Projet transversale — Solution web
“Projet de groupe : Préparation et présentation d’un projet solution web”

Participants :

Antoine BERTHE
Tom CLEMENT
William COLLE
Clément SALINGUE
Groupe numéro 2

Projet encadré par :

Gregory BOUDRINGHIN

Préparation et présentation d’un projet solution web - Première Année, 2025

EPSI — Ecole de l’ingénierie informatique, Arras

1/ Compréhension du sujet :

Contexte :
Pour ce projet, on doit réaliser un service web de livrables qui n'accepte que l'envoi de fichiers ZIP et de fichiers powerpoint. Le site doit s'adapter à tous les écrans.

Problématique :
Comment concevoir un service web sécurisé et ergonomique permettant la gestion et le dépôt de livrables pour un usage collaboratif tout en respectant les contraintes techniques spécifiques ?

Sous - Problématique :
. Gestion des livrables côté administration :
Comment permettre la création, modification, suppression et visualisation des dossiers et fichiers tout en assurant une interface utilisateur simple et intuitive ?

. Soumission des livrables côté utilisateur :
Comment garantir une soumission de fichiers restreinte aux formats ZIP et PowerPoint dans un système d'arborescence des dossiers ?

. Sécurité et performance du service web :
Comment sécuriser les données échangées et organiser le développement pour que le site reste performant ?

Objectif :
Rendre le site fonctionnel avec les couleurs de l'EPSI

2/ Brainstorming :

Trouver les idées :

Côté administration :
. Création d'un formulaire pour gérer les dossiers.
. Intégration d'une arborescence dynamique affichant dossiers et fichiers (utilisation de PHP).
. Mise en place d'une confirmation double avant suppression + second popup pour être sûr de le supprimer).
. Gestion de la date limite avec un calendrier (input type="date").
. Permettre à l'admin de choisir l'extension attendue par l'utilisateur
. Envoyer un mail 5 jours avant si l'utilisateur n'a toujours rien envoyer pour un rappel
. L'admin peut accepter/refuser/recommencer un fichier envoyer par l'utilisateur

Côté utilisateur :
. Affichage de l'arborescence des dossiers envoi de fichiers limité aux formats ZIP PowerPoint avec vérification côté client et serveur (PHP).
. Affichage des notifications de succès ou d'erreur pour chaque action (exemple : fichier mal formaté, dépôt réussi).

Performance :
. Vérification des extensions de fichiers à plusieurs niveaux (client et serveur).
. Protection contre les injections (utilisation de PDO pour les interactions avec une base de données).
. Les fichiers vont être stockés physiquement.

Analyse de l'existant + sources :

Ressources techniques :

Formulaires HTML5 :

`<form>`, `<input type="file">`, `<input type="date">`.

Gestion des fichiers avec PHP :

`opendir()`, `readdir()`, `unlink()` pour lire et gérer les fichiers.

CRUD :

Utilisation d'une base de données MySQL pour gérer les informations sur les dossiers et fichiers (nom, section, date limite, etc.).

Sources :

https://www.php.net/manual/fr/index.php

https://developer.mozilla.org/fr/docs/Learn_web_development/Core/Structuring_content

https://www.letecode.com/php-mysql-pour-debutant-application-crud-inserer-lire-modifier-et-supprimer

https://www.jstree.com/docs/html/

3/ Choix des techniques fonctionnelles :

Côté administration (admin.php)

. Qui : Destiné aux administrateurs ayant des droits spécifiques.

. Quoi : CRUD sur les dossiers (création, modification, suppression, visualisation).

. Où : Accessible via une page sécurisée (formulaire de type html).

. Quand : À tout moment, avec restrictions d'accès (authentification).

. Comment :

. Utilisation de PHP pour gérer les dossiers et fichiers sur le serveur.

. Base de données MySQL pour stocker les informations de gestion.

. Intégration d'un système d'arborescence.

. Pourquoi : Permettre une gestion centralisée des livrables.

Côté utilisateur (index.php)

. Qui : Utilisateurs souhaitant soumettre des livrables.

. Quoi : Envoi de fichiers (ZIP et PPT uniquement) dans des dossiers prédéfinis.

. Où : Via une page web responsive.

. Quand : Dans les limites des dates définies par l'administrateur.

. Comment :

. Interface utilisateur responsive (HTML5, CSS3, .

. Vérification des fichiers côté client et serveur.

. Arborescence des dossiers.

. Pourquoi : Faciliter le dépôt des livrables en respectant les formats autorisés.

4/ Découpage de tâches :

Tâche 1 :
Conception de la base de données
Création de la table depots_fichiers :
Colonnes : id, utilisateur_id, nom_fichier, date_envoi.
Création de la table devoirs_rendus :
Colonnes : id, utilisateur_id, depot_id, date_rendu.
Création de la table utilisateurs

Tâche 2 :
Développement côté administration
Page admin.php :

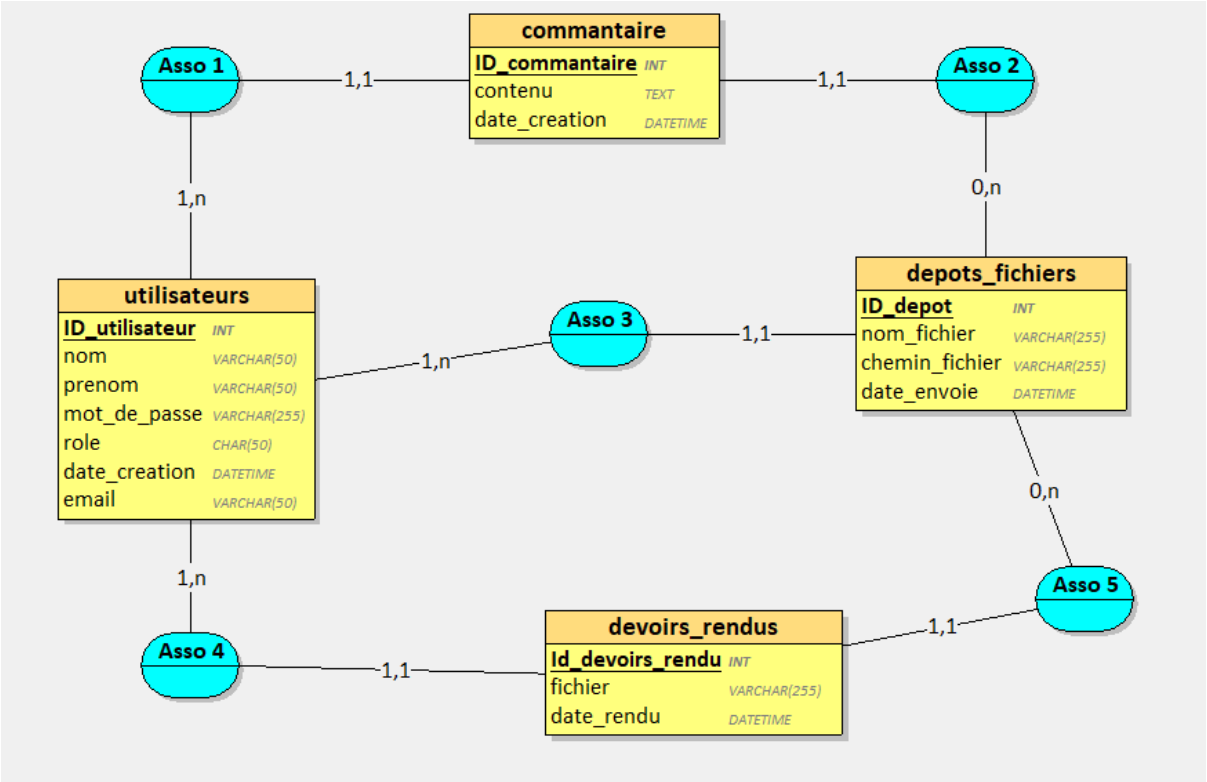
Formulaire de création de dossier avec champ section et date limite.
Arborescence (lecture des dossiers et fichiers depuis le serveur).
CRUD sur les dossiers (formulaire de modification et boutons de suppression).
Ajout des messages de confirmation avant suppression.

