utile pour déboguer)
        dbg($_POST);
    }
}
```

(Capture d'écran contrôleur publication d'article)

```php
function publication_article($id, $titre, $article , $image ) {
    require('connect.php');

    if(isset($_POST['sub'])) {
        // Récupérer les données du formulaire
        $id_utilisateur = $id;
        $titre = $titre ;
        $article = $article ;
        $image = $image;

        try {
            $pdo->beginTransaction();
            // Exécuter la requête SQL pour insérer l'article avec le chemin d'accès de l'image
            $sql = $pdo->prepare("INSERT INTO news (id_utilisateur, titre, article, image) VALUES (?, ?, ?, ?)");
            $sql->execute([$id_utilisateur, $titre, $article, $image]);

            $pdo->commit();
            // L'article a été inséré avec succès dans la base de données
            // Votre code supplémentaire ici...
        } catch(PDOException $e) {
            // Une erreur s'est produite lors de l'insertion de l'article
            echo "Erreur lors de l'insertion de l'article : " . $e->getMessage();

            throw $e;
        }
    }
```

(Capture d'écran model publication d'article)

Toujours dans la même logique de production, j'ai ensuite créé une fonction qui récupère les articles pour les afficher sur une page dynamique nommer news.

```php
function  affiche_news(){
    require('model/news.php');

    $datas = select_news();
    require('view/news.php');
}
```

(Capture d'écran contrôleur news)

```php
<?php
function select_news(){
    require('connect.php');
    try {
        $pdo->beginTransaction();

        $sql = $pdo->prepare('SELECT news.* FROM news');
        $sql->execute();  // Passer la valeur du paramètre $id_news à la méthode execute()

        $datas = $sql->fetchAll();  // Utiliser $sql au lieu de $pdo pour récupérer les données

        $pdo->commit();
        return $datas;

    }catch(PDOException $e){
        $pdo->rollback();
        throw $e;
    }
}
```

(Capture d'écran du model news)

```php
<?php
require('view/header.php');
?>
<div id="news">
        <h2 id="premier_titre">Découvrez les news chez Indies !</h2>

        <?php foreach ($datas as $data) { ?>

            <h3 id="titre_news"> <?= $data['titre'] ?> </h3>
            <div id="article"> <?= nl2br($data['article']) ?></div>
            <img id="image_news" src="/Dev/Indies/publique/img_article/<?= $data['image'] ?>" height="100" width="100">
            <p id="date"> <?= $data['news_date'] ?> </p>
<?php }
?>

 </div>

</body>

</html>
```

*(Capture d'écran view nwes***)**

Voici le style que j'ai appliqué par la suite en fin de projet sur la page new

```css
#news {
    border-radius: 20px;
    border: solid 4px rgb(0, 177, 255);
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    background-color: rgba(175, 175, 175, 0.924);
    display: flex;
    flex-direction: column;
    align-items: center;
    padding: 20px;
    margin: 0px 80px;
}

#premier_titre {
    font-family: 'SpidroMarley';
    font-size: 2rem;
}

#titre_news {
    font-family: 'SpidroMarley';
    font-size: 1.5rem;
    background-color: rgb(0, 177, 255);
    border-radius: 20px;
    padding: 8px;
    max-width: 900px;
    margin: 0 auto;
    color: aliceblue;
}

#article {
    font-family: 'SpidroMarley';
    font-size: 25px;
    text-align: center;
    padding: 20px 80px;
    margin: 0px;
}

#image_news {
    margin-top: 10px;
    height: 506px;
    width: 900px;

}

#date{
    padding-bottom: 60px;
}

.nav_modif{
    font-family: 'SpidroMarley';
    font-size: 1.5rem;
    color: rgb(205, 0, 0);
    text-decoration: none;
    padding-left: 20px;
}
```

```css
.nav_supp_news{
    font-family: 'SpidroMarley';
    font-size: 1.5rem;
    color: rgb(205, 0, 0);
    text-decoration: none;
    padding-left: 20px;
}

/* Media Queries */
@media screen and (max-width: 1024px) {
    #titre_news {
        max-width: 80%;
    }

    #image_news {
        height: auto;
        width: 80%;
    }
}

@media screen and (max-width: 768px) {
    #article {
        font-size: 20px;
        padding: 10px 20px;
    }
}

@media screen and (max-width: 480px) {
    #titre_news {
        font-size: 1.2rem;
        max-width: 90%;
    }

    #article {
        font-size: 18px;
        padding: 10px;
    }

    #image_news {
        height: auto;
        width: 90%;
    }
}

@media (max-width: 1280px) and (min-width: 1030px) {
    #image_news {
        height: auto;
        width: 70%;
    }

}
```

(Deux captures d'écran du code ccs appliquer à la page news)

Exemple 2

Suppression d’un utilisateur par un admin

Tout d’abord j’ai créé une page « gestion_utilisateur » qui affiche une liste d'utilisateurs avec leurs informations, récupérées à partir de la variable `$rows_users`.
Chaque utilisateur est affiché avec son nom d'utilisateur, ID utilisateur, adresse email et ID du rôle.

Pour chaque utilisateur, deux liens sont fournis : "Modifier cet utilisateur" et "Supprimer cet utilisateur".

Le lien de suppression a son ID utilisateur stocké dans l'attribut `data-id-user`, il appelle a l’aide d’un event.listenener mon controlleur admin_utiltilsateur et la fonction suppression_utilisateur

(La page charge également le fichier JavaScript `gestion_utilisateur.js` qui est le scipt en question.)

