.

13/03/2025

Une image contenant logo, Police, texte, Graphique

Le contenu généré par l’IA peut être incorrect.

## Rapport de projet

## METAIS Romain

## Projet ACCÈS LYCÉE

Badge RFID et Interface de gestion

# Table des matières

[Projet ACCÈS LYCÉE 0](#_Toc199200586)

[Table des matières 1](#_Toc199200587)

[Introduction Générale 2](#_Toc199200588)

[But du projet 4](#_Toc199200589)

[Diagramme de Gantt 5](#_Toc199200590)

[Répartition des tâches 6](#_Toc199200591)

[Analyse UML 7](#_Toc199200592)

[Diagrammes 8](#_Toc199200593)

[Interfaces web 15](#_Toc199200594)

[1.Interface d’authentification 17](#_Toc199200595)

[1.a – Interface de connexion HTML 19](#_Toc199200596)

[1.b – Traitement de l’authentification 20](#_Toc199200597)

[2 - Espace Agent d’accueil 22](#_Toc199200598)

[2.a – Contrôle d'accès à l'espace Agent 22](#_Toc199200599)

[2.b – Interface utilisateur : Agent 23](#_Toc199200600)

[2.c – Déconnexion de l’espace personnel 23](#_Toc199200601)

[2.1 - Gestion caméra et ouverture de la porte 24](#_Toc199200602)

[2.1.a – Affichage du flux vidéo 26](#_Toc199200603)

[2.1.b – Contrôle de la caméra 27](#_Toc199200604)

[3 - Espace Technicien 27](#_Toc199200605)

[3.a – Interface utilisateur : Technicien 28](#_Toc199200606)

[3.1 – Historique des passages 28](#_Toc199200607)

[3.1.a – Filtrage personnalisé des résultats 29](#_Toc199200608)

[3.1.b – Limitation des résultats 30](#_Toc199200609)

[3.2 - Ajout de membre 31](#_Toc199200610)

[3.2.a – Traitement des données 31](#_Toc199200611)

[3.2.b – Insertion du membre dans la base de données 32](#_Toc199200612)

[3.3 - Suppression de membre 33](#_Toc199200613)

[3.3.a – Affichage des membres dans la liste déroulante 34](#_Toc199200614)

[3.3.b - Suppression Sécurisée avec Transaction 34](#_Toc199200615)

[3.4 – Gestion des badges 35](#_Toc199200616)

[4 - Problèmes rencontrés 37](#_Toc199200617)

[5 – Bilan 38](#_Toc199200618)

[ANNEXES 39](#_Toc199200619)

# Introduction Générale



Le BTS CIEL est une formation de deux ans qui prépare les étudiants à intégrer le secteur de l’informatique. L’un des aspects majeurs de cette formation est le Projet E6. Il s’agit d’un travail pratique permettant de mettre en œuvre les compétences acquises tout au long du cursus. Ce projet constitue également une opportunité de valoriser l’esprit d’analyse ainsi que le sens de l’organisation des apprenants.

Présentation

Dans un monde où la sécurité et la gestion des accès sont devenues des enjeux majeurs, il est essentiel de mettre en place des systèmes performants et fiables.

Ce projet vise à reproduire et développer un système de gestion des accès basé sur le lycée Pilote Innovant International.

Une image contenant plein air, bâtiment, ciel, portail

Le contenu généré par l’IA peut être incorrect.Une image contenant Électroménager, machine, intérieur, mur

Le contenu généré par l’IA peut être incorrect.

**Matériels Utilisés :**

|  |  |
| --- | --- |
| Désignation | Caractéristiques |
| * 2 lecteurs de badges et des badges RFID * Un volet * Une caméra IP * Un capteur de présence photoélectrique | * De la marque Invéo et Minova * Matériel de récupération * De la marque D-Link DCS-5000L * De la marque OSIRIS |

**Logiciels Utilisés :**

|  |  |
| --- | --- |
| Désignation | Caractéristiques |
| Base de données :  Pilotage du Barionet :  Création des interfaces : | Moteur :  Applications :  Applications :    Applications : |

# But du projet

L’objectif principal est d’automatiser l’identification des utilisateurs grâce à des badges RFID, tout en permettant une gestion centralisée des accès via un serveur. Ce système devra être capable de :

* Lire et vérifier les informations contenues dans un badge RFID
* Accorder ou refuser l’accès en fonction des droits attribués à l’utilisateur
* Enregistrer les événements d’accès dans une base de données pour assurer un suivi
* Intégrer une interface de gestion pour les administrateurs et techniciens.

Pour mener à bien ce projet, nous avons adopté une approche structurée en plusieurs étapes :

1. Analyse des besoins et conception du système, en nous basant sur les diagrammes UML pour modéliser l’architecture et les interactions.
2. Mise en place de l’infrastructure matérielle, incluant des lecteurs RFID, un serveur, une base de données et un module Barionnet100 permettant de piloter les actionneurs.
3. Développement du système, avec l’implémentation du logiciel en utilisant des technologies comme PHP et MySQL.
4. Tests et validation, afin d’assurer le bon fonctionnement du dispositif et sa robustesse.

Ce document détaille l’ensemble des aspects du projet, depuis la conception jusqu’à l’intégration finale du système, en passant par les choix techniques et les contraintes rencontrées.

# Diagramme de Gantt

Une image contenant texte, capture d’écran, diagramme, ligne

Le contenu généré par l’IA peut être incorrect.

Le diagramme de Gantt est un outil qui permet de représenter visuellement l’état et l’avancement des différentes tâches qui constituent le projet.

Celui ci-dessus est séparé en 4 phases et relate l’avancement de chaque tâche que nous avons ou que nous sommes en train d’effectuer sur le projet :

* Analyse des besoins (Phase 1)
* Conception UML (Phase 2)
* Conception Interface Authentification (Phase 2)
* Création de Page Web Sorties Barionet (Phase2)
* Conception Site Web Technicien/Agent Barionet (Phase3)
* Création Interface Gestion d’Accès (Phase3)
* Conception Base de Données (Phase4)
* Ouverture des Portes via les badges (Phase4)
* Conception Script PHP(Phase4)
* Interface Historique (Phase4)
* Diagramme de séquences (Phase4)
* Technologie RFID (Phase4)
* Principe de développement (Phase4)
* Ajout de membres/Suppression (Phase4)
* Interface de Gestion des badges (Phase4)
* Hachage de mots de passe (Phase4)
* Script .htaccess blocage d’accès (Phase4)
* Script Php test pour vérifier le fonctionnement du lecteur Minova(Phase4)
* Diagramme de séquence (Phase4)

# Répartition des tâches

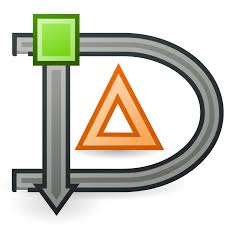
|  |  |
| --- | --- |
| Etudiant | Tâches |
| Akouman DIANGO | * Pilotage du Barionet * Mise en oeuvre du lecteur RFID Minova |
| Elias COSME VINOU | * Conception UML * Mise en place et structuration de la base de données * Création du Script Php de connexion à la base de données |
| Romain METAIS | * Développement de l’interface de l’agent d’accueil * Développement de l’interface du technicien * Communication avec le système RFID |

# Analyse UML

**Langage UML**

Le langage de modélisation unifié (UML) est le langage standard que de nombreux ingénieurs logiciels et de nombreuses entreprises utilisent pour avoir une vue d'ensemble de systèmes complexes.

Les différents diagrammes que nous allons vous proposer sont des diagrammes créés avec ce langage.



**DIA**

Dia est un logiciel libre de création de diagramme.

Dia est conçu de manière modulaire avec plusieurs paquetages de formes pour des besoins différents : diagramme de flux, diagramme de circuit électrique, diagramme UML, etc.

A partir de ce logiciel nous avons conçu les différents diagrammes un à un. Les diagrammes UML les plus adaptés à notre projet sont :

* Diagramme de déploiement
* Diagramme de cas d’utilisation
* Diagramme de séquences

# Diagrammes

**Diagramme de déploiement**

En UML, les diagrammes de déploiement modélisent l'architecture physique d'un système. Ces derniers affichent les relations entre les composants logiciels et matériels du système, d'une part, et la distribution physique du traitement, d'autre part.

Ils présentent la disposition physique des nœuds dans un système réparti, les artefacts qui sont stockés sur chaque nœud et les composants et autres éléments que les artefacts implémentent. Les nœuds représentent des périphériques matériels tels que des ordinateurs, des détecteurs et des imprimantes, ainsi que d'autres périphériques qui prennent en charge l'environnement d'exécution d'un système. Les chemins de communication et les relations de déploiement modélisent les connexions dans le système.

Une image contenant texte, diagramme, Plan, ligne

Le contenu généré par l’IA peut être incorrect.

Partie de Akouman DIANGO

Partie de Elias COSME VINOU

Partie de Romain METAIS

Ce diagramme de déploiement représente l'architecture d'un système utilisant des lecteurs RFID, un serveur, des postes de travail et des périphériques connectés via un réseau Ethernet. Voici une description détaillée de chaque composant :

**Serveur (traits unis, partie de Elias COSME VINOU)**

* Héberge Apache (pour gérer l'application web).
* Contient une base de données (stockage des informations RFID et autres données du système).

**Poste technicien (traits en pointillé, partie de Romain METAIS)**

* Connecté via Ethernet au réseau.
* Relié à un Lecteur RFID INVEO USB Desk via USB, permettant la lecture des badges RFID.

**Poste agent d'accueil (traits en pointillé, partie de Romain METAIS)**

* Connecté au réseau via Ethernet.
* Associé à une Caméra IP D-Link DCS pour la surveillance.

**Switch (élément central du réseau)**

* Connecte les différents équipements via Ethernet.

**Barionnet100 (partie de Akouman DIANGO)**

* Connecté au réseau Ethernet.
* Probablement un contrôleur ou un module de communication.

**Lecteur RFID Minova MCR04 (en rouge, partie de Akouman DIANGO)**

* Connecté au réseau via Ethernet.
* Utilisé pour lire les badges RFID.