Tâche 3 :
Développement côté utilisateur
Page index.php :
Affichage de l'arborescence des dossiers
Formulaire d'envoi de fichiers limité aux ZIP et PPT.
Messages de confirmation (succès/erreur).

Tâche 4 : Responsiveness et design
Intégration des couleurs et logos EPSI/WIS.
Utilisation de CSS3 pour adaptabilité.

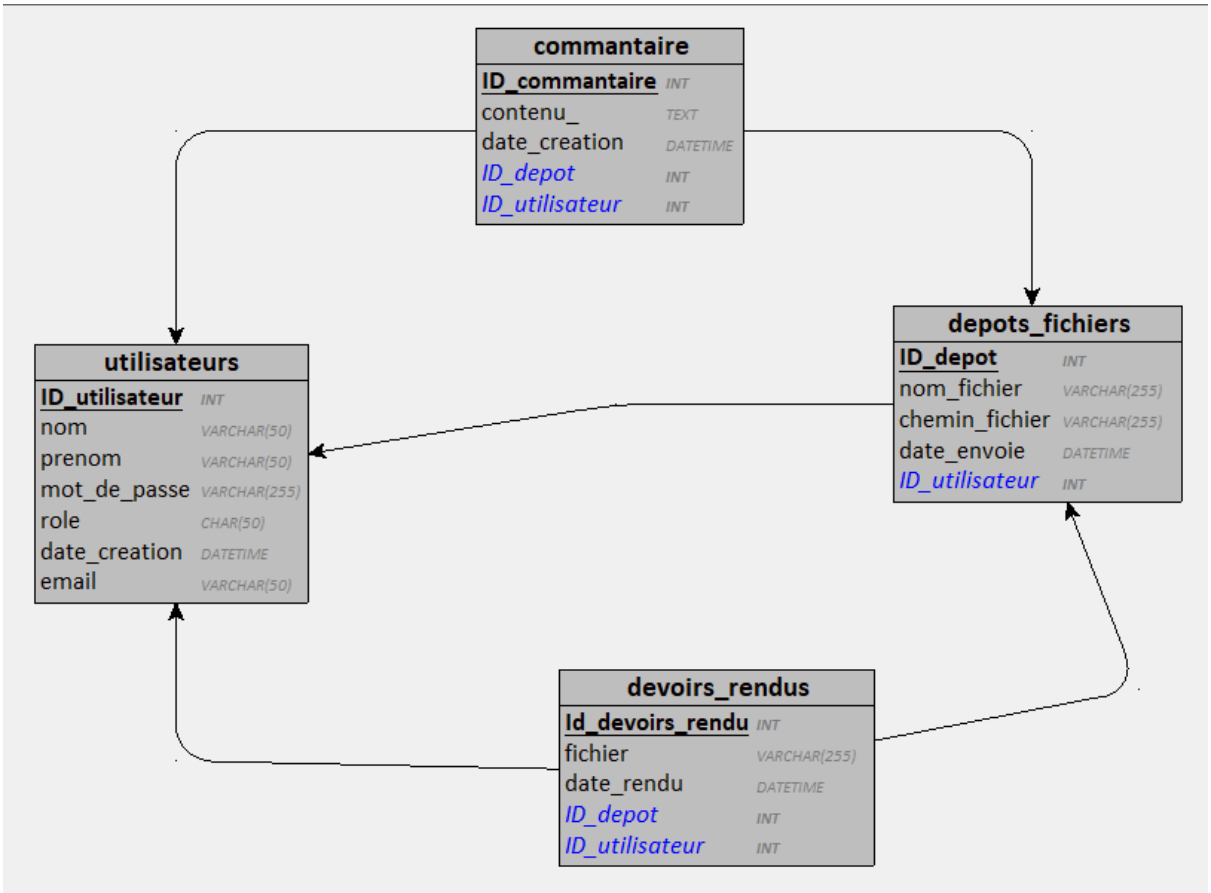
Tâche 5 : Tests et déploiement
Tests unitaires : Validation des fichiers, vérification CRUD.
Tests d'intégration : Interaction entre admin.php et index.php.
Déploiement final.

