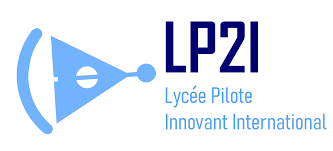
.

1. Table des matières

[Projet ACCÈS LYCÉE 0](#_5agf5dtgxrlw)

[Table des matières 1](#_lcgps1t8x69e)

[Introduction Générale 2](#_k6dq6044qqdw)

[But du projet 4](#_8veo9tmute80)

[Diagramme de Gantt 5](#_5nl3qh29ova6)

[RÉPARTITION DES TACHES 6](#_f8jvqm83al94)

[Analyse UML 7](#_tqh6fnx62o6a)

[BASE DE DONNÉES 12](#_kq5sq0btnsrn)

[PILOTAGE DU BARIONET 16](#_bzcw6ukjrmm4)

[MISE EN OEUVRE DES DIFFÉRENTS LECTEURS DE BADGES 21](#_5a70kqkhagwy)

[**Configuration du lecteur MCR04** 22](#_gyac72rtyi8k)

1. Introduction Générale



Le BTS CIEL est une formation de deux ans qui te prépare à travailler dans le secteur de l’informatique. Un des aspects cruciaux de cette formation est le Projet E6. C'est un travail pratique où tu mets en œuvre les compétences acquises durant notre formation. C'est également l'occasion de faire valoir notre esprit d'analyse et notre sens de l'organisation.

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 du 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 * 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 :  Application :  Application :    Application : |

1. 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.

1. Diagramme de Gantt

Une image contenant ligne, nombre, Tracé, texte

Le contenu généré par l’IA peut être incorrect.

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

Celui ci-dessus est séparé en 4 phases et relate l’avancement de chaque tâche qu’on a ou est 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)

1. RÉPARTITION DES TACHES

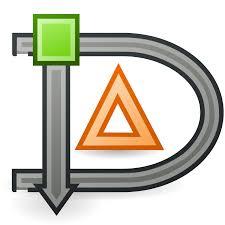
| Etudiant | Tâches |
| --- | --- |
| Akouman DIANGO | * Pilotage du Barionet * Mise en place du lecteur RFID Invéo |
| Elias COSME VINOU | * Conception UML * Mise en place de la base de données * Communication avec le système RFID |
| Romain METAIS | * Développement de l’interface de l’agent d’accueil * Développement de l’interface du technicien * Communication avec le système RFID |

1. 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 crée 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 d’activité
* Diagramme de séquences

Un algorigramme a été conçu afin d’avoir une représentation détaillée de notre projet avec le tout cumulé.

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 noeuds dans un système réparti, les artefacts qui sont stockés sur chaque noeud et les composants et autres éléments que les artefacts implémentent. Les noeuds 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 (en vert, 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 (en bleu 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 (en bleu 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 (en rouge, 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).

On peut 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 d’activité

Une image contenant diagramme, ligne

Le contenu généré par l’IA peut être incorrect.



Le diagramme d’activité ci-dessus décrit le fonctionnement d’un système de contrôle d’accès basé sur la technologie RFID (Radio Frequency Identification). Ce système pourrait, par exemple, être utilisé pour contrôler l’ouverture d’un portail ou d’un volet automatisé.

Concrètement, le processus commence dès qu’une personne présente son badge RFID devant le lecteur. Le système lit alors automatiquement le code inscrit sur le badge. Immédiatement après, une vérification est effectuée dans la base de données afin de savoir si le code du badge correspond à un utilisateur autorisé.

C’est à ce moment que le système prend une décision :

* **Si le badge n’est pas reconnu ou n’est pas autorisé**, le volet reste fermé. L’accès est alors refusé et aucune ouverture n’est déclenchée.
* **Si le badge est bien autorisé**, le volet s’ouvre automatiquement, permettant ainsi l’accès à la personne.

Une fois la décision prise (qu’il s’agisse d’une ouverture ou d’un refus), le système se prépare à retransmettre les événements qui viennent de se produire. Cette retransmission permet, par exemple, d’informer d’autres systèmes ou de déclencher des alertes si besoin.