**Diagramme de cas d’utilisation**

En langage de modélisation unifié (UML), un diagramme de cas d'utilisation peut servir à résumer les informations des utilisateurs de votre système (également appelés acteurs) et leurs interactions avec ce dernier. La création de ce type de diagramme UML requiert un ensemble de symboles et de connecteurs spécifiques. Lorsqu'ils sont bien conçus, les diagrammes de cas d'utilisation peuvent aider notre équipe à collaborer et représenter :

* Les scénarios dans lesquels notre système ou application interagit avec des personnes, des organisations ou des systèmes externes ;
* Les objectifs que notre système ou application permet aux entités (appelées acteurs) d'atteindre ;
* La portée de notre système.

Les diagrammes sont généralement composés :

* D’acteurs : utilisateurs qui interagissent avec un système. Un acteur peut être une personne, une organisation ou un système externe qui interagit avec votre application ou votre système. Il s'agit nécessairement d'objets externes qui produisent ou consomment des données.
* Du système : séquence spécifique d'actions et d'interactions entre les acteurs et le système. Un système peut également être appelé scénario.
* Des objectifs : résultat final de la plupart des cas d'utilisation. Un diagramme réussi doit décrire les activités et les variantes utilisées pour atteindre l'objectif.

Une image contenant texte, diagramme, cercle, ligne

Le contenu généré par l’IA peut être incorrect.

Dans notre diagramme de cas d’utilisation les acteurs sont :

* Le visiteur
* L’agent d’accueil
* Le technicien

La balise <<extend>> présente dans le diagramme indique une relation optionnelle entre deux cas d’utilisation. Elle signifie qu’un cas d’utilisation (appelé extension) peut être inséré, sous certaines conditions, dans l’exécution d’un autre cas d’utilisation (appelé de base).

Dans notre diagramme, le cas d’utilisation « Identifier le visiteur » possède une extension vers le cas d’utilisation « Autoriser manuellement l’accès ». Cette relation signifie que, lors de l’identification d’un visiteur, il est parfois nécessaire pour l’agent d’accueil d’autoriser manuellement l’accès. Mais cette action n’est pas systématique : elle ne se produit que si une condition particulière est remplie (par exemple, un visiteur non reconnu par le système ou nécessitant une autorisation exceptionnelle).

Nous pouvons retenir que :

* Le cas d'utilisation « Identifier le visiteur » se déroule normalement.
* Mais, dans certains scénarios (non automatiques), une action supplémentaire (« Autoriser manuellement l’accès ») peut être exécutée, déclenchée par le cas principal.
* La relation <<extend>> permet donc de modéliser des options supplémentaires ou des “ajouts” qui ne sont pas toujours réalisés.

Ainsi, l’utilisation de <<extend>> rend le diagramme plus flexible et plus précis en illustrant les exceptions ou déroulés alternatifs possibles.



Peut accéder par badges et aussi s’identifier auprès de l’agent d’accueil.



Peut gérer les modes d’accès, consulter les accès, identifier le visiteur et il peut optionnellement autoriser manuellement l’accès.



Peut gérer les badges utilisateurs, configurer le système, gérer les droits et il peut aussi consulter les accès.

**Diagramme de séquences**

Un diagramme de séquences permet de visualiser, étape par étape, le déroulé logique des échanges entre différents acteurs et systèmes lors d’une action précise. Ici, il schématise le contrôle d’accès par badge dans un établissement (école, collège, lycée).

Une image contenant texte, diagramme, Parallèle, ligne

Le contenu généré par l’IA peut être incorrect.

Dans le cadre d’un système de gestion des accès par badges, ce diagramme ci-dessus illustre le scénario typique pour contrôler l’accès à une porte via un badge RFID.

Ce que montre ce diagramme dans le contexte scolaire :

Imaginons un utilisateur souhaitant entrer dans l’établissement :

**Scannage du badge**

L’utilisateur approche son badge RFID du lecteur situé à l’entrée.

**Transmission de l’identifiant**

Le lecteur RFID lit l’identifiant du badge (id\_badge) et le transmet à un script PHP sur le serveur.

**Vérification des droits d’accès**

Le script PHP interroge la base de données contenant la liste des élèves autorisés à entrer :

* Il vérifie si cet élève est autorisé à accéder à l’établissement (présence dans la base, statut "autorisé", etc.).

**Bloc alt**

Le bloc alt est une notation spécifique à l’UML utilisée dans les diagrammes de séquences.  
Il signifie "alternative".

Le bloc alt permet de modéliser une condition ou un choix dans le scénario décrit.

Il indique qu’il existe plusieurs chemins possibles à partir d’un point donné du processus, en fonction d’un résultat ou d’une condition.

Le bloc alt sépare visuellement ces deux alternatives dans le diagramme, montrant que le système agit différemment selon le résultat de la vérification.

**Retour du résultat**

La base de données envoie la réponse (OK : accès autorisé, NOK : accès refusé) au script PHP, qui la transmet au lecteur RFID.

**Décision d’accès**

Le bloc alt est utilisé pour spécifier les deux scénarios possibles après la vérification du badge :

* Si le résultat est OK : L’élève peut entrer, la porte s’ouvre ou un signal sonore/lumineux s’active pour valider.
* Si le résultat est NOK : L’accès est refusé, la porte reste verrouillée et l’élève est informé.

**Boucle d’attente**

Le système attend le prochain passage d’un badge.

Ce diagramme permet d’apporter une vision claire des interactions entre l’utilisateur, le matériel et le serveur. Et il apporte une identification des étapes clés pour le contrôle d’accès.

L’élève scanne son badge → Le lecteur envoie l’identifiant au serveur → Vérification dans la base élève → Accès autorisé ou refusé selon les droits enregistrés.

Ce diagramme est essentiel pour garantir la gestion sécurisée et fluide des accès des élèves à l’établissement.

# Interfaces web

Dans le cadre du développement des interfaces web de notre projet, nous avons choisi l'environnement de développement intégré Visual Studio Code comme outil principal. Cette plateforme nous a permis de concevoir et d'implémenter efficacement l'ensemble des interfaces utilisateur requises. Nous nous sommes servis de plusieurs langages de programmation tels que HTML pour structurer le contenu, PHP pour gérer les aspects dynamiques et les interactions avec la base de données, et CSS pour styliser et mettre en forme l'ensemble des éléments visuels.

Une image contenant Graphique, logo, symbole, conception

Le contenu généré par l’IA peut être incorrect.Afin de pouvoir tester nos pages PHP et interagir avec la base de données en local, nous avons utilisé XAMPP Control Panel. Cet outil nous permet de lancer facilement un serveur local en activant les services Apache et MySQL. Grâce à XAMPP, nous pouvons accéder à nos interfaces via l’adresse localhost dans un navigateur et vérifier le bon fonctionnement de nos scripts en temps réel.

Une image contenant rouge, symbole, logo, conception

Le contenu généré par l’IA peut être incorrect.Le langage **HTML (HyperText Markup Language)** est un type de langage informatique qui décrit au navigateur comment afficher la page web. Il nous a servi à faire des formulaires, c’est-à-dire toutes les informations qui sont affichées sur l’interface de la page et qui permettent d’interagir. Par exemple pour la page Index.html (celle qui sert de page d’authentification), nous utilisons un formulaire pour afficher les champs « email » ou « mot de passe » mais également un bouton sur lequel nous pouvons appuyer pour valider la connexion.

Une image contenant symbole, Police, cercle, logo

Le contenu généré par l’IA peut être incorrect.Le langage **PHP (PHP Hypertext Preprocessor)** est un langage de scripts généraliste et Open Source, spécialement conçu pour le développement d'applications web. HTML permet de créer la structure et le contenu d'une page web tandis que PHP permet de créer des pages web dynamiques en traitant les informations côté serveur. Nous avons utilisé ce langage pour les pages ayant besoin d’aller vérifier des informations dans la base de données. Par exemple, la page Index.html a besoin d’une page traitement\_connexion.php qui permet de vérifier dans la base de données si les informations entrées dans les champs email et mot de passe correspondent bien à celles présentes dans la base de données. Ainsi l’accès à l’espace personnel de l’utilisateur web sera accepté ou refusé.

Une image contenant symbole, Rectangle, conception, Bleu électrique

Le contenu généré par l’IA peut être incorrect.Le langage **CSS (Cascading Style Sheets)** correspond à un langage informatique permettant de mettre en forme des pages web, notamment les pages HTML. Ce langage m’a permis d’améliorer la présentation de mes pages grâce à des images, en modifiant la couleur du texte, la police ou bien l’emplacement des éléments sur la page.

## 1.Interface d’authentification

Une image contenant texte, capture d’écran, Police, conception

Le contenu généré par l’IA peut être incorrect.

Cette interface d'authentification sert de **point d'entrée sécurisé** pour deux types d'utilisateurs : **l'agent d'accueil** et **le technicien**. Elle permet à chacun d'**accéder** à son **espace de travail** personnalisé en saisissant des **identifiants** et **mot de passe** préalablement enregistrés dans la base de données du système. Nous avons utilisé un programme permettant de **générer** des **mots de passe hachés** tous différents à chaque exécution, nous les avons ensuite insérés dans la base de données dans le champ **« mot\_de\_passe »** de la table **« utilisateur\_web »** pour ne pas mettre les mots de passe en **clairs** mais plutôt **chiffrés** afin de limiter les risques d’attaques.

Code source du hachage de mot de passe :  
  
Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.  
  
Le code PHP ci-dessus utilise la fonction « **password\_hash()** » avec l’option **PASSWORD\_DEFAULT** ce qui utilise l’algorithme **BCRYPT**.

Résultat après exécution :  
  
Une image contenant texte, Police, capture d’écran, ligne

Le contenu généré par l’IA peut être incorrect.

Les mots de passe générés par cet algorithme commencent toujours par « $2y$10$ ». Ceci a une signification particulière :

* Le préfixe **$2y$10$** indique :
* 2y = **version BCRYPT** (PHP)
* 10 = **coût**
* La taille du mot de passe sera toujours de **60 caractères** :
* 4 caractères pour l’en-tête : $2y$10$
* 2 chiffres pour le coût : 10
* 1 séparateur : $
* 22 caractères pour le sel
* 31 caractères pour le hash
* **password\_hash()** ajoute automatiquement un **sel** ce qui veut dire que les mots de passe sont différents à chaque exécution, même si le mot de passe est le même.

Le **coût** appelé **« cost factor »** ou **« work factor »** est un **paramètre de sécurité** qui détermine à quel point **le hachage du mot de passe** est « **lent** » à calculer.

Le **coût** est une **puissance de 2**.  
Par exemple :

* Coût = 10 → 2¹⁰ = **1 024 itérations**
* Coût = 12 → 2¹² = **4 096 itérations**
* Coût = 14 → 2¹⁴ = **16 384 itérations**

Plus le coût est **élevé**, plus le temps nécessaire pour calculer le hash est **long**.