5/ MCD MLD :



Dans ce MCD on constate que nous retrouvons que des relations père fils :

- l'utilisateur peut mettre 1 ou plus commentaire or le commentaire appartient à 1 et 1 seul utilisateur
- un commentaire appartient à 1 seul dépôts de fichier or le dépôt de fichier peut recevoir aucun ou plusieur commentaire
- le dépôt de fichier et en lien avec 1 et 1 seul utilisateur or l'utilisateur peut ce rendre sur plusieur dépôt de fichier
- l'utilisateur rend 1 ou plusieurs devoirs mais les devoirs appartiennent à 1 et 1 seul utilisateur
- les devoirs sont rendu sur 1 seul dépôt néanmoins les dépôts peuvent recevoirs plusieurs devoirs



6/ DICTIONNAIRE DE DONNEES

Nom de l'Attribut	Type de Donnée	Description	Contraintes
utilisateur			
id_utilisateur	INT	identifiant users	clé primaire
nom	Varchar(50)	nom users	
prenom	Varchar(50)	prenom users	not null
mail	Varchar(50)	mail users	not null
mot_de_passe	Varchar(255)	mdp users	not null
role	Char(50)	role users	
date_creation	DateTime	date de création du users	
commentaire			

id_commentaire	INT	identifiant commentaire	clé primaire
contenu	TEXT	contenu du commentaire	Not null
date_creation	DateTime	date de création du commentaire	
depot_fichier			
id_depot	INT	identifiant depot	clé primaire
nom_fichier	Varchar(255)	nom du fichier	not null
chemin_fichier	Varchar(255)	chemin du fichier	not null
date_envoie	DateTime	date envoie du fichier	not null
devoirs_rendus			
id_devoirs_rendus	INT	identifiant du devoir	clé primaire
fichier	Varchar(255)	fichier du devoir	not null
date_rendu	DateTime	date de rendu du devoir	not null

7/ DÉPENDANCE FONCTIONNELLE

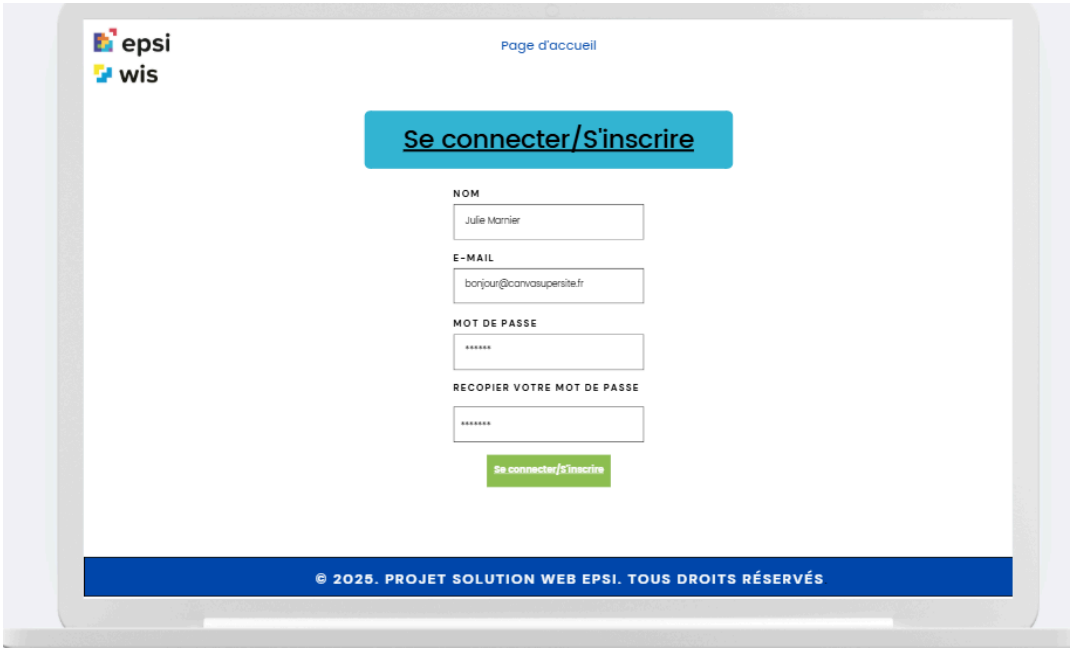
utilisateur
Id_utilisateur -> (nom, prenom, email, mot_de_passe, role, date_creation)

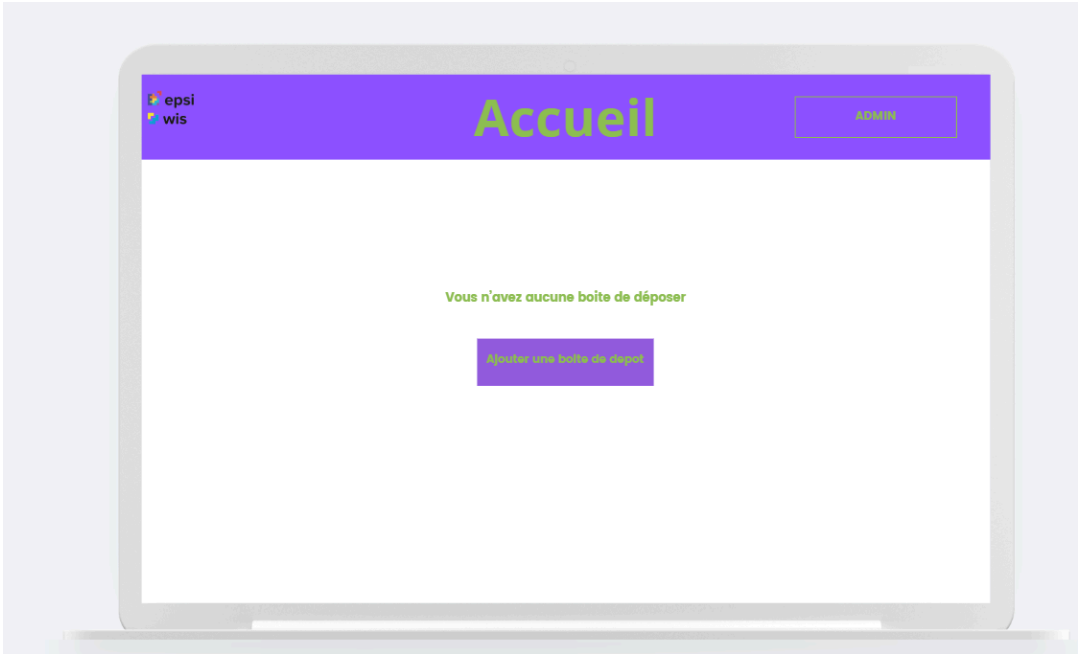
Commentaire
Id_Commentaire -> (contenu, date_creation, #id_utilisateur, #Id_depot)

