
Serveur raspberry pour la gestion des absences

RAPPORT DU PROJET TRANSVERSE

DATE : 28 AVRIL 2023



Table des matières

1	Introduction	2
2	Etat de l'art	3
	2.1 Travaux existants	3
	2.2 Description des concepts utilisés	4
3	Analyse et conception	8
	3.1 Exigences du projet	8
	3.2 Modélisation du système	9
	3.3 Choix de conception	11
4	Implémentation et résultats expérimentaux	13
	4.1 Implémentation	13
	4.2 Résultats	17
5	Mise en place d'un protocole	19
6	Conclusion	20

1 Introduction

Le projet que nous avons mené est un projet transverse réalisé au cours de notre année d'ingénierie de troisième année, option Informatique Embarquée (INEM), à l'école CyTech. L'objectif de ce projet était de mettre en pratique les compétences que nous avons acquises au cours de notre formation en concevant et en réalisant un système concret pour répondre à un besoin réel.

Dans le cadre de ce projet, nous avons travaillé sur un sujet pouvant être utile à plusieurs autres options de dernière année de l'école, ainsi qu'à l'administration, pour répondre à un besoin commun : la gestion des absences des étudiants, notamment ceux suivant le cursus en alternance.

Le système de gestion des absences existantes était basé sur des fiches papier, ce qui rendait le processus fastidieux et sujet aux erreurs. Nous avons donc décidé de concevoir et de réaliser un système automatisé et numérique pour simplifier le processus de gestion des absences.

Pour ce faire, nous avons opté pour l'utilisation d'un serveur Raspberry Pi 4, une solution peu coûteuse, économe en énergie et facile à utiliser. Nous avons également développé une partie pour l'envoi des fiches par mail afin de permettre à l'administration de visualiser les absences.

Dans ce rapport, nous détaillerons toutes les étapes du projet, depuis la phase de conception jusqu'à la phase d'implémentation ainsi que le protocole qui pourrait être mis en place sur le site de Saint-Martin. Nous expliquerons également les choix techniques que nous avons faits pour garantir la fiabilité et la sécurité du système, ainsi que les fonctionnalités que nous avons implémentées pour faciliter la gestion des absences.

Enfin, nous présenterons les résultats obtenus lors de nos tests de validation, ainsi que les perspectives d'amélioration.

2 Etat de l'art

2.1 Travaux existants

Les projets de gestion d'absence sont très répandus dans les établissements scolaires et universitaires, car ils permettent une gestion plus efficace et fiable des absences des étudiants. Cependant, chaque projet doit être adapté en fonction des besoins spécifiques de l'établissement et des contraintes techniques.

Il existe plusieurs projets du système de gestion des absences basées sur un Raspberry Pi. Le Raspberry Pi est un ordinateur de petite taille et peu coûteux qui peut être utilisé pour construire des systèmes de gestion des absences, des systèmes de contrôle d'accès, des systèmes de surveillance vidéo, et bien d'autres applications.

L'utilisation d'un Raspberry Pi pour la gestion des absences offre de nombreux avantages, notamment une faible consommation d'énergie, une grande flexibilité et une grande fiabilité. Les systèmes de gestion des absences basés sur Raspberry Pi peuvent être facilement personnalisés en fonction des besoins spécifiques de l'établissement, et peuvent être facilement intégrés à d'autres systèmes informatiques existants.

Certains projets se basent sur l'utilisation de cartes RFID ou de codes QR pour enregistrer les présences des étudiants. Ces systèmes permettent une reconnaissance rapide et automatique de la présence des étudiants, mais nécessitent des équipements coûteux et des installations spécifiques.

D'autres projets utilisent des applications mobiles ou des interfaces web pour permettre aux enseignants de saisir les absences. Ces solutions sont souvent plus flexibles et plus facilement accessibles, car elles ne nécessitent que des smartphones ou des ordinateurs portables, mais peuvent être plus sujettes aux erreurs de saisie.