Enfin, quelle que soit l’action réalisée, toutes les informations concernant la tentative d’accès (réussie ou non) sont enregistrées dans la base de données. Cela garantit la traçabilité de tous les événements : il est donc possible de savoir qui a essayé d’entrer, à quel moment, et si l’accès a été accordé ou non.

Le processus se termine une fois que l’événement a bien été sauvegardé.

En résumé, ce diagramme illustre un système automatisé, fiable et sécurisé qui gère de bout en bout une tentative d’accès par badge RFID. Toutes les opérations sont tracées, ce qui facilite le suivi et renforce la sécurité du dispositif.

1. BASE DE DONNÉES

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

Le contenu généré par l’IA peut être incorrect.

La base de données nous permet de stocker toutes les informations nécessaires à la réussite de ce projet comme la liste des utilisateurs. Notre SGB (système de gestion de base de données) est PhpMyAdmin.

La base de données comporte 5 tables :

* **Historique** : qui permet de relater les différentes entrées en donnant la date, l’heure, le statut (autorisé ou refusé), et le numéro de badge.
* **Badge** : Le badge qui permet d’avoir son identifiant unique (code badge) et si il est actif ou non.
* **Utilisateur** : Liste les utilisateurs avec les attributs nom, prénom chacun auront un rôle défini (principal, cuisinier…)
* **Rôle** : Liste les différents rôles qu’il peut y’avoir au sein d’un établissement dans notre cas lycée avec un plage horaire attribué pour chacun des rôles



La technologique RFID est une méthode pour mémoriser et récupérer des données à distance en utilisant des marqueurs appelés « radio-étiquettes ». Ces radio-étiquettes sont des petits objets qui peuvent être incorporés ou collés, ici on va s’en servir sur un badge (ou carte) RFID. Quand le badge va passer devant le lecteur, celui-ci va envoyer un signal à l’étiquette. L’étiquette retourne alors son identification numérique (Code RFID). Pour que l’utilisateur puisse accéder à l’établissement, il va passer sa carte devant le lecteur, le calculateur embarqué (Barionnet) va récupérer le code RFID et va envoyer ce code au serveur. Le serveur va ensuite interroger la base de données (Base de données (MariaDB). Ainsi on va pouvoir vérifier que l’utilisateur correspondant à ce code RFID existe réellement, et que l’utilisateur essaie d’accéder à la salle dans la bonne plage horaire par rapport à son statut (Etudiant, Personnel).

Le technicien et l’agent d’accueil a la possibilité d’effectuer plusieurs actions depuis le client. Ils sont les seuls à pouvoir s’identifier à l’application grâce à des identifiants stockés en base de données. Il peut effectuer des opérations CRUD (Create, Read, Update, Delete) sur les utilisateurs (étudiants et personnels du lycée) ainsi que sur les plages horaires en fonction du rôle de la personne. Pour finir, ils peuvent consulter l’historique des entrées. L’historique est constitué de toutes les tentatives d’accès aux salles (échouées ou réussies). Un étudiant doit passer sa carte RFID devant le lecteur, si le créneau horaire le permet, il peut accéder à la salle. Le personnel de l’établissement n’a pas de restrictions, lorsqu’il passe sa carte RFID devant le lecteur, la porte s’ouvre automatiquement.

1. PILOTAGE DU BARIONET
2. Présentation du BARIONET100

Le Barionet de Barix est une gamme de contrôleurs d'automatisation programmables et compatibles réseau permettant d'interfacer une grande variété d'appareils et de systèmes avec des réseaux IP pour les applications domotiques et industrielles. Avec Barionet, la plupart des appareils peuvent être connectés en réseau pour la surveillance et le contrôle via un navigateur web, ainsi que d'autres systèmes d'automatisation standardisés, tels que SNMP et Modbus.