```php
<?php
require('header.php');

?>
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Liste-Utilisateurs-Indies</title>
</head>

<body>
    <div id="données_utilisateur">
    <h2 id="titre_2">Liste des utilisateurs</h2>

    <?php foreach ($rows_users as $row) { ?>
        <h3 class="info_utilisateur">Nom d'utilisateur : <span class="donnee_utilisateur"> <?= $row['nom'] ?> </span></h3>
        <p class="info_utilisateur">>Id utilisateur : <span class="donnee_utilisateur"> <?= $row['id_utilisateur'] ?> </span> </p>
        <p class="info_utilisateur">>Adresse Email : <span class="donnee_utilisateur"> <?= $row['email'] ?> </span> </p>
        <p class="info_utilisateur">>Id_role : <span class="donnee_utilisateur"> <?= $row['id_role'] ?> </span> </p>
        <!-- Lien de modification de l'utilisateur ciblé par son Id -->
        <a class="nav_modif" href="<?= _BASE ?>/admin_utilisateur/affiche_modifier_utilisateur/<?= $row['id_utilisateur'] ?>">Modifier cet utilisateur</a>
        <!-- Lien de suppression de l'utilisateur ciblé par son Id -->
        <a class="nav_supp" href="" data-id-user="<?= $row['id_utilisateur'] ?>">Supprimer cet utilisateur</a>
    <?php } ?>

    <script>var base_url = "<?php echo _BASE; ?>";</script>
    <script src="<?= _BASE ?>/../publique/script/gestion_utilisateur.js"></script>
    </div>
</body>

</html>
```

(Capture d’écran de la page gestion_utilisateur)

Contrôleur :

Cette page représente le contrôleur appelé lorsque l'utilisateur clique sur le lien "Supprimer cet utilisateur".

La fonction `suppression_utilisateur($id)` prend en paramètre l'ID de l'utilisateur à supprimer

La fonction appelle `supp_utilisateur($id)` qui ce trouve dans le model pour effectuer la suppression de l'utilisateur.

Si la suppression réussit, un message de succès est retourné dans un objet JSON.
Si la suppression échoue, un message d'échec est retourné dans un objet JSON.

```php
function suppression_utilisateur($id) {
    require('model/admin_utilisateur.php');

    $data = supp_utilisateur($id);

    if ($data) {
        $response = [
            'success' => true,
            'message' => 'Utilisateur supprimé'
        ];
    } else {
        $response = [
            'success' => false,
            'message' => 'La suppression de l\'utilisateur a échoué !'
        ];
    }


    echo json_encode($response);
    exit;
}
```

Model :

Cette page est le modèle (`admin_utilisateur.php`) qui est appelé par le contrôleur pour effectuer la suppression réelle de l'utilisateur dans la base de données.

Le fichier `connect.php` est requis pour établir une connexion à la base de données.

Une requête SQL est préparer et exécutée pour supprimer l'utilisateur ayant l'ID correspondant.

Si la suppression réussit, la transaction est validée (`commit`), et la fonction renvoie `true`.

Si la suppression échoue (par exemple, en raison d'une exception PDO), la transaction est annulée (`rollback`), et la fonction renvoie `false`.

```php
function supp_utilisateur($id) {
    require('connect.php');

    try {
        $pdo->beginTransaction();
        $sql = $pdo->prepare('DELETE FROM utilisateur WHERE utilisateur.id_utilisateur = ?');
        $sql->execute([$id]);
        $pdo->commit();
        return true; // Succès
    } catch (PDOException $e) {
        return false; // Échec
    }
}
```

En résumé, lorsque l'utilisateur clique sur le lien "Supprimer cet utilisateur" sur la première page, le contrôleur est appelé. Celui-ci à son tour fait appel au modèle pour effectuer la suppression de l'utilisateur dans la base de données. Si la suppression réussit, un message de succès est renvoyé au contrôleur, qui renvoie ensuite une réponse JSON à la première page. La première page peut ensuite traiter cette réponse pour afficher un message approprié à l'utilisateur.

Utilisation d’ajax

L'utilisation d'une requête AJAX pour la suppression d'utilisateur présente plusieurs avantages significatifs. Tout d'abord cela permet d'établir une interaction asynchrone entre l'utilisateur et le serveur, évitant ainsi le rechargement complet de la page lorsqu'une action de suppression est effectuée.

De plus contribue à une économie de bande passante en n'échangeant que les données essentielles entre le navigateur et le serveur.

Sur le plan de la sécurité, l'utilisation d'AJAX pour les opérations critiques comme la suppression d'utilisateur peut renforcer la protection des données.

Exemple 3

Bien sûr, voici un résumé plus court en français à la première personne du singulier :

"Dans mon application web basée sur l'architecture MVC, le fichier 'header.php' joue un rôle clé en affichant de manière cohérente l'en-tête sur toutes les pages. Cela assure une apparence uniforme avec le logo, les liens de navigation et les styles CSS communs. Grâce à cette approche, je peux facilement maintenir et mettre à jour l'en-tête en effectuant des modifications dans un seul endroit. De plus, cette centralisation du code rend mon application plus évolutive et facilite la réutilisation du code. En incluant les balises méta et autres éléments SEO dans 'header.php', j'améliore également le référencement de mon site. En résumé, 'header.php' offre une cohérence visuelle, une maintenance simplifiée et une amélioration du référencement pour une meilleure expérience utilisateur."