Enfin, plusieurs solutions de gestion des absences utilisent des systèmes de notifications par SMS ou par e-mail pour alerter les parents ou les responsables en cas d'absence non justifiée. Ces systèmes permettent une communication rapide et efficace entre l'établissement et les responsables des étudiants, mais nécessitent un système de gestion des contacts et un suivi rigoureux des absences non justifiées.

En somme, la combinaison d'un Raspberry Pi et d'un logiciel de gestion d'absence offre une solution économique, flexible et personnalisable pour répondre aux besoins des établissements scolaires et universitaires en matière de suivi des absences des étudiants.

Dans tous les cas, chaque projet de gestion d'absence doit être conçu et mis en œuvre en fonction des besoins spécifiques de l'établissement, des contraintes budgétaires et techniques, ainsi que des exigences réglementaires et de sécurité.

2.2 Description des concepts utilisés

Pour notre projet de système de gestion des absences nous avons décidé d'utiliser un Raspberry Pi 4. Nous avons alors configuré cette dernière et programmé un serveur grâce au mécanisme de communication par socket afin que les élèves depuis leur pc puissent communiquer au Raspberry leur présence sur site. Nous avons également écrit un programme permettant de visualiser sous forme de graphique, une synthèse globale des absences grâce à processing.

Dans cette partie nous allons donc décrire et expliquer le fonctionnement des différents concepts utilisés dans ce projet transverse.

Raspberry Pi 4

Le Raspberry Pi 4 est un ordinateur monocarte de petite taille développé par la fondation Raspberry Pi. Il s'agit d'un système sur puce (SoC) qui intègre un processeur ARM Cortex-A72 quad-core 64 bits cadencé à 1,5 GHz, jusqu'à 8 Go de mémoire vive (RAM), des ports USB 3.0, des ports Ethernet, des ports HDMI pour la vidéo, ainsi qu'un port GPIO (General Purpose Input/Output) pour la connectivité externe avec divers capteurs, modules et périphériques pour des projets électroniques..

Le Raspberry Pi 4 est conçue pour être un ordinateur de bureau économique, pour les utilisateurs qui souhaitent effectuer des tâches informatiques de base telles que la navigation sur Internet, la bureautique, le visionnage de vidéos en haute définition, la programmation et l'expérimentation électronique.

Le système d'exploitation (OS) de le Raspberry Pi 4 est généralement installé sur une carte microSD, et peut inclure différentes distributions telles que Raspbian, Ubuntu, ou encore Kali Linux. Ici, nous avons mit l'OS Raspberry Pi Os (32-bit) qui est similaire à un OS Linux, que nous avons installé avec Raspberry Pi Imager.

Lorsque le Raspberry Pi 4 est mise sous tension, le système d'exploitation est chargé à partir de la carte microSD. Une fois que le système d'exploitation est chargé, l'utilisateur peut accéder à l'interface utilisateur graphique (GUI) via un moniteur connecté ou accéder à distance à le Raspberry Pi 4 via une connexion SSH.

Le Raspberry Pi 4 est également très appréciée des passionnés de bricolage et des développeurs, car elle permet une grande flexibilité et une personnalisation complète grâce à ses nombreux ports d'entrée/sortie et son support pour différents langages de programmation.

En effet, les projets possibles avec le Raspberry Pi 4 sont nombreux, notamment la création d'un media center, d'un serveur de fichiers, d'un contrôleur de domotique, d'un système de surveillance, d'un cluster de calcul et bien plus encore. En outre, le Raspberry Pi 4 est également utilisée dans des projets éducatifs et des initiatives communautaires pour l'enseignement de la programmation et de l'informatique.

En résumé, le Raspberry Pi 4 est un ordinateur monocarte polyvalent qui peut être utilisé pour une variété de projets informatiques. Le fonctionnement de le Raspberry Pi 4 est basé sur un processeur ARM Cortex-A72 à 1,5 GHz, un système d'exploitation Linux et plusieurs ports d'entrée/sortie pour la connexion de périphériques et de capteurs.