Dans notre cas, le coût est de 2^10 soit **1024 itérations**.

Aperçu dans la base de données :  


### 1.a – Interface de connexion HTML

L'utilisateur saisit son email et son mot de passe à l'aide d’un formulaire HTML. Ces informations sont ensuite transmises au script PHP « **traitement\_connexion.php »**.

                <form method="POST" action="traitement\_connexion.php">

                    <div class="inputbox">

                        <ion-icon name="mail"></ion-icon>

                        <input type="email" name="email" required>

                        <label>Email</label>

                    </div>

                    <div class="inputbox">

                        <ion-icon name="lock"></ion-icon>

                        <input type="password" name="password" required>

                        <label>Mot de passe</label>

                    </div>

                    <button type="submit">Connexion</button>

                </form>

* Le formulaire envoie les données au serveur via traitement\_connexion.php.
* Le champ email doit être de type valide et rempli (required).
* Le champ password masque la saisie et est lui aussi requis.
* Les icônes sont intégrées avec la bibliothèque **Ionicons**.

### 1.b – Traitement de l’authentification

Nous avons créé une page « traitement\_connexion.php » permettant de se connecter à la base de données pour **vérifier** la correspondance avec l’**adresse** **électronique et le mot de passe** qui ont été entrés dans les champs correspondants. Ainsi, l’utilisateur sera redirigé vers l’espace qui correspond à son rôle.

$email = trim($\_POST['email']);

    $password = trim($\_POST['password']);

    try {

        $pdo = new PDO(DB\_DSN, DB\_USER, DB\_PASS, $pdo\_options);

        $sql = $pdo->prepare("SELECT id\_utilisateur\_web, login, mot\_de\_passe, role FROM utilisateur\_web WHERE login = ?");

        $sql->execute([$email]);

        $user = $sql->fetch(PDO::FETCH\_ASSOC);

        if ($user) {

            if (password\_verify($password, $user['mot\_de\_passe'])) {

                $\_SESSION['user\_id'] = $user['id\_utilisateur\_web'];

                $\_SESSION['user\_email'] = $user['login'];

                $\_SESSION['user\_role'] = $user['role'];

                if ($user['role'] == 'Agent') {

                    header("Location: Agent/espace\_agent.php");

                } elseif ($user['role'] == 'Technicien') {

                    header("Location: Technicien/espace\_technicien.php");

                }

                exit();

            } else {

                header("Location: Index.html?error=motdepasse");

                exit();

            }

* Connexion à la base via PDO.
* Requête sécurisée avec **prepare()** pour éviter l'injection SQL.
* Vérification du mot de passe via **password\_verify().**
* **Initialisation de la session** si les identifiants sont valides :

**-** $\_SESSION['user\_id'] : identifiant unique de l’utilisateur.

- $\_SESSION['user\_email'] : login (email) de l’utilisateur.

- $\_SESSION['user\_role'] : rôle de l’utilisateur, utilisé pour contrôler les accès.

* **Utilité des sessions** :

- Ces variables de session permettent de **maintenir l’état de connexion** entre plusieurs pages.

- Elles sont ensuite utilisées pour sécuriser les autres pages du site (ex : vérification avec isset($\_SESSION['user\_id'])), ou restreindre certaines interfaces selon le rôle ($\_SESSION['user\_role']).

**- Redirection** automatique vers l’espace correspondant selon le rôle de l’utilisateur (Agent ou Technicien).

## 2 - Espace Agent d’accueil

Une image contenant texte, capture d’écran, Bleu électrique, Police

Le contenu généré par l’IA peut être incorrect.

Voici **l’espace personnel de l’agent d’accueil**. Il a accès à une **caméra IP, la gestion des modes d’accès et un historique**. Un **bouton de déconnexion** est à disposition pour quitter son espace et revenir à la page d’authentification.

### 2.a – Contrôle d'accès à l'espace Agent

Ce fichier PHP protège l'accès à l'interface "Espace Agent". Il vérifie si l'utilisateur est connecté et s'il possède le bon rôle.

**Code source (PHP – contrôle d’accès):**

<?php

session\_start();

// Vérifier que l'utilisateur a le rôle Agent

if ($\_SESSION['user\_role'] != 'Agent') {

    header("Location: ../Index.html?error=acces\_non\_autorise");

    exit();

}

?>

* La session est démarrée avec **session\_start()**.
* Si l'utilisateur n’a pas le bon rôle, il est **redirigé vers la page de connexion** avec un paramètre d’erreur.
* Seuls les utilisateurs ayant le rôle **Agent** peuvent accéder à cette page.

### 2.b – Interface utilisateur : Agent

Une fois l’utilisateur autorisé, il accède à une interface simple lui donnant accès à ses fonctionnalités : gestion de la caméra IP et historique des accès.

<h1>Espace Agent</h1>

    <ul>

    <li><p><a href="Camera.php">Caméra IP/Gestion d'accès</a></p></li>

     <li><p><a href="HistoriqueAgent.php">Historique des passages</a></p></li>

    </ul>

     <a href="deconnexion.php" class="button-danger">Se déconnecter</a>

* Accès à la gestion de la caméra IP,
* Consultation de l’historique des accès,
* Déconnexion sécurisée via un lien dédié.

### 2.c – Déconnexion de l’espace personnel

<?php

session\_start();

// Détruire toutes les données de la session

session\_unset();     // Supprime toutes les variables de session

session\_destroy();   // Détruit la session

// Rediriger vers la page de connexion

header("Location: /Projet\_Acces\_lycee/Interfaces\_Web/Index.html");  // Chemin qui redirige vers la page de connexion

exit();

?>

## 2.1 - Gestion caméra et ouverture de la porte

Une image contenant texte, meubles, capture d’écran, ordinateur

Le contenu généré par l’IA peut être incorrect.

Cette interface montre un aperçu du **flux vidéo** de la caméra IP au centre de la page, ainsi que des **boutons pour contrôler la direction de la caméra** sur la droite. Elle possède également une partie **« Gestion d’accès »** à gauche de la page qui permet de **choisir le mode de fonctionnement**, soit un accès **par badge** ou la possibilité que ce soit l’agent qui ouvre en cliquant sur le bouton **« OUVRIR »**, soit un **mode automatique** c’est-à-dire que lorsque l’on se présente devant la porte les capteurs détectent une présence qui enclenche l’ouverture de celle-ci.

Une image contenant intérieur, Appareils électroniques, Appareil électronique, Périphérique de sortie

Le contenu généré par l’IA peut être incorrect.

La caméra utilisée est de la marque **D-Link**, le modèle est **DCS-5000L**.Cette caméra sert à avoir un aperçu de la personne se présentant à l'entrée afin de vérifier si son accès est autorisé. De plus, elle peut être orientée dans toutes les directions grâce à des commandes dédiées.

Concernant le fonctionnement de la caméra D-Link DCS-5000L, nous avons utilisé un logiciel appelé **« Advanced IP Scanner »** permettant d’afficher toutes les adresses IP des appareils connectés au réseau du lycée.

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Le contenu généré par l’IA peut être incorrect.

Ainsi, nous avons pu **déterminer l’adresse IP** de la caméra puis en tapant l’adresse IP que nous avons trouvé dans la barre de recherche, nous avons pu accéder à une page D-Link permettant d’avoir un aperçu du flux vidéo mais surtout de la configurer en mettant une **adresse IP fixe** pour qu’elle ne change pas lorsque que l’on débranche et rebranche le câble réseau.

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Le contenu généré par l’IA peut être incorrect.

### 2.1.a – Affichage du flux vidéo

Afin d’obtenir le flux vidéo nous avons dû récupérer le lien permettant d’y accéder. Pour cela, nous avons trouvé une documentation D-Link nous indiquant la ligne de commande à insérer dans le code pour **afficher le flux vidéo**.

<img *src*="[http://[ADRESSE\_IP\_DE\_LA\_CAMERA]/video/mjpg.cgi](http://172.22.20.29/video/mjpg.cgi)"/>

Pour notre cas, cette ligne de code a donné :

<img src="http://172.22.20.29/video/mjpg.cgi" alt="Flux vidéo caméra D-Link">

### 2.1.b – Contrôle de la caméra

<h2>Contrôle de la caméra</h2>

          <form action="http://172.22.20.29/pantiltcontrol.cgi" method="post">

              <input type="hidden" name="PanSingleMoveDegree" value="5">

              <input type="hidden" name="TiltSingleMoveDegree" value="5">

              <input type="hidden" name="PanTiltSingleMove" value="0">

              <input type="submit" value="↖">

          </form>

Chaque bouton de la grille permet de **contrôler les mouvements de la caméra IP**. Le formulaire envoie une requête **POST** à l’adresse IP de la caméra avec les paramètres de direction. Par exemple, value="0" contrôle un mouvement vers le haut-gauche.

## 3 - Espace Technicien

Une image contenant texte, capture d’écran, technologie

Le contenu généré par l’IA peut être incorrect.

Voici maintenant **l’espace** personnel du **technicien**. Lui, a accès à un **historique des passages**, il peut aussi **ajouter des membres, supprimer des membres** et aussi s’occuper de la **gestion des badges**. Comme l’agent d’accueil, il possède également un **bouton de déconnexion** pour revenir à la page d’authentification.

### 3.a – Interface utilisateur : Technicien

<h1>Espace Technicien</h1>

     <ul>

     <li><p><a href="HistoriqueTechnicien.php">Historique des passages</a></p>

    </li>

    <li><p><a href="AjouterMembre.php">Ajouter un membre</a></p>

    </li>

    <li><p><a href="SupprimerMembre.php">Supprimer un membre</a></p>

    </li>

    <li><p><a href="GestionBadges.php">Gestion des badges</a></p>

    </li>

    </ul>

     <a href="deconnexion.php" class="button-danger">Se déconnecter</a>

* Consulter les historiques d’accès.
* Gérer les membres (ajout, suppression).
* Gérer les badges.
* Déconnexion sécurisée via un lien dédié.

## 3.1 – Historique des passages

Une image contenant texte, capture d’écran, nombre, logiciel

Le contenu généré par l’IA peut être incorrect.

L’historique permet au technicien de visualiser les accès enregistrés dans la base de données. Il intègre un système de filtres (nom, prénom, badge, rôle, date, heure, statut) et offre un affichage dynamique et personnalisable. L’ensemble du système est sécurisé via la gestion de session et l’utilisation de requêtes préparées (voir ANNEXES).