```php
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Les Indies </title>
    <link rel="stylesheet" href="<?php echo _BASE; ?>/view/styles/header.css">
    <link rel="stylesheet" href="<?php echo _BASE; ?>/view/styles/accueil.css">
    <link rel="stylesheet" href="<?php echo _BASE; ?>/view/styles/news.css">
    <link rel="stylesheet" href="<?php echo _BASE; ?>/view/styles/admin.css">
    <link rel="stylesheet" href="<?php echo _BASE; ?>/view/styles/publication_news.css">
    <link rel="stylesheet" href="<?php echo _BASE; ?>/view/styles/modification_news.css">
    <link rel="stylesheet" href="<?php echo _BASE; ?>/view/styles/utilisateur.css">
    <link rel="stylesheet" href="<?php echo _BASE; ?>/view/styles/modif_utilisateur.css">
    <link rel="stylesheet" href="<?php echo _BASE; ?>/view/styles/liste_utilisateur.css">
    <link rel="stylesheet" href="<?php echo _BASE; ?>/view/styles/connexion.css">
    <link rel="stylesheet" href="<?php echo _BASE; ?>/view/styles/inscription.css">

</head>

<body>
    <header>
        <?php if (session_status() !== PHP_SESSION_ACTIVE) session_start();

        if (!empty($_SESSION['id'])) { ?>
            <div id="deco_gestion">
                <!-- Lien de Deconnexion -->
                <div id="nav_deconnexion">
                    <a href="<?php echo _BASE; ?>/inscription_connexion/se_deconnecter">Déconnexion</a>
                </div>
                <!-- Lien de gestion du compte de l'utilisateur connecté -->
                <div id="nav_gestion_du_compte">
                    <a href="<?php echo _BASE; ?>/utilisateur/gestion_compte/<?php echo $_SESSION['id'] ?>">Gérer mon compte utilisateur</a>
                </div>
            </div>
            <?php
            if ($_SESSION['role'] == 1) {
            ?>
                <!-- Lien d'administration -->
                <div id="nav_admin">
                    <a href="<?php echo _BASE; ?>/admin/affiche_administration">Administration</a>
                </div>
        <?php
            }
        }
        ?>
```

(Capture du header partie 1)

(Capture du header partie 2)

Conclusion

Le projet

Le site web n'étant pas encore terminé en raison du manque de temps, il me reste encore beaucoup de travail à accomplir. Cependant, je tiens à souligner que je suis déterminé à poursuivre mes efforts sur ce projet et à le mener à terme. Ce projet revêt une dimension personnelle pour moi, mais il est également destiné à servir ma communauté, qui était consciente dès le départ que sa réalisation nécessiterait plus de temps.

Voici ma liste d'objectifs pour les prochaines étapes :

1. Finaliser toutes les fonctionnalités prévues dans le cahier des charges.

2. Améliorer le référencement en mettant en place les balises méta et en optimisant les autres éléments pour le SEO.

3. Mettre en place la gestion des cookies, notamment en intégrant Google Analytics.

4. Assurer des mises à jour régulières pour les systèmes utilisés.

5. Renforcer la sécurité côté client, en particulier en confirmant la validité des données personnelles.

6. Installer un certificat SSL pour garantir le protocole HTTPS.

7. Mettre en place une protection contre les attaques CSRF (Cross-Site Request Forgery).

8. Effectuer des recherches et mettre en place une protection contre les attaques Clickjacking.

En accomplissant ces objectifs, je m'efforcerai de créer un site web fiable, sécurisé et bénéfique à la fois pour moi-même et pour la communauté qui en profitera.

Mon ressentis sur cette expérience et mes futurs objectifs

Malgré la difficulté et le caractère périlleux de l'exercice, cette expérience a renforcé mon attrait pour le développement web. Travailler sur ce projet m'a permis d'acquérir de nombreuses connaissances, de mettre en application celles apprise et je suis résolu à continuer à enrichir mes compétences dans ce domaine.

Le facteur temps et la nature du travail à distance, en dehors d'un cadre professionnel, se sont avérés plus difficiles à gérer que je ne l'avais anticipé.

Néanmoins, je suis satisfait du travail que j'ai accompli. En obtenant cette certification DWWM, j'ai l'intention de poursuivre ma quête de compétences en optant pour un apprentissage, afin de continuer à me former tout en plongeant dans le monde professionnel du développement web.