Barionet offre une variété d'interfaces matérielles et logicielles standard, ainsi que des entrées et sorties polyvalentes pour les applications de contrôle et de surveillance.

Deux modèles Barionet sont actuellement disponibles : le Barionet 100 (le Barionet original) et le nouveau Barionet 50. Ces deux modèles se distinguent principalement par le nombre d'entrées et de sorties disponibles.

Le Barionet 100 offre :

* 4 sorties numériques
* 4 entrées analogiques ou numériques
* 4 entrées numériques
* 2 sorties relais
* Interface série RS-422/485
* Interface lecteur Wiegand

Une image contenant texte

Le contenu généré par l’IA peut être incorrect.

Pour la configuration du Barionet, j'ai opté pour un pilotage via un site web grâce aux commandes CGI intégrées à mon programme HTML. Cette approche offre plusieurs avantages :

* **Flexibilité d’accès :** L'interface web permet un contrôle depuis n'importe quel poste connecté au réseau
* **Simplicité d’implémentation :** Pas besoin d'installer de logiciel spécifique sur les postes clients
* **Intégration facilitée** avec le système d'information existant du lycée

**Utilisation des commandes CGI**

Le Barionet dispose d'un serveur web intégré qui implémente quatre commandes CGI principales que j'ai exploitées :

* **rc.cgi**: Utilisée pour contrôler les sorties et les relais du Barionet  
   Copier  
  http://<adresse\_IP\_barionet>/rc.cgi?o=<adresse\_E/S>,<valeur>[&L=<page\_réponse>]
* Par exemple, pour activer le relais 1 qui contrôle l'ouverture de la porte:  
  http://192.168.1.100/rc.cgi?o=1,1
* Pour désactiver le relais 1 qui contrôle l'ouverture de la porte:  
  http://192.168.1.100/rc.cgi?o=1,0
* Cette commande active le relais pendant 5 secondes:  
  http://192.168.1.100/rc.cgi?o=1,50  
    
  **Ouverture de la porte via la page web par l’agent d'accueille**
* La communication mise en place est la suivante:
* **Côté agent:** L'interface web est chargée dans le navigateur de l'agent d'accueil (il dispose d’un bouton ouvrir pour laisser passer une personne sans badge).
* **Requêtes HTTP:** Lorsque l'agent clique sur un bouton, le JavaScript génère une requête HTTP vers le Barionet
* **Traitement CGI:** Le serveur web du Barionet reçoit la requête et exécute la commande CGI correspondante
* **Action matérielle:** Le Barionet active ou désactive les sorties physiques en fonction de la commande reçue
* Pour le contrôle de l'accès au lycée, la sortie 1 est configurée pour commander l'ouverture de la porte avec un délai de temporisation de 10 secondes (valeur 100), ce qui correspond au temps nécessaire pour que l'utilisateur puisse passer.

Sachant que la porte est équipée de différents capteurs de présence pour éviter la fermeture automatique de la porte pendant le passage d’un individu.

Lors d'événements spéciaux tels que des réunions parents-professeurs ou des journées portes ouvertes, le système de contrôle d'accès peut être configuré pour offrir une expérience fluide et accueillante aux visiteurs. Pendant ces événements, la porte principale est programmée pour s'ouvrir automatiquement à l'approche d'un individu, sans nécessiter l'utilisation de badges RFID.

Configuration pour les Événements Spéciaux

Activation des Capteurs de Présence :Une image contenant câble, Appareils électroniques, fils électriques, intérieur

Le contenu généré par l’IA peut être incorrect.

*XUK0ARCTL2 Photoelectric sensors XU, XUK, multi, Sn 0...30 m, 24...240VAC/DC, cable 2 m*

Les capteurs de présence Osiris sont activés et configurés pour détecter l'approche des visiteurs. Lorsqu'une présence est détectée, un signal est envoyé au Barionet.

Ouverture Automatique de la Porte :

Le Barionet est programmé pour activer le relais de la porte dès qu'un signal de présence est reçu. La porte s'ouvre automatiquement, permettant ainsi un accès sans entrave pour les visiteurs.