Socket

Les sockets sont un mécanisme de communication entre deux processus informatiques sur un réseau. Ils permettent à des programmes d'envoyer et de recevoir des données sur un réseau, qu'il s'agisse d'un réseau local ou d'Internet. Les sockets sont largement utilisés dans les applications client-serveur, où un programme client se connecte à un programme serveur pour recevoir des données ou envoyer des requêtes. Dans notre cas, nous utilisons un réseau local, pour que notre programme client se connecte à notre programme serveur, les deux doivent être reliés au même réseau.

Le fonctionnement des sockets est basé sur le modèle client-serveur. Le serveur écoute sur un port spécifique pour les connexions entrantes des clients, et chaque client se connecte au serveur en utilisant une adresse IP et un numéro de port. Une fois que la connexion est établie, les données peuvent être envoyées et reçues entre les deux processus.

Les sockets peuvent être créés en utilisant différents protocoles de réseau tels que TCP (Transmission Control Protocol) ou UDP (User Datagram Protocol). TCP est un protocole orienté connexion qui garantit la livraison des données dans l'ordre et sans perte, tandis qu'UDP est un protocole sans connexion qui ne garantit pas la livraison des données ou leur ordre. Pour notre projet, nous avons décidé d'utiliser le protocole TCP.

Le processus serveur crée une socket d'écoute sur un port spécifique et attend les connexions entrantes des clients. Une fois qu'un client se connecte, le serveur crée une nouvelle socket de communication dédiée à ce client, ce qui lui permet de communiquer avec le client de manière indépendante des autres connexions.

Le processus client crée une socket de communication et se connecte à la socket d'écoute du serveur. Une fois la connexion établie, le client peut envoyer des données au serveur ou recevoir des données du serveur.

Les sockets sont utilisées dans de nombreuses applications, telles que les navigateurs web, les clients de messagerie, les jeux en ligne et les applications de partage de fichiers. Les sockets sont également largement utilisées en informatique industrielle pour la communication entre les systèmes automatisés et les capteurs.

SMTP

SMTP (Simple Mail Transfer Protocol) est un protocole de communication utilisé pour transférer des e-mails sur Internet. Le fonctionnement de SMTP est basé sur un modèle client-serveur, où le client envoie des e-mails au serveur SMTP pour qu'il les transfère au destinataire final.

Lorsqu'un utilisateur envoie un e-mail à partir de son client de messagerie, tel que Gmail ou Outlook, le client envoie le message au serveur sortant SMTP configuré pour le compte de messagerie. Le serveur SMTP envoie ensuite le message au serveur de messagerie du destinataire, qui est généralement géré par le fournisseur de messagerie du destinataire.

Le processus de transfert d'un e-mail via SMTP implique plusieurs étapes. Tout d'abord, le client envoie une demande de connexion au serveur SMTP via un port spécifique (par défaut, le port 25).

Nous avons décidé de ne pas utiliser le port 25 car ce port est souvent utilisé pour les connexions SMTP non sécurisées et peut-être bloqué par certains fournisseurs de services Internet ou par des pare-feu d'entreprise. Les connexions SMTP sur le port 25 sont également souvent associées au spamming et au phishing, donc de nombreux fournisseurs de messagerie ont commencé à bloquer les connexions sortantes sur le port 25 pour limiter ces pratiques.

Nous avons donc le choix entre utiliser le port 465 ou le port 587 qui sont tous les 2 des ports utilisés pour les connexions SMTP chiffrées par TLS (Transport Layer Security). Le port 465 était autrefois utilisé pour les connexions SMTP chiffrées par SSL (Secure Sockets Layer), mais il est maintenant considéré comme obsolète.

Nous avons donc choisi d'utiliser le port 587 car il est généralement utilisé pour les connexions SMTP soumises par des clients (tels que des clients de messagerie) et nécessite une authentification pour envoyer des e-mails. Les serveurs SMTP utilisant le port 587 peuvent également prendre en charge la négociation de chiffrement de niveau de transport pour une communication sécurisée.

Le serveur SMTP vérifie ensuite les informations d'identification et la validité du message, puis transfère le message au serveur de messagerie du destinataire. Si le serveur de messagerie du destinataire ne peut pas être contacté, le serveur SMTP peut renvoyer le message à l'expéditeur ou le stocker temporairement pour une nouvelle tentative ultérieure.