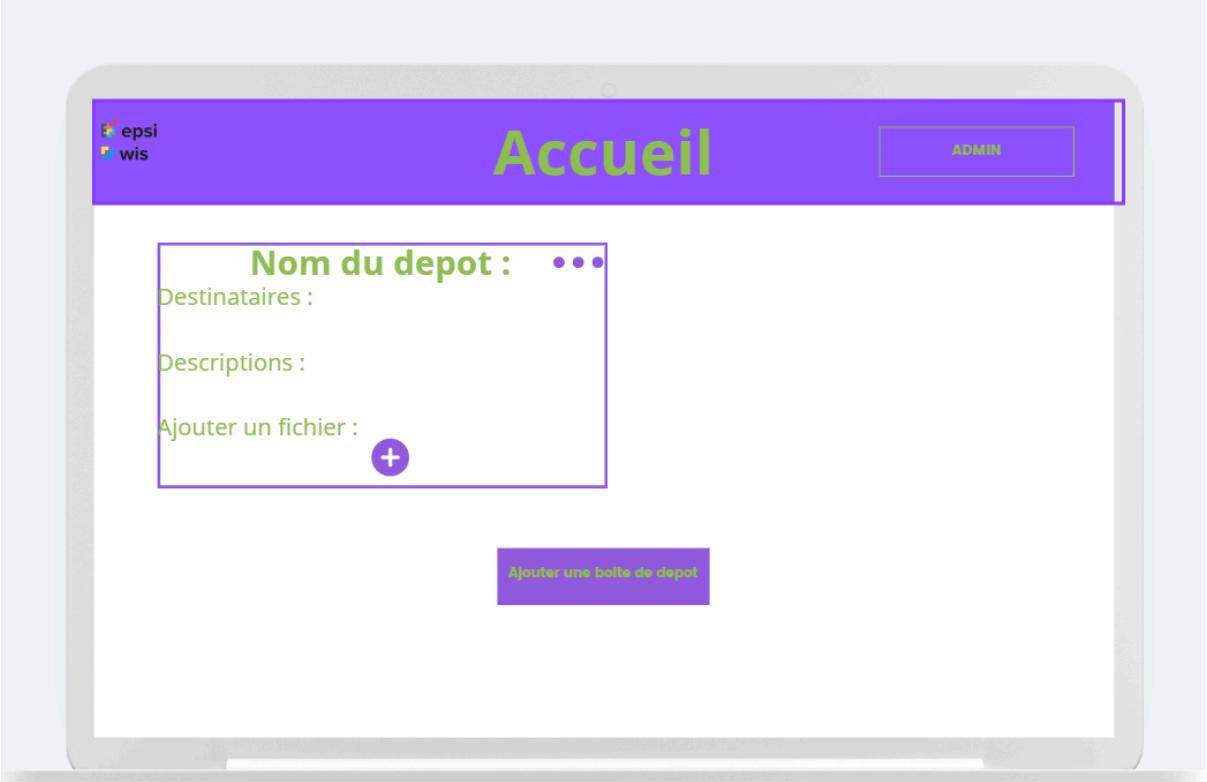
depot_fichier
Id_depot -> (nom_fichier, chemin_fichier, date_envoie, #id_utilisateur)

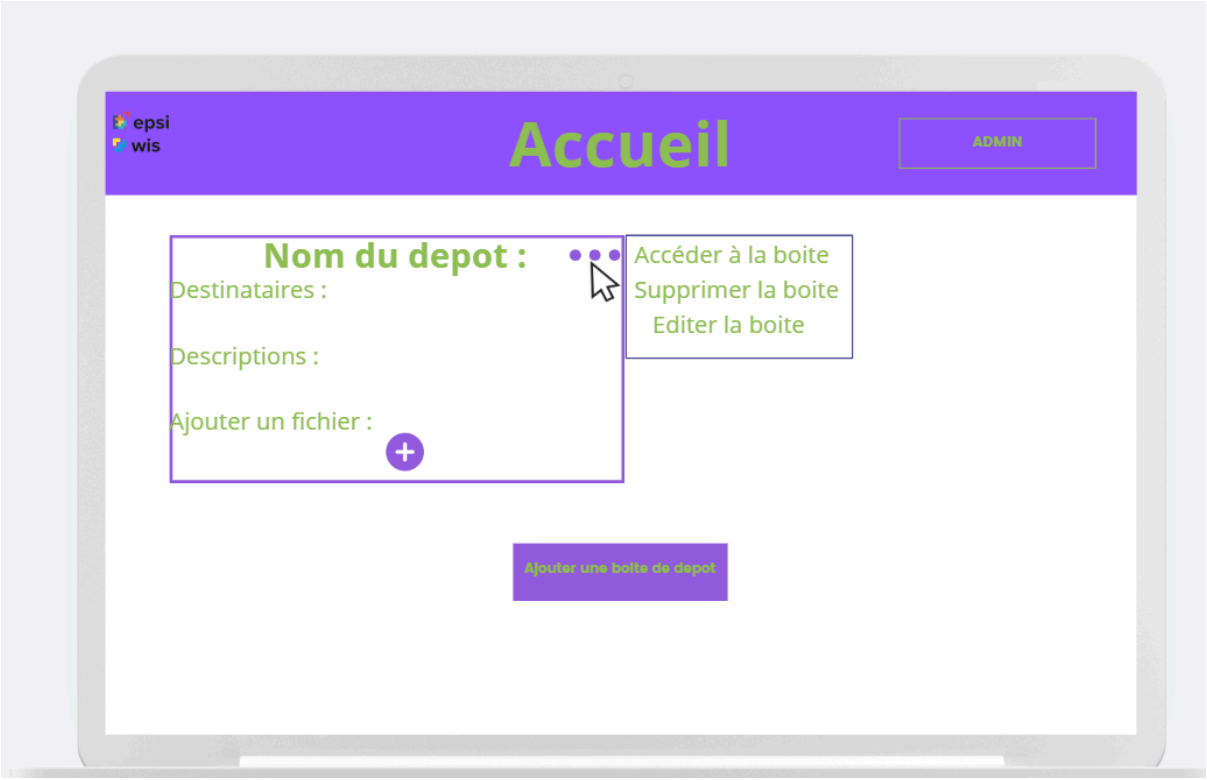
devoirs_rendus
Id_devoirs_rendus -> (fichier, date_rendu, #id_depot, #id_utilisateur)