Durée de l'Ouverture :

La porte reste ouverte pendant une durée prédéfinie grâce à un contacteur avec une temporisation (par exemple, 10 secondes), suffisante pour permettre le passage des

visiteurs. Si une présence continue est détectée, la porte reste ouverte plus longtemps.

Retour à la Configuration Normale :

À la fin de l'événement, le système est reconfiguré pour revenir au mode de fonctionnement normal, nécessitant l'utilisation de badges RFID pour l'accès.

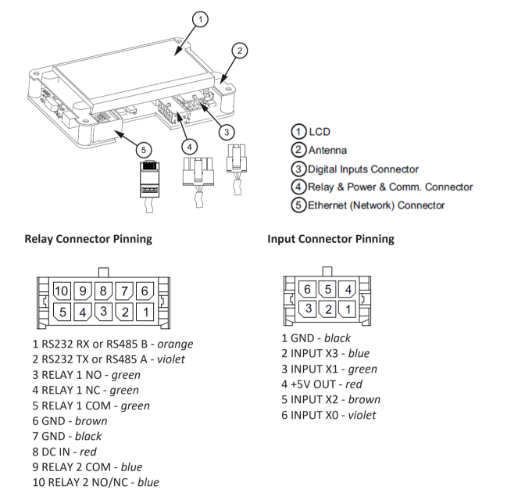
1. [MISE EN OEUVRE DES DIFFÉRENTS LECTEURS DE BADGES](https://docs.google.com/document/d/1bYEm6mHmrN-AchzVYAoI1oXCUlacNjlW0Os6PstJrBw/edit#heading=h.4d34og8)

**Présentation du lecteur MINOVA MCR04**

Pour l'identification ainsi que l'accès au lycée, nous avons opté pour le capteur Minova MCR04. Ce lecteur de badge Ethernet est spécialement conçu pour les applications de contrôle d'accès et offre une solution complète grâce à son écran LCD intégré qui permet d'afficher des informations aux utilisateurs.

Le MCR04 est un lecteur sans contact haute performance compatible avec la technologie MIFARE et la norme ISO14443-A. Sa robustesse et sa flexibilité en font un choix idéal pour notre environnement scolaire, où la fiabilité et la facilité d'utilisation sont essentielles.

* + - 1. **Caractéristiques clés du lecteur MCR04 :**
* **Interface utilisateur** : Écran LCD intégré pour l'affichage de messages (accès autorisé/refusé, instructions)
* **Connectivité réseau** : Interface Ethernet TCP/IP intégrée
* **Contrôle d'accès physique** : 2 relais de sortie pour commander les dispositifs de verrouillage des portes
* **Surveillance** : 4 entrées numériques pour l'intégration de capteurs (état de porte, bouton de sortie)
* **Modes de fonctionnement** : En ligne connecté au serveur



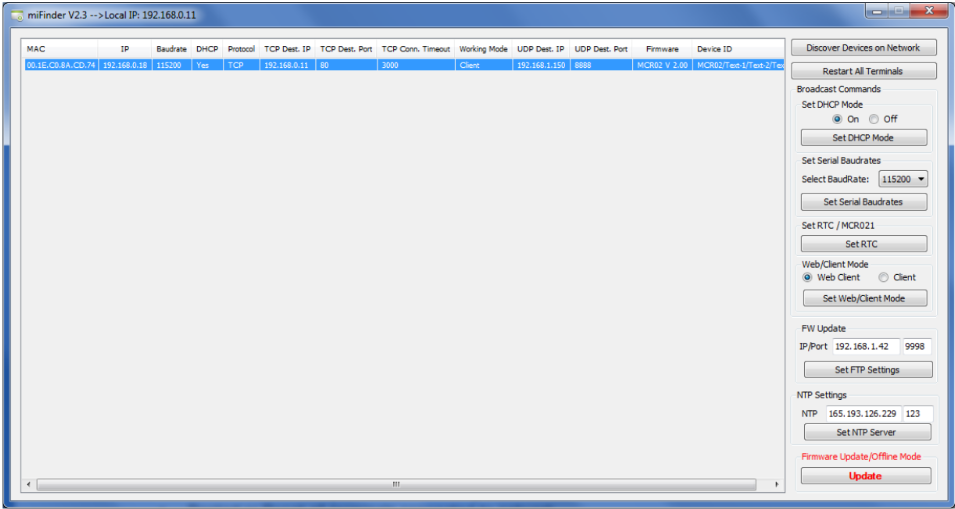
* + 1. **Configuration du lecteur MCR04**