### 3.1.a – Filtrage personnalisé des résultats

Nous pouvons également **filtrer** l’historique en recherchant **un nom, un prénom, un rôle, ou un numéro de badge** en particulier. Nous avons aussi la possibilité de filtrer par **date** et en fonction de **l’heure** qui nous intéresse.

Exemple de filtrage :

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Résultat après filtrage :

Une image contenant capture d’écran, texte, nombre, ligne

Le contenu généré par l’IA peut être incorrect.

### 3.1.b – Limitation des résultats

Nous pouvons afficher plus ou moins d’enregistrements en fonction de ce que l’on souhaite par exemple 50 si l’on souhaite que ce ne soit pas trop encombrant ou plus si l’on veut voir des enregistrements plus anciens.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

<select id="limit" name="limit">

    <option value="50" <?= $limit == 50 ? 'selected' : '' ?>>50</option>

    <option value="100" <?= $limit == 100 ? 'selected' : '' ?>>100</option>

    <option value="200" <?= $limit == 200 ? 'selected' : '' ?>>200</option>

    <option value="500" <?= $limit == 500 ? 'selected' : '' ?>>500</option>

    <option value="1000" <?= $limit == 1000 ? 'selected' : '' ?>>1000</option>

</select>

$limit = isset($\_GET['limit']) ? (int)$\_GET['limit'] : 50;

$query = "... LIMIT :limit";

$stmt->bindValue(':limit', $limit, PDO::PARAM\_INT);

* La valeur sélectionnée est transmise via l’URL en **GET**.
* Le **LIMIT :limit** dans la requête SQL limite le nombre de lignes renvoyées.
* Cela améliore les performances sur de gros volumes de données.

## 3.2 - Ajout de membre

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

L’ajout d’un membre se fait par le biais de cette interface dans laquelle il faut renseigner **le nom, le prénom, le code du badge** qui va être attribué à l’utilisateur ainsi que **son rôle** (lycéen, étudiant, personnel, etc.). Il ne reste plus qu’à cliquer sur le bouton **« Ajouter »**. Grâce à ces informations, l’utilisateur sera directement ajouté dans la base de données.

### 3.2.a – Traitement des données

// Vérifier si tous les champs sont remplis

if (!empty($nom) && !empty($prenom) && !empty($code\_badge) && !empty($nom\_role)) {

// Vérifier si le code du badge existe déjà

$stmt = $pdo->prepare("SELECT code\_badge FROM badge WHERE code\_badge = ?");

$stmt->execute([$code\_badge]);

$badge\_existant = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($badge\_existant) {

    echo "<p style='color: red;'>Erreur : Ce code de badge existe déjà dans la base de données.</p>";

}else {

   // Récupérer l'ID du rôle sélectionné

   $stmt = $pdo->prepare("SELECT id\_role FROM role WHERE nom = ?");

   $stmt->execute([$nom\_role]);

  $role = $stmt->fetch(PDO::FETCH\_ASSOC);

* Vérifie la **présence des champs** et l’**unicité du badge**
* Une requête sécurisée vérifie si le rôle existe dans la table **rôle**

### 3.2.b – Insertion du membre dans la base de données

Si les vérifications sont bonnes, deux insertions sont faites :

* Dans la table **utilisateur**
* Puis dans la table **badge** (en associant l’utilisateur via son **id)**

// Insérer l'utilisateur

$stmt = $pdo->prepare("INSERT INTO utilisateur (nom, prenom, id\_role) VALUES (?, ?, ?)");

if ($stmt->execute([$nom, $prenom, $id\_role])) {

    $id\_utilisateur = $pdo->lastInsertId();

// Insérer le badge avec l'id\_utilisateur et initialiser l'état à 1 (actif)

$stmt = $pdo->prepare("INSERT INTO badge (code\_badge, id\_utilisateur, etat) VALUES (?, ?, 1)");

if ($stmt->execute([$code\_badge, $id\_utilisateur])) {

    echo "<p style='color: green;'>Utilisateur et badge ajoutés avec succès.</p>";

} else {

    echo "<p style='color: red;'>Erreur lors de l'ajout du badge.</p>";

} else {

    echo "<p style='color: red;'>Erreur lors de l'ajout de l'utilisateur.</p>";

}

* Deux requêtes **INSERT** sont liées : le badge dépend de l’utilisateur.
* **lastInsertId()** récupère l’ID automatiquement généré.
* Statut du badge initialisé à 1 (actif).
* Chaque requête est sécurisée via **prepare + execute**.

## 3.3 - Suppression de membre

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Le technicien a la possibilité de supprimer un membre via cette interface.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Un menu déroulant permet d’afficher l’ensemble des membres enregistrés dans la base de données. Il est également possible d’effectuer une recherche en saisissant le nom du membre à supprimer, ce qui facilite et accélère la sélection. Une fois le membre identifié, un clic sur le bouton **« Supprimer »** déclenche la procédure de suppression, suivi d’une demande de confirmation afin de garantir la sécurité de l’action.

### 3.3.a – Affichage des membres dans la liste déroulante

Avant d'afficher la liste dans le formulaire, la page interroge la base pour récupérer tous les membres et leurs rôles.

// Récupérer tous les membres disponibles

$membres = [];

$stmt = $pdo->query("SELECT u.id\_utilisateur, u.nom, u.prenom, r.nom as role

                     FROM utilisateur u

                     JOIN role r ON u.id\_role = r.id\_role

                     ORDER BY u.nom, u.prenom");

while ($row = $stmt->fetch()) {

  $membres[$row['id\_utilisateur']] = $row['nom'] . ' ' . $row['prenom'] . ' (' . $row['role'] . ')';

}

* Ce code effectue une **jointure** entre **utilisateur** et **rôle** pour afficher des infos complètes.
* Les résultats sont **triés** par nom, facilitant la navigation dans la liste déroulante.

### 3.3.b - Suppression Sécurisée avec Transaction

Lorsqu’un utilisateur est sélectionné, le code supprime d’abord son badge puis l’utilisateur ensuite. Le tout est effectué dans une transaction pour éviter toute incohérence.

try {

    $pdo->beginTransaction();

    $stmt = $pdo->prepare("DELETE FROM utilisateur WHERE id\_utilisateur = ?");

    $stmt->execute([$id\_utilisateur]);

    $pdo->commit();

    echo "<p style='color: green;'>Membre supprimé avec succès.</p>";

catch (Exception $e) {

    $pdo->rollBack();

    echo "<p style='color: red;'>Erreur lors de la suppression : " . $e->getMessage() . "</p>";

}

* **beginTransaction()** et **commit()** garantissent que **les deux suppressions se font ensemble**, ou pas du tout.
* Cela évite un cas où un badge serait supprimé mais pas l’utilisateur (ou inversement).
* En cas d'erreur, **rollBack()** annule tout.

## 3.4 – Gestion des badges

Une image contenant texte, logiciel, nombre, Icône d’ordinateur

Le contenu généré par l’IA peut être incorrect.

La gestion des badges permet au technicien de gérer les badges d'accès des utilisateurs plus précisément de **consulter, rechercher et désactiver ou réactiver** les badges attribués aux utilisateurs. C’est un outil essentiel pour assurer le **suivi** et la **sécurité** des accès. Cette page affiche **tous les badges** contenus dans la base de données.

Une image contenant texte, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.

Voici la rubrique présentant seulement les badges **actifs**.

Une image contenant capture d’écran, texte, logiciel

Le contenu généré par l’IA peut être incorrect.

Voici la rubrique présentant seulement les badges **inactifs**.

## 4 - Problèmes rencontrés

Pour sécuriser l'accès aux différentes pages du site, nous avons dû modifier l’extension de plusieurs fichiers, initialement en .html, en .php. Cette modification a permis d’ajouter un contrôle d’accès via les sessions PHP.

L’objectif était de s’assurer que seules les personnes authentifiées (Agent ou Technicien) puissent accéder aux interfaces dédiées. Sans ce mécanisme, un utilisateur non connecté pouvait accéder directement aux pages simplement en saisissant leur URL dans le navigateur.

Nous avons donc ajouté, en haut de chaque page sensible, le code suivant permettant de vérifier si une session utilisateur est active :

<?php

session\_start();

// Si l'utilisateur n'est pas connecté, redirige vers la page de connexion

if (!isset($\_SESSION['user\_id'])) {

    header("Location: /Projet\_Acces\_lycee/Interfaces\_Web/Index.html");

    exit();

}

?>

Ce script vérifie que la variable de session user\_id est bien définie. Dans le cas contraire, l'utilisateur est immédiatement redirigé vers la page d'authentification (Index.html), empêchant ainsi tout accès non autorisé.

Nous avons également créé un dossier *Agent* ainsi qu’un dossier *Technicien* dans notre répertoire, dans lesquels nous avons placé toutes les pages correspondant à chaque rôle. Par la suite, nous avons créé un fichier .htaccess permettant de bloquer l’accès direct à ces deux dossiers.

**Code source fichier .htaccess :**

# Interdire l'affichage du contenu des répertoires

Options -Indexes

# Activer le module de réécriture

RewriteEngine On

# Bloquer l'accès direct au dossier Agent

RewriteRule ^(.)$ /Interfaces\_Web/Index.html[L,R=301]

## 5 – Bilan

Ce projet m’a permis de mettre en pratique de nombreuses compétences acquises au cours de ma formation, tant sur le plan technique que méthodologique.

Sur le plan technique, j’ai renforcé mes connaissances en :

* Développement web (HTML, CSS, PHP),
* Gestion de base de données avec MySQL,
* Utilisation de sessions PHP pour la gestion sécurisée des utilisateurs,
* Manipulation de périphériques réseau comme les lecteurs RFID et caméras IP.

J’ai également compris l’importance de la sécurité, en particulier sur :

* La protection des interfaces via des vérifications de session ($\_SESSION),
* Le chiffrement des mots de passe via password\_hash(),
* La protection des répertoires sensibles avec .htaccess.

Sur le plan organisationnel, j’ai appris à :

* Travailler en équipe en répartissant les tâches de manière cohérente,
* Structurer mon code de façon claire et réutilisable,
* Rédiger un rapport technique structuré et documenté.

Les différentes difficultés rencontrées, comme la gestion des accès selon les rôles ou l’intégration de la caméra IP, ont été des opportunités d’apprentissage que j’ai su surmonter.