SMTP utilise également des codes de réponse pour indiquer le statut de la transmission des messages. Par exemple, le code 250 indique que la transmission s'est déroulée avec succès, tandis que le code 550 indique qu'il y a eu une erreur permanente et que le message n'a pas été transmis.

En résumé, SMTP est un protocole de communication essentielle pour l'envoi d'e-mails sur Internet. Il utilise un modèle client-serveur pour transférer les messages entre les serveurs SMTP et les serveurs de messagerie du destinataire. Le protocole SMTP est également responsable de l'authentification de l'utilisateur, de la vérification de la validité du message et de la gestion des codes de réponse pour signaler le statut de la transmission des messages.

Une IHM avec Processing

Une IHM (Interface Homme-Machine) est un ensemble de moyens techniques permettant à un utilisateur de communiquer avec une machine, un logiciel ou un système. Elle est souvent utilisée pour rendre les interactions entre l'homme et la machine plus intuitives et conviviales.

L'IHM peut être composée de différents éléments, tels que des boutons, des champs de texte, des menus déroulants, des icônes, des graphiques et bien d'autres encore. L'utilisateur peut interagir avec ces éléments à travers une souris, un clavier, un écran tactile ou tout autre périphérique d'entrée.

Son objectif principal est de faciliter l'utilisation d'un système en offrant une expérience utilisateur agréable et efficace. Elle permet également de présenter les informations de manière claire et concise, de réduire les erreurs de l'utilisateur et de fournir des feedbacks appropriés à ses actions. L'interface homme-machine est donc un élément clé dans la conception de tout système interactif, que ce soit une application, un site web ou un dispositif électronique. Elle doit être conçue de manière à répondre aux besoins et aux attentes des utilisateurs, tout en prenant en compte les contraintes technologiques et ergonomiques.

Dans notre projet nous avons décidé d'élaborer une IHM sur processing pour analyser les données de présences récoltée par l'administration.

Processing est un langage de programmation open-source créé pour faciliter la création de projets visuels et interactifs.

Il a été créé en 2001 par deux étudiants du MIT, Ben Fry et Casey Reas, et est basé sur le langage de programmation Java. Processing utilise une syntaxe simple et des fonctions graphiques pour rendre la programmation plus accessible à un public plus large.

Il permet aux utilisateurs de créer des animations, des graphiques, des interfaces utilisateur et des visualisations de données. Il peut être utilisé pour créer des projets autonomes ou pour interagir avec des périphériques externes tels que des caméras, des microphones ou des capteurs.

Nous avons choisi Processing pour afficher les graphiques et les statistiques d'absence des élèves pour plusieurs raisons :

- **Facilité d'utilisation** : Processing est un langage de programmation relativement simple à apprendre. Les fonctions graphiques de Processing sont également faciles à utiliser, ce qui permet de créer rapidement des visualisations de données.
- **Fonctionnalités graphiques avancées** : Processing offre de nombreuses fonctionnalités graphiques avancées, y compris des graphiques en 2D et en 3D, des animations, des transitions et des effets visuels.
- **Possibilité de personnaliser l'interface utilisateur** : Processing offre la possibilité de personnaliser l'interface utilisateur, ce qui permet de créer des interfaces utilisateur uniques et adaptées à nos besoins.
- **Grande communauté de développeurs** : Processing a une grande communauté de développeurs actifs qui partagent régulièrement des projets et des exemples de code en ligne. Cela rend plus facile de trouver des solutions à nos problèmes.
- **Portabilité** : Processing est conçu pour fonctionner sur plusieurs systèmes d'exploitation, ce qui permet de créer des visualisations de données qui peuvent être exécutées sur différents ordinateurs et appareils.

3 Analyse et conception

3.1 Exigences du projet

Le projet de gestion de présence au sein de l'école CyTech sur le site de Saint Martin pour les options de dernière année exige plusieurs éléments clés pour répondre aux besoins spécifiques du système.