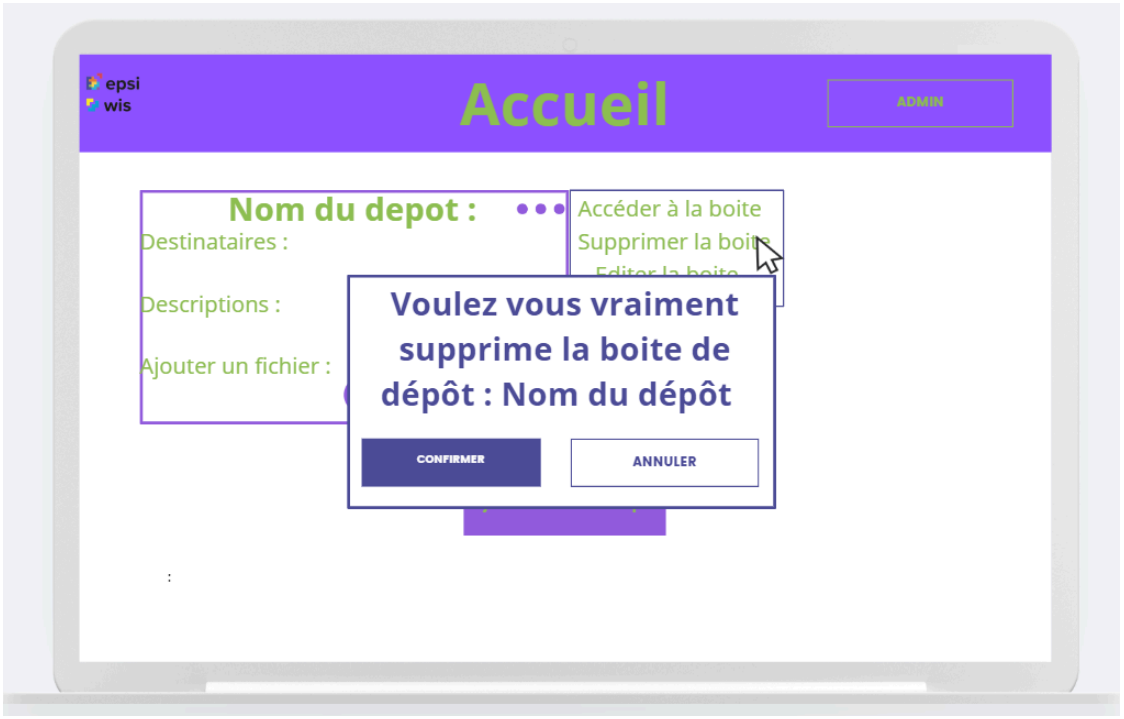
8/ Maquette


epsi
wis
Page d'accueil
Se connecter/S'inscrire
NOM
Julie Marnier
E-MAIL
bonjour@canvasupersite.fr
MOT DE PASSE
RECOPIER VOTRE MOT DE PASSE
Se connecter/S'inscrire
© 2025. PROJET SOLUTION WEB EPSI. TOUS DROITS RÉSERVÉS

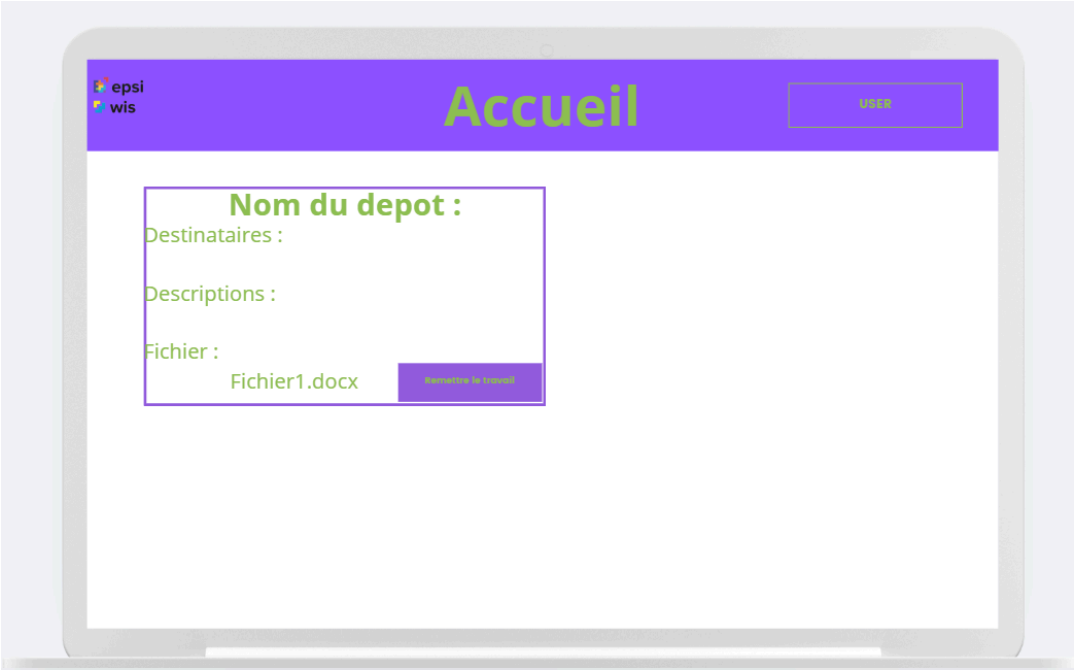

epsi
wis
Accueil
ADMIN
Vous n'avez aucune boite de déposer
Ajouter une boite de dépot


epsi
wis
Accueil
ADMIN
Nom du depot :
Destinataires :
Descriptions :
Ajouter un fichier :
Ajouter une boite de depot


epsi
wis
Accueil
ADMIN
Nom du depot :
Destinataires :
Descriptions :
Ajouter un fichier :
Accéder à la boite
Supprimer la boite
Editer la boite
Ajouter une boite de depot


epsi
wis
Accueil
ADMIN
Nom du depot :
Destinataires :
Descriptions :
Ajouter un fichier :
Accéder à la boite
Supprimer la boite
Voulez vous vraiment supprime la boite de dépôt : Nom du dépôt
CONFIRMER
ANNULER
:


epsi
wis
Accueil
ADMIN
Nom du depot :
Destinataires :
Descriptions :
Ajouter un fichier :
Ajouter une boite de depot



9/ Explication des techniques utilisé et du code :

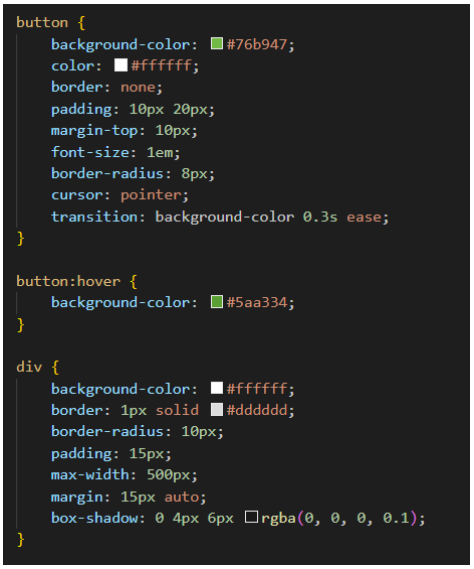
Pour ce projet nous avons donc utilisé du HTML (HyperText Markup Language) qui permet de structurer le contenu d'une page : titres, paragraphes, images, liens, formulaires, etc.
du CSS (Cascading Style Sheets) qui permet de rendre le site joli en modifiant les couleurs, la taille du texte, la mise en page, les animations, etc.
et du PHP (Hypertext Preprocessor) qui est un langage côté serveur qui permet de rendre le site dynamique : gérer des formulaires, se connecter à une base de données, afficher des données différentes selon les utilisateurs, etc.