Enfin, ce projet m’a permis de mieux comprendre le fonctionnement global d’un système de contrôle d’accès physique, et de concevoir une interface claire, fonctionnelle et adaptée aux différents utilisateurs (agent et technicien).

Je suis satisfait du résultat final, et je considère ce projet comme une expérience valorisable pour mes futures démarches professionnelles.

## 

# ANNEXES

Code source Index.html :

<!DOCTYPE html>

<html lang="fr">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Authentification AC-L</title>

    <link rel="stylesheet" href="connexion.css">

</head>

<body>

     <div class="container">

        <div class="wrapper">

            <section class="login">

                <h1>Connexion</h1>

                <form method="POST" action="traitement\_connexion.php">

                    <div class="inputbox">

                        <ion-icon name="mail"></ion-icon>

                        <input type="email" name="email" required>

                        <label>Email</label>

                    </div>

                    <div class="inputbox">

                        <ion-icon name="lock"></ion-icon>

                        <input type="password" name="password" required>

                        <label>Mot de passe</label>

                    </div>

                    <button type="submit">Connexion</button>

                </form>

            </section>

            <script src="https://unpkg.com/ionicons@4.5.10-0/dist/ionicons.js"></script>

        </div>

     </div>

</body>

</html>

Code source traitement\_connexion.php :

<?php

session\_start();

require 'config.php';

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

    $email = trim($\_POST['email']);

    $password = trim($\_POST['password']);

    try {

        $pdo = new PDO(DB\_DSN, DB\_USER, DB\_PASS, $pdo\_options);

        $sql = $pdo->prepare("SELECT id\_utilisateur\_web, login, mot\_de\_passe, role FROM utilisateur\_web WHERE login = ?");

        $sql->execute([$email]);

        $user = $sql->fetch(PDO::FETCH\_ASSOC);

        if ($user) {

            if (password\_verify($password, $user['mot\_de\_passe'])) {

                $\_SESSION['user\_id'] = $user['id\_utilisateur\_web'];

                $\_SESSION['user\_email'] = $user['login'];

                $\_SESSION['user\_role'] = $user['role'];

                if ($user['role'] == 'Agent') {

                    header("Location: Agent/espace\_agent.php");

                } elseif ($user['role'] == 'Technicien') {

                    header("Location: Technicien/espace\_technicien.php");

                }

                exit();

            } else {

                header("Location: Index.html?error=motdepasse");

                exit();

            }

        } else {

            header("Location: Index.html?error=utilisateur");

            exit();

        }

    } catch (PDOException $e) {

        header("Location: Index.html?error=serveur");

        exit();

    }

} else {

    header("Location: Index.html");

    exit();

}

?>

**Espace Agent d’accueil**

Code source espace\_agent.php :

<?php

session\_start();

// Vérifier si l'utilisateur est connecté

if (!isset($\_SESSION['user\_id']) || !isset($\_SESSION['user\_role'])) {

    // Rediriger vers la page de connexion

    header("Location: /Projet\_Acces\_lycee/Interfaces\_Web/Index.html?error=acces\_non\_autorise");

    exit();

}

// Vérifier que l'utilisateur a le rôle Agent

if ($\_SESSION['user\_role'] != 'Agent') {

    header("Location: ../Index.html?error=acces\_non\_autorise");

    exit();

}

?>

<!DOCTYPE html>

<html lang="fr">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Espace Agent</title>

    <link rel="stylesheet" href="espace\_agent.css">

</head>

<body>

    <div class="container">

        <div class="wrapper">

            <h1>Espace Agent</h1>

            <ul>

            <li>

                <p><a href="Camera.php">Caméra IP/Gestion d'accès</a></p>

            </li>

            <li>

                <p><a href="HistoriqueAgent.php">Historique des passages</a></p>

            </li>

            </ul>

            <a href="deconnexion.php" class="button-danger">Se déconnecter</a>

        </div>

    </div>

</body>

</html>

Code source Camera.php :

<?php

session\_start();

if (!isset($\_SESSION['user\_id'])) {

    header("Location: /Projet\_Acces\_lycee/Interfaces\_Web/Index.html");

    exit();

}

?>

<!DOCTYPE html>

<html lang="fr">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Gestion caméra et Barionet</title>

    <link rel="stylesheet" href="Camera.css">

    <script>

        function getBarionetIP() {

            return document.getElementById('ipAddress').value;

        }

        function controlOutput(output, value) {

            const ip = getBarionetIP();

            const url = "http://" + ip + "/rc.cgi?o=" + output + "," + value;

            fetch(url)

                .then(response => {

                    if (response.ok) {

                        showStatus(`Commande envoyée : Sortie ${output} - Valeur ${value}`, 'success');

                    } else {

                        showStatus('Erreur lors de l\'envoi de la commande', 'error');

                    }

                })

                .catch(error => {

                    console.error('Erreur:', error);

                });

        }

        function showStatus(message, type) {

            const statusDiv = document.getElementById('statusMessage');

            statusDiv.textContent = message;

            statusDiv.className = `status ${type}`;

        }

    </script>

</head>

<body>

    <h1>Interface de gestion caméra et ouverture de la porte</h1>

    <div class="main-container">

        <!-- Contrôles caméra -->

        <div class="controls">

            <h2>Contrôle de la caméra</h2>

            <table class="camera-table">

            <tr>

      <td><form action="http://172.22.20.29/pantiltcontrol.cgi" method="post">

                  <input type="hidden" name="PanSingleMoveDegree" value="5">

                   <input type="hidden" name="TiltSingleMoveDegree" value="5">

                  <input type="hidden" name="PanTiltSingleMove" value="0">

                   <input type="submit" value="↖">

          </form></td>

      <td><form action="http://172.22.20.29/pantiltcontrol.cgi" method="post">

                  <input type="hidden" name="PanSingleMoveDegree" value="5">

                   <input type="hidden" name="TiltSingleMoveDegree" value="5">

                  <input type="hidden" name="PanTiltSingleMove" value="1">

                   <input type="submit" value="↑">

          </form></td>

<td><form action="http://172.22.20.29/pantiltcontrol.cgi" method="post">

                  <input type="hidden" name="PanSingleMoveDegree" value="5">

                  <input type="hidden" name="TiltSingleMoveDegree" value="5">

                  <input type="hidden" name="PanTiltSingleMove" value="2">

                   <input type="submit" value="↗">

          </form></td>

                </tr>

                <tr>

      <td><form action="http://172.22.20.29/pantiltcontrol.cgi" method="post">

                  <input type="hidden" name="PanSingleMoveDegree" value="5">

                  <input type="hidden" name="TiltSingleMoveDegree" value="5">

                   <input type="hidden" name="PanTiltSingleMove" value="3">

                  <input type="submit" value="←">

          </form></td>

<td><form action="http://172.22.20.29/pantiltcontrol.cgi" method="post">

                   <input type="hidden" name="PanSingleMoveDegree" value="5">

                  <input type="hidden" name="TiltSingleMoveDegree" value="5">

                   <input type="hidden" name="PanTiltSingleMove" value="4">

                  <input type="submit" class="reset-btn" value="⭯">

          </form></td>

      <td><form action="http://172.22.20.29/pantiltcontrol.cgi" method="post">

                   <input type="hidden" name="PanSingleMoveDegree" value="5">

                   <input type="hidden" name="TiltSingleMoveDegree" value="5">

                   <input type="hidden" name="PanTiltSingleMove" value="5">

                  <input type="submit" value="→">

          </form></td>

                </tr>

                <tr>

      <td><form action="http://172.22.20.29/pantiltcontrol.cgi" method="post">

                   <input type="hidden" name="PanSingleMoveDegree" value="5">

                   <input type="hidden" name="TiltSingleMoveDegree" value="5">

                   <input type="hidden" name="PanTiltSingleMove" value="6">

                   <input type="submit" value="↙">

          </form></td>

      <td><form action="http://172.22.20.29/pantiltcontrol.cgi" method="post">

                   <input type="hidden" name="PanSingleMoveDegree" value="5">

                  <input type="hidden" name="TiltSingleMoveDegree" value="5">

                   <input type="hidden" name="PanTiltSingleMove" value="7">

                  <input type="submit" value="↓">

          </form></td>

      <td><form action="http://172.22.20.29/pantiltcontrol.cgi" method="post">

                   <input type="hidden" name="PanSingleMoveDegree" value="5">

                  <input type="hidden" name="TiltSingleMoveDegree" value="5">

                  <input type="hidden" name="PanTiltSingleMove" value="8">

                  <input type="submit" value="↘">

          </form></td>

                </tr>

            </table>

        </div>

        <!-- Flux vidéo -->

        <div class="video-feed">

            <img src="http://172.22.20.29/video/mjpg.cgi" alt="Flux vidéo caméra D-Link">

        </div>

    </div>

    <!-- Contrôles Barionet -->

    <div class="barionet-container">

        <h2>Gestion d'accès</h2>

        <div class="config-section">

            <div class="input-group hidden">

                <label for="ipAddress">Adresse IP:</label>

                <input type="text" id="ipAddress" value="172.22.21.164">

            </div>

        </div>

        <div class="controls">

            <div class="output-group">

                <h3>Ouverture de la porte</h3>

                <button onclick="controlOutput(1, 10)" class="btn">OUVRIR</button>

            </div>

            <div class="output-group">

                <h3>Modes de fonctionnement</h3>

     <button onclick="controlOutput(2, 0)" class="btn">Badge ou Agent</button>

     <button onclick="controlOutput(2, 1)" class="btn">Mode Auto</button>

            </div>

        </div>

        <div id="statusMessage" class="status"></div>

        </div>

    </div>

    <div class="return-button">

            <a href="espace\_agent.php">Retour à la page d'accueil</a>

    </div>

</body>

</html>

Code source HistoriqueAgent.php :

<?php

session\_start();

// Redirection si non connecté

if (!isset($\_SESSION['user\_id'])) {

    header("Location: /Projet\_Acces\_lycee/Interfaces\_Web/Index.html");

    exit();

}

try {

    $pdo = new PDO(DB\_DSN, DB\_USER, DB\_PASS, $pdo\_options);

} catch (PDOException $e) {

    die("<p style='color: red;'>Erreur de connexion : " . $e->getMessage() . "</p>");

}

// Récupération des paramètres de filtrage

$recherche = isset($\_GET['recherche']) ? trim($\_GET['recherche']) : '';

$date\_filtre = isset($\_GET['date\_filtre']) ? $\_GET['date\_filtre'] : '';