Tout d'abord, le système doit être entièrement électronique pour remplacer le système papier existant et réduire ainsi l'impact environnemental. En outre, le système devait également être capable de stocker des données précises et fiables sur la présence des étudiants pour permettre une gestion efficace de la présence.

Le système doit être conçu pour être installé localement sur le site de Saint Martin, ce qui garantira que les étudiants ne pourront pas se connecter à partir d'un autre site pour noter leur présence, ce qui est crucial pour éviter la falsification des données.

Pour atteindre cet objectif, le système peut être installé sur un serveur local avec un accès limité aux étudiants et au personnel autorisé de l'école.

Un autre aspect important du système de gestion de présence est la sécurité. Le système doit être conçu pour empêcher les étudiants de noter la présence de leurs camarades à leur place, en utilisant par exemple l'authentification à deux facteurs (2FA) pour s'assurer que seul l'étudiant réellement présent peut noter sa présence.

En outre, le système doit être facile à utiliser pour les étudiants et le personnel de l'école afin de réduire le temps nécessaire à la gestion de la présence.

Cela peut être réalisé en utilisant une interface utilisateur simple et intuitive qui permet aux étudiants de noter leur présence en quelques clics et au personnel de l'école de visualiser rapidement les données de présence.

En résumé, le projet de gestion de présence pour l'école CyTech sur le site de Saint Martin pour les options de dernière année doit répondre à plusieurs exigences, notamment la fiabilité des données, la sécurité, la localisation du système et la facilité d'utilisation.

La mise en place de ces exigences clés aidera l'école à gérer efficacement la présence des étudiants tout en réduisant les coûts administratifs et en améliorant la précision des données de présence.

3.2 Modélisation du système

Pour mieux comprendre le fonctionnement de notre système et ses exigences nous avons réalisé un diagramme de modélisation de notre système.

Nous pouvons alors voir que le serveur Raspberry se trouve sur le site de Saint-Martin et qu'il n'est accessible que par les élèves se trouvant sur le site. Ainsi les élèves 1 à 9 se trouvant dans les différentes salles d'options peuvent s'authentifier tandis que l'élève 10 se trouvant par exemple à la maison n'a pas accès au serveur Raspberry.

Les élèves communiquent avec le Raspberry Pi via le port 8080 de chaque machine mais également du serveur.

Enfin, nous voyons bien que le serveur communique avec l'administration grâce aux mails via un protocole SMTP.