Partie explicative sur des fonctionnalité du CSS

```css
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
    color: #333333;
}

header {
    background: linear-gradient(90deg, #004b87, #76b947);
    color: #ffffff;
    display: flex;
    align-items: center;
    padding: 15px 30px;
    justify-content: space-between;
    flex-wrap: wrap;
}

header img.logoepsi {
    margin-right: 20px;
}

header h1.green {
    font-size: 2em;
    flex-grow: 1;
    margin: 0;
}

.pagemenu {
    font-size: 1.1em;
    font-weight: bold;
    color: #ffffff;
}

.right {
    text-align: right;
    margin: 10px 30px;
    font-weight: bold;
}
```

Pour le body , on a changé la police sur toute la page avec un fond (background-color)gris clair, nous avons enlevé les marges et le padding par défaut du navigateur et ajouté une couleur du texte qui est gris foncé .

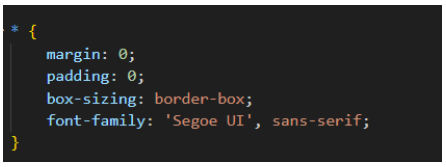
pour le header : ce qui change c'est le display : flex ca permet de bien placer les éléments.
le justify-content: space-between : permet d'avoir un espace égal entre les blocs (logo a gauche , titre au centre et menu à droite)
flex-wrap:wrap: permet aux éléments d'aller à la ligne quand on rétrécie l'ecran.



```css
button {
    background-color: #76b947;
    color: #ffffff;
    border: none;
    padding: 10px 20px;
    margin-top: 10px;
    font-size: 1em;
    border-radius: 8px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #5aa334;
}

div {
    background-color: #ffffff;
    border: 1px solid #dddddd;
    border-radius: 10px;
    padding: 15px;
    max-width: 500px;
    margin: 15px auto;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}
```

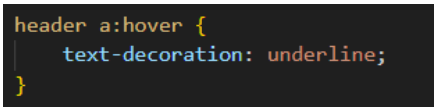
Pour les boutons, on a le padding qui permets d'arrondir les bords.
le cursor : pointeur permet de changer le pointeur de la souris quand on passe sur le bouton.
button:hover: permet d'avoir un changement de couleur quand on survole le bouton

pour les div la box-shadow: permet d'avoir un petit ombrage



```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Segoe UI', sans-serif;
}
```

le * est un sélecteur universel qui permet d'appliquer le css à tous les éléments de la page HTML.



```css
header a:hover {
    text-decoration: underline;
}
```

Le text-decoration: underline sert à lorsqu'on passe la souris dessus ça devient souligné.

Partie explicative sur des fonctionnalité du HTML

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Création d'un nouveau dépôt </title>
    <link href="../css/createDepot.css" rel="stylesheet"/>
    <link rel="icon" type="image/png" href="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTS5VYolg4PlHUkQ7wMn4lTENI-rS9XfFDTOg&s">
</head>
<body>

<header>
    <img src="https://www.nuitdelinfo.com/inscription/uploads/ecoles/573/logos/logo.png" alt="Logo EPSI/WIS" class="logoepsi" width="85" height="80">
    <h1 class="green"> Création d'un nouveau dépôt : </h1>
    <p class="center">
        <p class="pagemenu"><a href="../Admin/indexAdmin.php"> Accueil </a></p>
    </p>
</header>
```

Les balises titles permettent d'afficher le texte dans l'onglet du navigateur

- <link href="../css/createDepot.css" rel="stylesheet"/> c'est le lien en relation avec le fichier css pour avoir le style de la page
- <link rel="icon" type="image/png" href="https://..."> permet d'afficher le logo dans l'onglet du navigateur

La balise body c'est tout ce que voit l'utilisateur
Le header qui est l'en-tête de page comporte les images avec une différentes taille, le titre principal et un lien accueil qui retourne qui retourne vers indexAdmin

```html
<form method="POST" action="createDepot.php" enctype="multipart/form-data">
    <label for="nom"> Titre du devoir : </label><br>
    <input type="text" id="nom" name="nom" required><br><br>

    <label for="destinataire"> Destinataire : </label><br>
    <select id="destinataire" name="destinataire" required>
        <option value="" disabled selected>Choisir le destinataire</option>
        <option value="sn1">SN1 - Arras</option>
        <option value="sn2">SN2 - Arras</option>
    </select><br><br>

    <label for="matiere"> Matière : </label><br>
    <select id="matiere" name="matiere" required>
        <option value="" disabled selected>Choisir la matière</option>
        <option value="Algo et PHP">Algo et PHP</option>
        <option value="Anglais">Anglais</option>
        <option value="ATL Git">ATL Git</option>
        <option value="ATL Google Outils">ATL Google Outils</option>
        <option value="SISR">SISR</option>
        <option value="SLAM">SLAM</option>
        <option value="HEP">HEP</option>
        <option value="Mathématiques">Mathématiques</option>
        <option value="CEJM">CEJM</option>
        <option value="Projet Transversal">Projet transversal</option>
        <option value="PHP et MySQL">PHP et MySQL</option>
        <option value="Python">Python</option>
    </select><br><br>

    <label for="description"> Description : </label><br>
    <input type="text" id="description" name="description" required><br><br>

    <label for="date"> A rendre pour le : </label><br>
    <input type="date" id="date" name="date" required><br><br>

    <label for="monfichier"> Joindre un fichier : </label><br>
    <input type="file" id="monfichier" name="monfichier" accept=".zip,.pptx" required><br><br>

    <input type="submit" class="submitcreatedepot" value="Créer le Dépôt">
</form><br>

<footer>
    <p>&copy; 2025. Projet Solution Web EPSI. Tous droits réservés.</p>
</footer>