$heure\_filtre = isset($\_GET['heure\_filtre']) ? $\_GET['heure\_filtre'] : '';

$limit = isset($\_GET['limit']) ? (int)$\_GET['limit'] : 50;  // Par défaut, limite à 50 passages

// Construction de la condition WHERE

$conditions = [];

$binds = [];

if (!empty($recherche)) {

    $conditions[] = "(u.nom LIKE :recherche

                   OR u.prenom LIKE :recherche

                   OR b.code\_badge LIKE :recherche

                   OR r.nom LIKE :recherche)";

    $binds[':recherche'] = "%$recherche%";

    if (strtolower($recherche) === 'accepté') {

        $conditions[] = "h.statut = 1";

    } elseif (strtolower($recherche) === 'refusé') {

        $conditions[] = "h.statut = 0";

    }

}

if (!empty($date\_filtre)) {

    $conditions[] = "h.date = :date\_filtre";

    $binds[':date\_filtre'] = $date\_filtre;

}

if (!empty($heure\_filtre)) {

    // Modification : affiche tous les passages avant ou à l'heure spécifiée

    $conditions[] = "h.heure <= :heure\_filtre";

    $binds[':heure\_filtre'] = $heure\_filtre;

}

// Construction de la clause WHERE

$where\_clause = '';

if (!empty($conditions)) {

    $where\_clause = "WHERE " . implode(' AND ', $conditions);

}

// Requête pour compter le nombre total d'enregistrements (sans limite)

$query\_count = "SELECT COUNT(\*)

                FROM historique h

                JOIN badge b ON h.id\_badge = b.id\_badge

                JOIN utilisateur u ON b.id\_utilisateur = u.id\_utilisateur

                JOIN role r ON u.id\_role = r.id\_role

                $where\_clause";

$stmt\_count = $pdo->prepare($query\_count);

foreach ($binds as $key => $value) {

    $stmt\_count->bindValue($key, $value, PDO::PARAM\_STR);

}

$stmt\_count->execute();

$total\_count = $stmt\_count->fetchColumn();

// Requête pour récupérer les données avec la limite

$query = "SELECT h.id\_historique, u.nom, u.prenom, b.code\_badge, h.date, h.heure, h.statut, r.nom AS nom\_role

          FROM historique h

          JOIN badge b ON h.id\_badge = b.id\_badge

          JOIN utilisateur u ON b.id\_utilisateur = u.id\_utilisateur

          JOIN role r ON u.id\_role = r.id\_role

          $where\_clause

          ORDER BY h.date DESC, h.heure DESC

          LIMIT :limit";

$stmt = $pdo->prepare($query);

foreach ($binds as $key => $value) {

    $stmt->bindValue($key, $value, PDO::PARAM\_STR);

}

$stmt->bindValue(':limit', $limit, PDO::PARAM\_INT);

$stmt->execute();

$historique = $stmt->fetchAll(PDO::FETCH\_ASSOC);

// Nombre d'enregistrements affichés actuellement

$total\_affiches = count($historique);

?>

<!DOCTYPE html>

<html lang="fr">

<head>

    <meta charset="UTF-8">

    <title>Historique des passages</title>

    <link rel="stylesheet" href="historique.css">

    <style>

        .filter-container {

            display: flex;

            flex-wrap: wrap;

            gap: 10px;

            margin-bottom: 20px;

            align-items: flex-end;

        }

        .filter-group {

            display: flex;

            flex-direction: column;

        }

        .btn-group {

            display: flex;

            gap: 10px;

            align-items: center;

        }

        .limit-selector {

            margin-left: auto;

        }

    </style>

</head>

<body>

    <h1>Historique des passages</h1>

    <div class="filter-container">

        <form method="get" action="" class="filter-form">

            <div class="filter-group">

                <label for="recherche">Recherche :</label>

                <input type="text" id="recherche" name="recherche" value="<?= htmlspecialchars($recherche); ?>"

                       placeholder="Nom, prénom, badge...">

            </div>

            <div class="filter-group">

                <label for="date\_filtre">Date :</label>

                <input type="date" id="date\_filtre" name="date\_filtre" value="<?= htmlspecialchars($date\_filtre); ?>">

            </div>

            <div class="filter-group">

                <label for="heure\_filtre">Heure (jusqu'à) :</label>

                <input type="time" id="heure\_filtre" name="heure\_filtre" value="<?= htmlspecialchars($heure\_filtre); ?>">

            </div>

            <div class="filter-group limit-selector">

                <label for="limit">Nombre de résultats :</label>

                <select id="limit" name="limit">

    <option value="50" <?= $limit == 50 ? 'selected' : '' ?>>50</option>

    <option value="100" <?= $limit == 100 ? 'selected' : '' ?>>100</option>

    <option value="200" <?= $limit == 200 ? 'selected' : '' ?>>200</option>

    <option value="500" <?= $limit == 500 ? 'selected' : '' ?>>500</option>

    <option value="1000" <?= $limit == 1000 ? 'selected' : '' ?>>1000</option>

                </select>

            </div>

            <div class="btn-group">

                <button type="submit" class="btn btn-primary">Filtrer</button>

                <a href="HistoriqueAgent.php" class="btn btn-secondary">Réinitialiser</a>

            </div>

        </form>

    </div>

    <?php if (empty($historique)): ?>

        <p style="text-align: center;">Aucun passage enregistré correspondant aux critères.</p>

    <?php else: ?>

        <div class="table-container">

            <table>

                <thead>

                    <tr>

                        <th>Nom</th>

                        <th>Prénom</th>

                        <th>Rôle</th>

                        <th>Numéro de badge</th>

                        <th>Date</th>

                        <th>Heure</th>

                        <th>Statut</th>

                    </tr>

                </thead>

                <tbody>

                    <?php foreach ($historique as $row): ?>

                        <tr>

                            <td><?= htmlspecialchars($row['nom']) ?></td>

                            <td><?= htmlspecialchars($row['prenom']) ?></td>

                            <td><?= htmlspecialchars($row['nom\_role']) ?></td>

                            <td><?= htmlspecialchars($row['code\_badge']) ?></td>

                            <td><?= date('d/m/Y', strtotime($row['date'])) ?></td>

                            <td><?= htmlspecialchars($row['heure']) ?></td>

                            <td class="<?= intval($row['statut']) === 1 ? 'statut-accepte' : 'statut-refuse' ?>">

                                <?= intval($row['statut']) === 1 ? 'Accepté' : 'Refusé' ?>

                            </td>

                        </tr>

                    <?php endforeach; ?>

                </tbody>

            </table>

        </div>

    <?php endif; ?>

    <footer>

        <p>Affichage: <?= $total\_affiches ?> sur <?= $total\_count ?> enregistrement(s) trouvé(s)</p>

        <a href="espace\_agent.php" class="retour-btn">Retour à la page d'accueil</a>

    </footer>

</body>

</html>

**Espace Technicien**

Code source espace\_technicien.php :

<?php

session\_start();

// Si l'utilisateur n'est pas connecté, redirige vers la page de connexion

if (!isset($\_SESSION['user\_id'])) {

    header("Location: /Projet\_Acces\_lycee/Interfaces\_Web/Index.html");

    exit();

}

?>

<!DOCTYPE html>

<html lang="fr">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Espace Technicien</title>

    <link rel="stylesheet" href="espace\_technicien.css">

</head>

<body>

    <div class="container">

        <div class="wrapper">

            <h1>Espace Technicien</h1>

            <ul>

            <li>

                <p><a href="HistoriqueTechnicien.php">Historique des passages</a></p>

            </li>

            <li>

                <p><a href="AjouterMembre.php">Ajouter un membre</a></p>

            </li>

            <li>

                <p><a href="SupprimerMembre.php">Supprimer un membre</a></p>

            </li>

            <li>

                <p><a href="GestionBadges.php">Gestion des badges</a></p>

            </li>

            </ul>

            <a href="deconnexion.php" class="button-danger">Se déconnecter</a>

        </div>

    </div>

</body>

</html>

Code source HistoriqueTechnicien.php :

<?php

session\_start();

require '../config.php';

// Redirection si non connecté

if (!isset($\_SESSION['user\_id'])) {

    header("Location: /Projet\_Acces\_lycee/Interfaces\_Web/Index.html");

    exit();

}

try {

    $pdo = new PDO(DB\_DSN, DB\_USER, DB\_PASS, $pdo\_options);

} catch (PDOException $e) {

    die("<p style='color: red;'>Erreur de connexion : " . $e->getMessage() . "</p>");

}

// Récupération des paramètres de filtrage

$recherche = isset($\_GET['recherche']) ? trim($\_GET['recherche']) : '';

$date\_filtre = isset($\_GET['date\_filtre']) ? $\_GET['date\_filtre'] : '';

$heure\_filtre = isset($\_GET['heure\_filtre']) ? $\_GET['heure\_filtre'] : '';

$limit = isset($\_GET['limit']) ? (int)$\_GET['limit'] : 50;  // Par défaut, limite à 50 passages

// Construction de la condition WHERE

$conditions = [];

$binds = [];

if (!empty($recherche)) {

    $conditions[] = "(u.nom LIKE :recherche

                   OR u.prenom LIKE :recherche

                   OR b.code\_badge LIKE :recherche

                   OR r.nom LIKE :recherche)";

    $binds[':recherche'] = "%$recherche%";

    if (strtolower($recherche) === 'accepté') {

        $conditions[] = "h.statut = 1";

    } elseif (strtolower($recherche) === 'refusé') {

        $conditions[] = "h.statut = 0";

    }

}

if (!empty($date\_filtre)) {

    $conditions[] = "h.date = :date\_filtre";

    $binds[':date\_filtre'] = $date\_filtre;

}

if (!empty($heure\_filtre)) {

    // Modification : affiche tous les passages avant ou à l'heure spécifiée

    $conditions[] = "h.heure <= :heure\_filtre";

    $binds[':heure\_filtre'] = $heure\_filtre;

}

// Construction de la clause WHERE

$where\_clause = '';

if (!empty($conditions)) {

    $where\_clause = "WHERE " . implode(' AND ', $conditions);

}

// Requête pour compter le nombre total d'enregistrements (sans limite)

$query\_count = "SELECT COUNT(\*)

                FROM historique h

                JOIN badge b ON h.id\_badge = b.id\_badge

                JOIN utilisateur u ON b.id\_utilisateur = u.id\_utilisateur

                JOIN role r ON u.id\_role = r.id\_role

                $where\_clause";