Pour la mise en œuvre du lecteur MCR04, nous l'avons adaptée à notre infrastructure réseau et à nos besoins en matière de contrôle d'accès. Voici les étapes de configuration que nous avons suivies :

Nous avons installé le lecteur à proximité de lo porte, en veillant à :

* Le fixer solidement
* Connecter le câble Ethernet au réseau du lycée
* Raccorder l'alimentation 12V DC

Pour intégrer le lecteur au réseau du lycée, nous avons utilisé le logiciel miFinder fourni par le fabricant :



* Nous avons configuré une adresse IP statique pour le lecteur afin de faciliter son identification sur le réseau
* La passerelle et le masque sous-réseau ont été configurés conformément à l'architecture réseau existante
* Nous avons paramétré les lecteurs en mode client TCP/IP pour qu'ils se connectent à notre serveur central de contrôle d'accès

Cette configuration nous permet de gérer efficacement les accès au lycée tout en offrant une expérience utilisateur claire grâce à l'affichage LCD du MCR04.

**Présentation du lecteur Inveo**

Une image contenant texte, gadget, Appareil électronique, Téléphone mobile

Le contenu généré par l’IA peut être incorrect.

Le lecteur de badges Inveo RFID USB Desk est un dispositif polyvalent conçu pour lire divers types de badges RFID et transmettre les informations directement à un ordinateur via une connexion USB. Ce lecteur est particulièrement utile pour les applications nécessitant une saisie rapide et précise des données, telles que l'enregistre

ment des utilisateurs dans une base de données.

Caractéristiques Techniques

* **Compatibilité** : Le lecteur est compatible avec plusieurs normes de badges RFID, y compris MIFARE.
* **Modes de fonctionnement** : Il fonctionne en mode émulation de clavier, offrant ainsi une grande flexibilité d'intégration.
* **Connectivité** : Se connecte via un port USB, ce qui permet une installation facile et une utilisation plug-and-play.
* **Indicateurs Visuels et Sonores** : Équipé de trois LEDs et d'un buzzer pour fournir un retour visuel et sonore lors de la lecture des badges.

Le lecteur de badges Inveo RFID USB Desk est un outil précieux pour l'enregistrement des utilisateurs dans une base de données. Sa capacité à fonctionner comme un émulateur de clavier facilite grandement le processus de saisie des données, tout en offrant une grande flexibilité et efficacité. Il constitue une solution idéale pour les environnements nécessitant une gestion précise et rapide des informations des utilisateurs.

1. 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 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), on utilise un formulaire pour afficher les champs « email » ou « mot de passe » mais également un bouton sur lequel on peut 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 regarder 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.

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.

Une image contenant symbole, Rectangle, conception, Bleu électrique

Le contenu généré par l’IA peut être incorrect.

**1.Première interface : Page de connexion**

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 inséré 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 qu’ils ne soient pas compréhensibles.

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

Le contenu généré par l’IA peut être incorrect.