</body>
</html>
```

<form method="POST" action="createDepot.php" enctype="multipart/form-data"> : c'est un formulaire qui permet à l'utilisateur de saisir les infos du devoirs à déposer.

ensuite on a les champs du formulaire
avec un menu déroulant pour choisir la classe concernée
ensuite on peut choisir la matière
puis mettre une description
une date limite
un fichier à uploader qui peut être seulement en .zip ou .pptx
et pour finir un bouton de soumission du formulaire

```php
<header>
    <h1>Devoirs des étudiants</h1>
    <a href="../Admin/indexAdmin.php">← Retour à l'accueil</a>
</header>

<main>
    <?php if (empty($rendus)): ?>
        <p>Aucun devoir n'a été rendu pour ce dépôt.</p>
    <?php else: ?>
        <table>
            <thead>
                <tr>
                    <th>Étudiant</th>
                    <th>Fichier</th>
                    <th>Date de rendu</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <?php foreach ($rendus as $rendu): ?>
                    <tr>
                        <td><?= htmlspecialchars($rendu['prenom'] . ' ' . $rendu['nom']) ?></td>
                        <td><?= basename($rendu['fichier']) ?></td>
                        <td><?= $rendu['date_rendu'] ?></td>
                        <td>
                            <form method="post" action="download.php" style="margin:0;">
                                <input type="hidden" name="fichier" value="<?= htmlspecialchars($rendu['fichier']) ?>">
                                <button type="submit">Télécharger</button>
                            </form>
                        </td>
                    </tr>
                <?php endforeach; ?>
            </tbody>
        </table>
    <?php endif; ?>
</main>

</body>
</html>
```

le main correspond au contenu principal on retrouve dedans <main>
<?php if (empty($rendus)): ?>
Aucun devoir n'a été rendu pour ce dépôt. ca vérifie si la variable $rendus est vide et si c'est vide on a un message si c'est pas vide ca contient les devoirs .

Ca affiche un tableau contenant les colonnes Etudiant , Fichier , Date de rendu et le bouton pour télécharger.

Partie explicative sur des fonctionnalité du PHP

```php
<?php foreach ($rendus as $rendu): ?>
    <tr>
        <td><?= htmlspecialchars($rendu['prenom'] . ' ' . $rendu['nom']) ?></td>
        <td><?= basename($rendu['fichier']) ?></td>
        <td><?= $rendu['date_rendu'] ?></td>
        <td>
            <form method="post" action="download.php" style="margin:0;">
                <input type="hidden" name="fichier" value="<?= htmlspecialchars($rendu['fichier']) ?>">
                <button type="submit">Télécharger</button>
            </form>
        </td>
    </tr>
<?php endforeach; ?>
```

Ce code PHP permet l'affichage de chaque ligne de devoir, c'est une boucle qui parcourt chaque devoir rendu et affiche le nom complet de l'étudiant, le nom du fichier (basename permet d'afficher juste le nom pas le chemin complet)
la date de rendu et le bouton pour télécharger le fichier)

```php
<?php

session_start();
if (!isset($_SESSION["user_id"])) {
    header("Location: ../account/login.php");
    exit;
}
if ($_SESSION["role"] !== "admin") {
    header("Location: ../User/indexUser.php");
    exit;
}

require '../config/database.php';
```

permet de démarrer la session pour accéder aux données de l'utilisateur connecté
si aucun utilisateurs n'est connecté ca renvoie vers la page de login

et si l'utilisateur est connecté mais pas admin il est redirigé vers sa page utilisateur
et on a la connexion avec la base de donnée

```php
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $nom = trim($_POST['nom'] ?? '');
    $destinataire = $_POST['destinataire'] ?? '';
    $matiere = trim($_POST['matiere'] ?? '');
    $description = $_POST['description'] ?? '';
    $date = $_POST['date'] ?? '';
    $file = $_FILES['monfichier'] ?? null;
    $user_id = $_SESSION['user_id'];


    $subDir = '../uploads/' . $matiere . '/' . $nom . '/';
    if (!is_dir($subDir)) {
        mkdir($subDir, 0777, true);
    }

    $uploadFile = $subDir . basename($file['name']);


    if (move_uploaded_file($file['tmp_name'], $uploadFile)) {
        echo "<p style='color:green;'>Le dépôt a été créé avec succès !</p>";


        $sql = "INSERT INTO depots_fichiers (utilisateur_id, nom_fichier, chemin_fichier, date_envoi)
                VALUES (:uid, :nom, :chemin, NOW())";
        $stmt = $conn->prepare($sql);
        $stmt->execute([
            ':uid' => $user_id,
            ':nom' => $nom,
            ':chemin' => $uploadFile
        ]);
```

On commence par savoir si le formulaire a été soumis en méthode POST
On récupère ensuite toutes les données du formulaire .
On crée ensuite un fichier Dans uploads/[matière]/[nom_du_devoir]/ et si le fichier n'existe pas on le crée avec les droits 0777.

$uploadFile = $subDir . basename($file['name']); on crée le chemin du ficher et on le déplace depuis un emplacement temporaire vers un bon fichier .

ensuite on insère l'id de l'utilisateur , le nom du devoir , le chemin du fichier, la date et l'heure dans la base de donnée.

```php
$utilisateur_id = $_SESSION['user_id'];
$depot_id = $_GET['depot_id'] ?? null;


$nom = $matiere = $description = $date = "";

if ($depot_id) {
    $stmt = $conn->prepare("SELECT * FROM depots_fichiers WHERE id = :id");
    $stmt->execute([':id' => $depot_id]);
    $depot = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($depot) {
        $nom = $depot['nom_fichier'];
        $description = $depot['description'] ?? '';
        $date = $depot['date_limite'] ?? '';
        $matiere = "";
    } else {
        echo "<p>Dépot introuvable.</p>";
        exit;
    }
}
```

Pour rendre un devoir , au début on récupère l'identifiant de l'utilisateur connecté et la récupération de l'id dépôt via le $_Get par l'URL .

Si l'id du dépôt est présent on recupere les infos via la BDD et on récupère donc les informations nom_fichier ect .

sinon le dossier est introuvable.