$stmt\_count = $pdo->prepare($query\_count);

foreach ($binds as $key => $value) {

    $stmt\_count->bindValue($key, $value, PDO::PARAM\_STR);

}

$stmt\_count->execute();

$total\_count = $stmt\_count->fetchColumn();

// Requête pour récupérer les données avec la limite

$query = "SELECT h.id\_historique, u.nom, u.prenom, b.code\_badge, h.date, h.heure, h.statut, r.nom AS nom\_role

          FROM historique h

          JOIN badge b ON h.id\_badge = b.id\_badge

          JOIN utilisateur u ON b.id\_utilisateur = u.id\_utilisateur

          JOIN role r ON u.id\_role = r.id\_role

          $where\_clause

          ORDER BY h.date DESC, h.heure DESC

          LIMIT :limit";

$stmt = $pdo->prepare($query);

foreach ($binds as $key => $value) {

    $stmt->bindValue($key, $value, PDO::PARAM\_STR);

}

$stmt->bindValue(':limit', $limit, PDO::PARAM\_INT);

$stmt->execute();

$historique = $stmt->fetchAll(PDO::FETCH\_ASSOC);

// Nombre d'enregistrements affichés actuellement

$total\_affiches = count($historique);

?>

<!DOCTYPE html>

<html lang="fr">

<head>

    <meta charset="UTF-8">

    <title>Historique des passages</title>

    <link rel="stylesheet" href="historique.css">

    <style>

        .filter-container {

            display: flex;

            flex-wrap: wrap;

            gap: 10px;

            margin-bottom: 20px;

            align-items: flex-end;

        }

        .filter-group {

            display: flex;

            flex-direction: column;

        }

        .btn-group {

            display: flex;

            gap: 10px;

            align-items: center;

        }

        .limit-selector {

            margin-left: auto;

        }

    </style>

</head>

<body>

    <h1>Historique des passages</h1>

    <div class="filter-container">

        <form method="get" action="" class="filter-form">

            <div class="filter-group">

                <label for="recherche">Recherche :</label>

                <input type="text" id="recherche" name="recherche" value="<?= htmlspecialchars($recherche); ?>"

                       placeholder="Nom, prénom, badge...">

            </div>

            <div class="filter-group">

                <label for="date\_filtre">Date :</label>

                <input type="date" id="date\_filtre" name="date\_filtre" value="<?= htmlspecialchars($date\_filtre); ?>">

            </div>

            <div class="filter-group">

                <label for="heure\_filtre">Heure (jusqu'à) :</label>

                <input type="time" id="heure\_filtre" name="heure\_filtre" value="<?= htmlspecialchars($heure\_filtre); ?>">

            </div>

            <div class="filter-group limit-selector">

                <label for="limit">Nombre de résultats :</label>

                <select id="limit" name="limit">

    <option value="50" <?= $limit == 50 ? 'selected' : '' ?>>50</option>

    <option value="100" <?= $limit == 100 ? 'selected' : '' ?>>100</option>

    <option value="200" <?= $limit == 200 ? 'selected' : '' ?>>200</option>

    <option value="500" <?= $limit == 500 ? 'selected' : '' ?>>500</option>

    <option value="1000" <?= $limit == 1000 ? 'selected' : '' ?>>1000</option>

                </select>

            </div>

            <div class="btn-group">

                <button type="submit" class="btn btn-primary">Filtrer</button>

                <a href="HistoriqueTechnicien.php" class="btn btn-secondary">Réinitialiser</a>

            </div>

        </form>

    </div>

    <?php if (empty($historique)): ?>

        <p style="text-align: center;">Aucun passage enregistré correspondant aux critères.</p>

    <?php else: ?>

        <div class="table-container">

            <table>

                <thead>

                    <tr>

                        <th>Nom</th>

                        <th>Prénom</th>

                        <th>Rôle</th>

                        <th>Numéro de badge</th>

                        <th>Date</th>

                        <th>Heure</th>

                        <th>Statut</th>

                    </tr>

                </thead>

                <tbody>

                    <?php foreach ($historique as $row): ?>

                        <tr>

                            <td><?= htmlspecialchars($row['nom']) ?></td>

                            <td><?= htmlspecialchars($row['prenom']) ?></td>

                            <td><?= htmlspecialchars($row['nom\_role']) ?></td>

                            <td><?= htmlspecialchars($row['code\_badge']) ?></td>

                            <td><?= date('d/m/Y', strtotime($row['date'])) ?></td>

                            <td><?= htmlspecialchars($row['heure']) ?></td>

                            <td class="<?= intval($row['statut']) === 1 ? 'statut-accepte' : 'statut-refuse' ?>">

                                <?= intval($row['statut']) === 1 ? 'Accepté' : 'Refusé' ?>

                            </td>

                        </tr>

                    <?php endforeach; ?>

                </tbody>

            </table>

        </div>

    <?php endif; ?>

    <footer>

        <p>Affichage: <?= $total\_affiches ?> sur <?= $total\_count ?> enregistrement(s) trouvé(s)</p>

        <a href="espace\_technicien.php" class="retour-btn">Retour à la page d'accueil</a>

    </footer>

</body>

</html>

Code source AjouterMembre.php :

<?php

session\_start();

require '../config.php';

// Si l'utilisateur n'est pas connecté, redirige vers la page de connexion

if (!isset($\_SESSION['user\_id'])) {

    header("Location: /Projet\_Acces\_lycee/Interfaces\_Web/Index.html");

    exit();

}

?>

<!DOCTYPE html>

<html lang="fr">

<head>

    <meta charset="UTF-8">

    <title>Ajouter un membre</title>

    <link rel="stylesheet" href="AjouterMembre.css">

</head>

<body>

    <h2>Ajouter un nouveau membre</h2>

    <?php

    try {

        $pdo = new PDO(DB\_DSN, DB\_USER, DB\_PASS, $pdo\_options);

    } catch (PDOException $e) {

        die("<p>Erreur de connexion : " . $e->getMessage() . "</p>");

    }

    // Récupérer tous les rôles disponibles pour les afficher dans le formulaire

    $roles = [];

    $stmt = $pdo->query("SELECT id\_role, nom FROM role"); // Utilisation de "nom" au lieu de "nom\_role"

    while ($row = $stmt->fetch(PDO::FETCH\_ASSOC)) {

        $roles[$row['id\_role']] = $row['nom'];

    }

    // Vérifier si le formulaire a été soumis

    if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

        $nom = trim($\_POST['nom']);

        $prenom = trim($\_POST['prenom']);

        $code\_badge = trim($\_POST['code\_badge']);

        $nom\_role = trim($\_POST['nom\_role']); // Récupération du nom du rôle

        // Vérifier si tous les champs sont remplis

        if (!empty($nom) && !empty($prenom) && !empty($code\_badge) && !empty($nom\_role)) {

            // Vérifier si le code du badge existe déjà

            $stmt = $pdo->prepare("SELECT code\_badge FROM badge WHERE code\_badge = ?");

            $stmt->execute([$code\_badge]);

            $badge\_existant = $stmt->fetch(PDO::FETCH\_ASSOC);

            if ($badge\_existant) {

                echo "<p style='color: red;'>Erreur : Ce code de badge existe déjà dans la base de données.</p>";

            } else {

                // Récupérer l'ID du rôle sélectionné

                $stmt = $pdo->prepare("SELECT id\_role FROM role WHERE nom = ?");

                $stmt->execute([$nom\_role]);

                $role = $stmt->fetch(PDO::FETCH\_ASSOC);

                if (!$role) {

                    die("<p>Erreur : Le rôle spécifié n'existe pas.</p>");

                }

                $id\_role = $role['id\_role'];

                // Insérer l'utilisateur

                $stmt = $pdo->prepare("INSERT INTO utilisateur (nom, prenom, id\_role) VALUES (?, ?, ?)");

                if ($stmt->execute([$nom, $prenom, $id\_role])) {

                    $id\_utilisateur = $pdo->lastInsertId();

                    // Insérer le badge avec l'id\_utilisateur et initialiser l'état à 1 (actif)

                    $stmt = $pdo->prepare("INSERT INTO badge (code\_badge, id\_utilisateur, etat) VALUES (?, ?, 1)");

                    if ($stmt->execute([$code\_badge, $id\_utilisateur])) {

                        echo "<p style='color: green;'>Utilisateur et badge ajoutés avec succès.</p>";

                    } else {

                        echo "<p style='color: red;'>Erreur lors de l'ajout du badge.</p>";

                    }

                } else {

                    echo "<p style='color: red;'>Erreur lors de l'ajout de l'utilisateur.</p>";

                }

            }

        } else {

            echo "<p style='color: red;'>Veuillez remplir tous les champs.</p>";

        }

    }

    ?>

    <!-- Formulaire pour ajouter un membre -->

    <form method="post">

        <label for="nom">Nom :</label>

        <input type="text" id="nom" name="nom" required><br><br>

        <label for="prenom">Prénom :</label>

        <input type="text" id="prenom" name="prenom" required><br><br>

        <label for="code\_badge">Code du badge :</label>

        <input type="text" id="code\_badge" name="code\_badge" required><br><br>

        <label for="nom\_role">Rôle :</label>

        <select id="nom\_role" name="nom\_role" required>

            <option value="">-- Sélectionner un rôle --</option>

            <?php

            foreach ($roles as $id => $nom) {

                echo "<option value=\"$nom\">$nom</option>";

            }

            ?>

        </select><br><br>

        <input type="submit" value="Ajouter" onclick="return confirm('Êtes-vous sûr de vouloir ajouter ce membre ?')">

    </form>

    <p><a href="espace\_technicien.php">Retour à la page d'accueil</a></p>

</body>

</html>

Code source SupprimerMembre.php :

<?php

session\_start();

require '../config.php'; // adapte le chemin si besoin

// Si l'utilisateur n'est pas connecté, redirige vers la page de connexion

if (!isset($\_SESSION['user\_id'])) {

    header("Location: /Projet\_Acces\_lycee/Interfaces\_Web/Index.html");

    exit();

}

?>

<!DOCTYPE html>

<html lang="fr">

<head>

    <meta charset="UTF-8">

    <title>Supprimer un membre</title>

    <link rel="stylesheet" href="SupprimerMembre.css">

</head>

<body>

    <h2>Supprimer un membre</h2>

    <?php

    // Connexion à la base de données avec PDO via les constantes du config.php

    try {

        $pdo = new PDO(DB\_DSN, DB\_USER, DB\_PASS, $pdo\_options);