Aperçu dans la base de données :



Code source de l’interface web :

<!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>

                    <div *class*="forgot">

                        <a *href*="#">Mot de passe oublié ?</a>

                    </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 connexion.css :

\*{

    margin: 0;

    padding: 0;

    box-sizing: border-box;

    font-family: Arial, Helvetica, sans-serif;

}

body{

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    background-color: white;

    background: url(images/arriereplancyber.jpg);

    background-repeat: no-repeat;

    background-position: center;

    background-size: cover;

}

.*container*{

    width: 100%;

    max-width: 30rem;

    background-color: transparent;

    padding: 3rem 2rem;

    border-radius: 10px;

    color: white;

    border: grey solid 2px;

    backdrop-filter: blue 15px;

    overflow: hidden;

    position: relative;

}

h1{

    font-size: 3rem;

    text-align: center;

}

form{

    margin: 2rem 0;

}

.*inputbox*{

    position: relative;

    margin: 2rem 0;

}

.*inputbox* input{

    width: 100%;

    height: 4rem;

    border: none;

    outline: none;

    border-bottom: white solid 2px;

    background-color: transparent;

    color: white;

    font-size: 1.5;

    font-weight: 500;

}

.*inputbox* label{

    position: absolute;

    top: 60%;

    left: 0;

}

.*inputbox* ion-icon{

    position: absolute;

    right: 5px;

    top: 20px;

}

input:*focus* ~ label,

input:*valid* ~ label{

    top: -5px;

}

.*forgot*{

    display: flex;

    justify-content: center;

}

.*forgot* a{

    text-decoration: none;

    font-size: 1rem;

    margin-left: 4px;

    text-transform: lowercase;

    color: white;

    font-weight: 600;

}

.*forgot* a:*hover*{

    text-decoration: underline;

}

button{

    width: 100%;

    height: 3rem;

    font-size: 1rem;

    font-weight: 600;

    border: none;

    outline: none;

    border-radius: 2rem;

    margin: 1.5rem 0;

}

Nous avons créé une page « traitement\_connexion.php » permettant de se connecter à la base de données pour vérifier la correspondance avec l’adresse e-mail et le mot de passe qui ont été entrés dans les champs correspondants. Ainsi, la connexion à l’espace sera établie ou non.

Code source :

<?php

session\_start();

*require* 'config.php';

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

    $email = trim($\_POST['email']);

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

*try* {

        $pdo = new PDO($dsn, $login, $mdp, $pdo\_options);

        $pdo->*setAttribute*(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

        $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/Accès\_cam-modeAgent.php");

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

                    header("Location: Technicien/Accès\_cam-modeTechnicien.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*();

}

?>

**2.Deuxième interface : 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 présent pour quitter son espace et revenir à la page d’authentification.

Code source :

<?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: http://localhost/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*="Accès\_cam-modeAgent.css">

</head>

<body>

    <div *class*="container">

        <div *class*="wrapper">

            <h1>Espace Agent</h1>

            <ul>

            <li>

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

            </li>

            <li>

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

            </li>

            </ul>

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

        </div>

    </div>

</body>

</html>

**3.Troisième interface : 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 offre 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.  
  
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>"/>

La gestion des modes d’accès permet de choisir si l’on veut être en mode manuel ou en mode automatique en fonction de la situation. Par exemple lors d’une journée portes ouvertes, on souhaiterait plutôt utiliser le mode automatique pour que lorsque qu’une personne se présente devant la porte elle s’ouvre sans présenter de badge. En revanche, si c’est une journée de cours classique il faudrait utiliser le mode manuel pour que l’on soit obligé de présenter un badge afin de pouvoir vérifier s’il est autorisé à entrer dans l’établissement.

Ce code permet lors d’un passage de badge d’aller vérifier dans la base de données les informations de l’utilisateur afin de voir si les informations de son badge sont valides ou non. En fonction de cela, l’individu sera autorisé ou pas à accéder au lycée.

Code source :

<?php

session\_start();

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

    header("Location: http://localhost/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*="Accès\_cam-modeAgent.php">Retour à la page d'accueil</a>

    </div>

</body>

</html>

**4.Quatrième interface : 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 et 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.

Code source :

<?php

session\_start();

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

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

    header("Location: http://localhost/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*="Accès\_cam-modeTechnicien.css">

</head>

<body>

    <div *class*="container">

        <div *class*="wrapper">

            <h1>Espace Technicien</h1>

            <ul>

            <li>

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