```php
if ($_SERVER["REQUEST_METHOD"] === "POST" && isset($_FILES['mondevoir'])) {
    $file = $_FILES['mondevoir'];
    $filename = basename($file['name']);
    $uploadDir = '../uploads/rendus/';

    if (!is_dir($uploadDir)) {
        mkdir($uploadDir, 0777, true);
    }

    $target = $uploadDir . $filename;

    if (move_uploaded_file($file['tmp_name'], $target)) {
        $stmt = $conn->prepare("INSERT INTO devoirs_rendus (utilisateur_id, depot_id, fichier) VALUES (:uid, :depot_id, :fichier)");
        $stmt->execute([
            ':uid' => $utilisateur_id,
            ':depot_id' => $depot_id,
            ':fichier' => $target
        ]);
        echo "<p style='color:green;'>Fichier envoyé avec succès !</p>";
    } else {
        echo "<p style='color:red;'>Erreur lors de l'envoi du fichier.</p>";
    }
}
?>

<!DOCTYPE html>
```

si le formulaire a été soumis en post et que le champ "mondevoir" contient un fichier, on récupere le fichier envoyé avec le nom du fichier et ou il va être stocké avec $uploadir.

On détermine le chemin complet du fichier et on déplace le fichier temporaire vers le dossier final
et si ca reussit ca enregiste les inforamtions dans la table devoir_rendus.

```php
<?php
session_start();
require '../config/database.php';

if (!isset($_SESSION["user_id"]) || $_SESSION["role"] !== "admin") {
    header("Location: ../account/login.php");
    exit;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['depot_id'])) {
    $depot_id = intval($_POST['depot_id']);
```

Permet de supprimer un dépôt dans la base de donnée et sur le serveur que côté admin.

Si un dépôt est présent et qu'elle est appelé en POST on le récupère, on prépare une requête pour récupérer le dépôt correspondant à l'id dont $depot contient toutes les infos. On récupère le chemin du fichier

```php
    if (file_exists($chemin)) {
        if (is_file($chemin)) {
            unlink($chemin);
        } elseif (is_dir($chemin)) {
            foreach (new RecursiveIteratorIterator(
                new RecursiveDirectoryIterator($chemin, RecursiveDirectoryIterator::SKIP_DOTS),
                RecursiveIteratorIterator::CHILD_FIRST
            ) as $element) {
                $element->isFile() ? unlink($element->getPathname()) : rmdir($element->getPathname());
            }
            rmdir($chemin);
        }
    }

    $delete = $conn->prepare("DELETE FROM depots_fichiers WHERE id = :id");
    $delete->execute([':id' => $depot_id]);

    echo "<p>Dépôt supprimé avec succès (fichier + base de données).</p>";
} else {
    echo "<p>Aucun dépôt spécifié.</p>";
}
```

si les dossiers existe parcourt récursivement le dossier et supprime tout ce qu'il contient c'est à dire les dossiers en profondeurs , les fichiers avec (unlink() et les dossier vide avec (rmdir)

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $nom = $_POST["nom"];
    $prenom = $_POST["prenom"];
    $email = $_POST["email"];
    $mot_de_passe = password_hash($_POST["mot_de_passe"], PASSWORD_DEFAULT);
    $role = "utilisateur";


    $check = $conn->prepare("SELECT * FROM utilisateurs WHERE email = :email");
    $check->execute(['email' => $email]);
    if ($check->rowCount() > 0) {
        $message = "<p style='color:red;'>Cet email est déjà utilisé.</p>";
    } else {
        $sql = "INSERT INTO utilisateurs (nom, prenom, email, mot_de_passe, role)
                VALUES (:nom, :prenom, :email, :mot_de_passe, :role)";
        $stmt = $conn->prepare($sql);
        $success = $stmt->execute([
            ':nom' => $nom,
            ':prenom' => $prenom,
            ':email' => $email,
            ':mot_de_passe' => $mot_de_passe,
            ':role' => $role
        ]);

        if ($success) {
            header("Location: login.php");
            exit();
        } else {
            $message = "<p style='color:red;'>Erreur lors de l'inscription.</p>";
        }
    }
}
```

Pour s'inscrire , on vérifie si le formulaire a été soumis en POST lorsque l'utilisateur clique sur s'inscrire

On récupère les infos saisies par l'utilisateur. le mot de passe est crypté avec password_hash et le rôle par défaut est utilisateur il faut changer le rôle via la base de donnée.

ensuite on vérifie si l'adresse est déjà utilisée dans la table utilisateur et ca envoie les données dans la table si tout est bon et ca envoie l'utilisateur sur la page login .

```php
require '../config/database.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $email = $_POST["email"];
    $password = $_POST["mot_de_passe"];

    $sql = "SELECT * FROM utilisateurs WHERE email = :email";
    $stmt = $conn->prepare($sql);
    $stmt->execute(['email' => $email]);
    $user = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($user && password_verify($password, $user["mot_de_passe"])) {
        $_SESSION["user_id"] = $user["id"];
        $_SESSION["role"] = $user["role"];

        if ($user["role"] === "admin") {
            header("Location: ../Admin/indexAdmin.php");
```

On récupère les données du formulaire ensuite on cherche dans la BDD l'utilisateur par son email et si on la trouve et on récupère toutes les données.

ensuite on compare les mots de passe avec deux conditions ; si l'utilisateur existe et si le mot de passe est correct.
on enregistre l'utilisateur dans la session si il est admin ou utilisateur
et ça le redirige vers la page correspondante.

10/ Difficultés rencontrées

-difficultés à se répartir les tâches et à avoir un travail équitable sur l'ensemble du projet.
- détail à régler dans le projet comme la partie sur les commentaires ou la date limite à rendre
-groupe composé de 1 dev et 3 réseaux ce qui à particulièrement été compliqué sur la partie dev
- code qui n'as pas pu suivre une architecture MVC avec l'utilisation de fichier (entité, contrôleur,modem, router, et twig) en raison de connaissances diversifiées.
-difficulté à répondre à l'entièreté du sujet demandé
- difficultés rencontrés sur git hub lorsqu'on voulait "push" ou "pull" on avait des messages d'erreur comme "permission denied" ou des conflits ce qui retardait le travail.

11/ Améliorations

-Utilisation de l'architecture MVC et du CRUD avec les entités ,models , contrôleurs ect afin d'avoir un code très propres et bien rangé
- Amélioration du code et des du css pour certaines pages
-Améliorer la partie commentaires , date de limite et la partie admin pour voir les projets des élèves selon la matière.
-Améliorer la répartition du travail au sein du groupe.

12/ Conclusion :

Ce projet de groupe avait pour objectif de développer une plateforme web en PHP, HTML et CSS permettant aux enseignants de créer des dépôts de devoirs, et aux étudiants d'y déposer leurs travaux.

Ce travail nous a permis de consolider nos compétences en développement web (formulaires, gestion des fichiers, base de données, sessions) tout en apprenant à collaborer efficacement en équipe. Il nous a également sensibilisés à l'importance de la répartition des tâches, de la communication et de l'entraide dans un projet collectif.