    } catch (PDOException $e) {

        die("<p>Erreur de connexion : " . $e->getMessage() . "</p>");

    }

    // Récupérer tous les membres disponibles

    $membres = [];

    $stmt = $pdo->query("SELECT u.id\_utilisateur, u.nom, u.prenom, r.nom as role

                         FROM utilisateur u

                         JOIN role r ON u.id\_role = r.id\_role

                         ORDER BY u.nom, u.prenom");

    while ($row = $stmt->fetch()) {

        $membres[$row['id\_utilisateur']] = $row['nom'] . ' ' . $row['prenom'] . ' (' . $row['role'] . ')';

    }

    // Traitement du formulaire

    if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['id\_utilisateur'])) {

        $id\_utilisateur = $\_POST['id\_utilisateur'];

        try {

            $pdo->beginTransaction();

            $stmt = $pdo->prepare("DELETE FROM utilisateur WHERE id\_utilisateur = ?");

            $stmt->execute([$id\_utilisateur]);

            $pdo->commit();

            echo "<p style='color: green;'>Membre supprimé avec succès.</p>";

            // Rafraîchir la liste des membres

            $stmt = $pdo->query("SELECT u.id\_utilisateur, u.nom, u.prenom, r.nom as role

                                FROM utilisateur u

                                JOIN role r ON u.id\_role = r.id\_role

                                ORDER BY u.nom, u.prenom");

            $membres = [];

            while ($row = $stmt->fetch()) {

                $membres[$row['id\_utilisateur']] = $row['nom'] . ' ' . $row['prenom'] . ' (' . $row['role'] . ')';

            }

        } catch (Exception $e) {

            $pdo->rollBack();

            echo "<p style='color: red;'>Erreur lors de la suppression : " . $e->getMessage() . "</p>";

        }

    }

    ?>

    <!-- Formulaire -->

    <form method="post">

        <label for="id\_utilisateur">Sélectionner un membre à supprimer :</label>

        <select id="id\_utilisateur" name="id\_utilisateur" required>

            <option value="">-- Sélectionner un membre --</option>

            <?php

            foreach ($membres as $id => $description) {

                echo "<option value=\"$id\">$description</option>";

            }

            ?>

        </select><br><br>

        <input type="submit" value="Supprimer" onclick="return confirm('Êtes-vous sûr de vouloir supprimer ce membre ?');">

    </form>

    <p><a href="espace\_technicien.php">Retour à la page d'accueil</a></p>

</body>

</html>

Code source GestionBadges.php :

<?php

session\_start();

require '../config.php';

if (!isset($\_SESSION['user\_id'])) {

    header("Location: /Projet\_Acces\_lycee/Interfaces\_Web/Index.html");

    exit();

}

try {

    $pdo = new PDO(DB\_DSN, DB\_USER, DB\_PASS, $pdo\_options);

} catch (PDOException $e) {

    die("<p>Erreur de connexion : " . $e->getMessage() . "</p>");

}

if (isset($\_POST['desactiver\_badge'])) {

    $id\_badge = $\_POST['id\_badge'];

    try {

        $stmt = $pdo->prepare("UPDATE badge SET etat = 0 WHERE id\_badge = ?");

        $message = $stmt->execute([$id\_badge]) ? "Le badge a été désactivé avec succès." : "Erreur lors de la désactivation du badge.";

        $messageType = $stmt->execute([$id\_badge]) ? "success" : "error";

    } catch (PDOException $e) {

        $message = "Erreur: " . $e->getMessage();

        $messageType = "error";

    }

}

if (isset($\_POST['reactiver\_badge'])) {

    $id\_badge = $\_POST['id\_badge'];

    try {

        $stmt = $pdo->prepare("UPDATE badge SET etat = 1 WHERE id\_badge = ?");

        $message = $stmt->execute([$id\_badge]) ? "Le badge a été réactivé avec succès." : "Erreur lors de la réactivation du badge.";

        $messageType = $stmt->execute([$id\_badge]) ? "success" : "error";

    } catch (PDOException $e) {

        $message = "Erreur: " . $e->getMessage();

        $messageType = "error";

    }

}

$filter = isset($\_GET['filter']) ? $\_GET['filter'] : 'actif';

$recherche = isset($\_GET['recherche']) ? $\_GET['recherche'] : '';

$condition\_recherche = '';

$binds = [];

if (!empty($recherche)) {

    $condition\_recherche = " AND (u.nom LIKE :recherche

                             OR u.prenom LIKE :recherche

                             OR b.code\_badge LIKE :recherche

                             OR b.id\_badge LIKE :recherche)";

    $binds[':recherche'] = "%" . $recherche . "%";

}

switch ($filter) {

    case 'actif':

        $whereClause = "WHERE b.etat = 1";

        break;

    case 'inactif':

        $whereClause = "WHERE b.etat = 0";

        break;

    default:

        $whereClause = "WHERE 1=1";

        break;

}

$query = "

    SELECT b.id\_badge, b.code\_badge, b.etat, u.nom, u.prenom

    FROM badge b

    JOIN utilisateur u ON b.id\_utilisateur = u.id\_utilisateur

    $whereClause

    $condition\_recherche

    ORDER BY u.nom, u.prenom

";

$stmt = $pdo->prepare($query);

foreach ($binds as $key => $value) {

    $stmt->bindValue($key, $value, PDO::PARAM\_STR);

}

$stmt->execute();

$badges = $stmt->fetchAll(PDO::FETCH\_ASSOC);

$total\_badges = count($badges);

?>

<!DOCTYPE html>

<html lang="fr">

<head>

    <meta charset="UTF-8">

    <title>Gestion des Badges</title>

    <link rel="stylesheet" href="GestionBadges.css">

</head>

<body>

    <h2>Gestion des Badges</h2>

    <?php if (isset($message)): ?>

        <div class="message <?php echo $messageType; ?>">

            <?php echo $message; ?>

        </div>

    <?php endif; ?>

    <div class="options">

        <form method="get" action="">

            <input type="hidden" name="filter" value="<?php echo htmlspecialchars($filter); ?>">

            <label for="recherche">Rechercher :</label>

            <input type="text" id="recherche" name="recherche" value="<?php echo htmlspecialchars($recherche); ?>"

                   placeholder="Nom, prénom ou code badge...">

            <button type="submit">Rechercher</button>

        </form>

    </div>

    <div class="filter-section">

        <h3>Liste des badges</h3>

        <div class="filter-tabs">

            <a href="?filter=actif<?php echo !empty($recherche) ? '&recherche='.urlencode($recherche) : ''; ?>"

               class="<?php echo $filter == 'actif' ? 'active' : ''; ?>">Badges actifs</a>

            <a href="?filter=inactif<?php echo !empty($recherche) ? '&recherche='.urlencode($recherche) : ''; ?>"

               class="<?php echo $filter == 'inactif' ? 'active' : ''; ?>">Badges inactifs</a>

            <a href="?filter=tous<?php echo !empty($recherche) ? '&recherche='.urlencode($recherche) : ''; ?>"

               class="<?php echo $filter == 'tous' ? 'active' : ''; ?>">Tous les badges</a>

        </div>

    </div>

    <?php if (empty($badges)): ?>

        <p>Aucun badge trouvé correspondant aux critères de recherche.</p>

    <?php else: ?>

        <div class="table-container">

            <table>

                <thead>

                    <tr>

                        <th>Code Badge</th>

                        <th>Nom</th>

                        <th>Prénom</th>

                        <th>État</th>

                        <th>Action</th>

                    </tr>

                </thead>

                <tbody>

                    <?php foreach ($badges as $badge): ?>

                        <tr>

                            <td><?php echo htmlspecialchars($badge['code\_badge']); ?></td>

                            <td><?php echo htmlspecialchars($badge['nom']); ?></td>

                            <td><?php echo htmlspecialchars($badge['prenom']); ?></td>

                            <td class="<?php echo $badge['etat'] == 1 ? 'statut-actif' : 'statut-inactif'; ?>">

                                <?php echo $badge['etat'] == 1 ? 'Actif' : 'Désactivé'; ?>

                            </td>

                            <td>

                                <?php if ($badge['etat'] == 1): ?>

                                    <form method="post" onsubmit="return confirm('Êtes-vous sûr de vouloir désactiver ce badge?');">

                                        <input type="hidden" name="id\_badge" value="<?php echo $badge['id\_badge']; ?>">

                                        <button type="submit" name="desactiver\_badge" class="deactivate">Désactiver (Badge perdu)</button>

                                    </form>

                                <?php else: ?>

                                    <form method="post" onsubmit="return confirm('Êtes-vous sûr de vouloir réactiver ce badge?');">

                                        <input type="hidden" name="id\_badge" value="<?php echo $badge['id\_badge']; ?>">

                                        <button type="submit" name="reactiver\_badge" class="activate">Réactiver le badge</button>

                                    </form>

                                <?php endif; ?>

                            </td>

                        </tr>

                    <?php endforeach; ?>

                </tbody>

            </table>

        </div>

    <?php endif; ?>

    <footer>

        <p>Total: <?php echo $total\_badges; ?> badge(s)</p>

        <p><a href="espace\_technicien.php" class="retour-btn">Retour à la page d'accueil</a></p>

    </footer>

</body>

</html>

Code source config.php :

<?php

// Paramètres de la base de données

$host = 'localhost';

$DB = 'acl';

$port = '3306';

// Définition du Data Source Name (DSN)

$dsn = 'mysql:host=' . $host . ';dbname=' . $DB . ';charset=utf8;port=' . $port;

// Définition des identifiants de connexion

$login = 'acl';

$mdp = 'acllp2i3!';

// Définition des options de PHP Data Objects (PDO)

$pdo\_options = [

    PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION,

    PDO::ATTR\_DEFAULT\_FETCH\_MODE => PDO::FETCH\_ASSOC

];

// Définir les constantes pour la connexion à la base de données

define('DB\_DSN', $dsn);

define('DB\_USER', $login);

define('DB\_PASS', $mdp);

?>

Code source deconnexion.php :

<?php

session\_start();

// Détruire toutes les données de la session

session\_unset();     // Supprime toutes les variables de session

session\_destroy();   // Détruit la session

// Rediriger vers la page de connexion

header("Location: /Projet\_Acces\_lycee/Interfaces\_Web/Index.html");  // Chemin qui redirige vers la page de connexion

exit();

?>