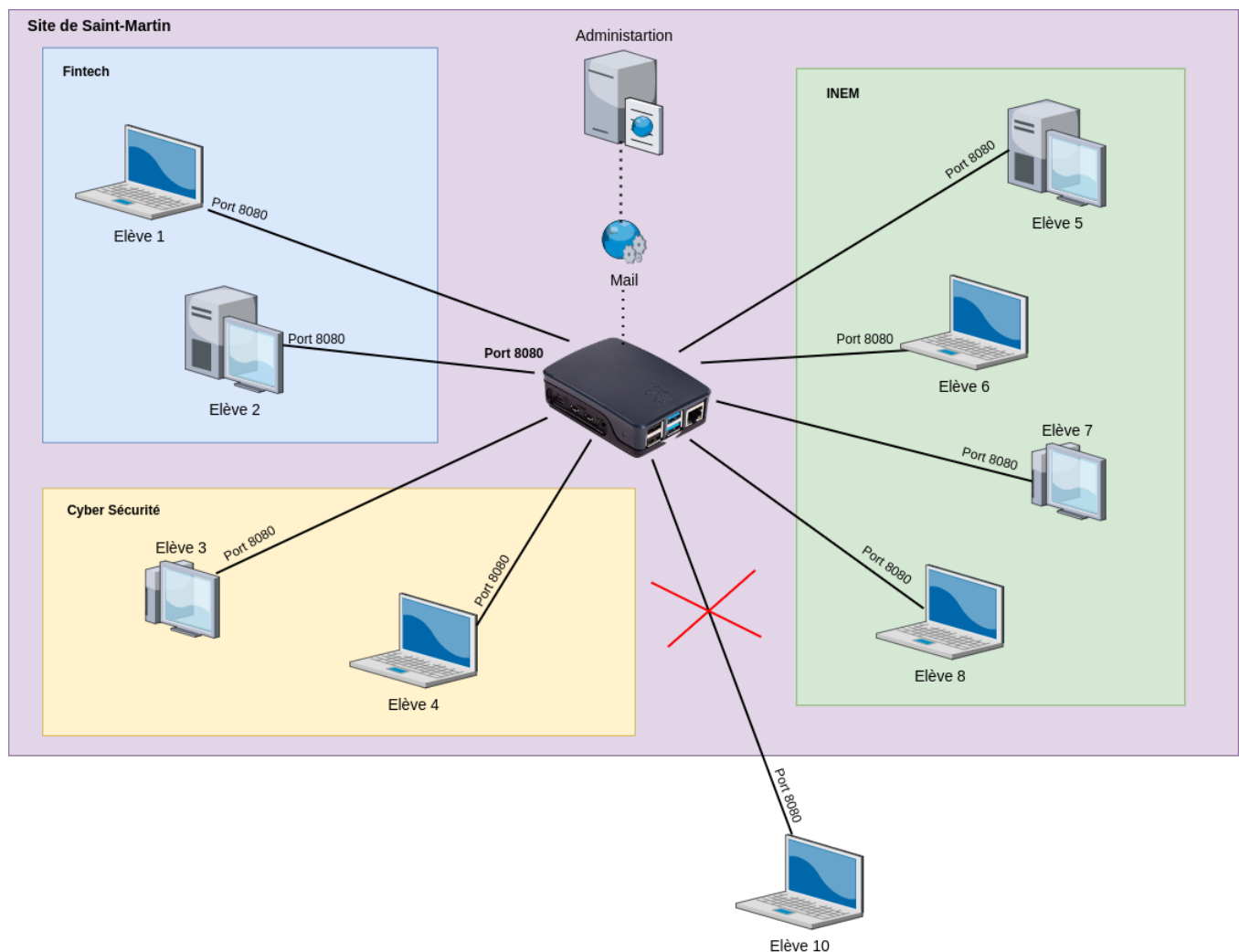


FIGURE 1 – Diagramme de modélisation du système

Le protocole TCP (Transmission Control Protocol) est un protocole de communication utilisé pour transférer des données entre des ordinateurs sur un réseau. Le modèle OSI (Open Systems Interconnection) est un modèle de référence pour les protocoles de communication en réseau, qui divise la communication en sept couches distinctes.

TCP fonctionne principalement dans les couches 4 (Transport) et 3 (Réseau) du modèle OSI. La couche 4 (Transport) assure que les données sont transmises de manière fiable et ordonnée entre les applications qui communiquent, en s'occupant de la segmentation, de la numérotation et de la reconstitution des paquets de données. La couche 3 (Réseau) s'occupe du routage des paquets entre les différents réseaux, en utilisant une adresse IP.

Lorsqu'une application utilise le protocole TCP pour envoyer des données, ces données sont découpées en segments par la couche Transport (couche 4). Chaque segment contient des informations de contrôle, telles que les numéros de séquence et d'acquittement, pour garantir que les données sont reçues dans le bon ordre et de manière fiable. Ensuite, ces segments sont encapsulés dans des paquets IP par la couche Réseau (couche 3), qui ajoute une adresse IP source et destination à chaque paquet.

Les paquets sont ensuite transmis à travers le réseau en utilisant des commutateurs, des routeurs et d'autres équipements réseau jusqu'à leur destination. Lorsqu'un paquet arrive à sa destination, la couche Réseau (couche 3) extrait le segment TCP et le transmet à la couche Transport (couche 4), qui reconstitue les données dans leur ordre d'origine et les transmet à l'application qui les a initiées.

En résumé, le protocole TCP utilise le modèle OSI pour assurer une transmission fiable des données entre les applications qui communiquent, en utilisant des segments de contrôle et en encapsulant ces segments dans des paquets IP pour être acheminés sur le réseau.

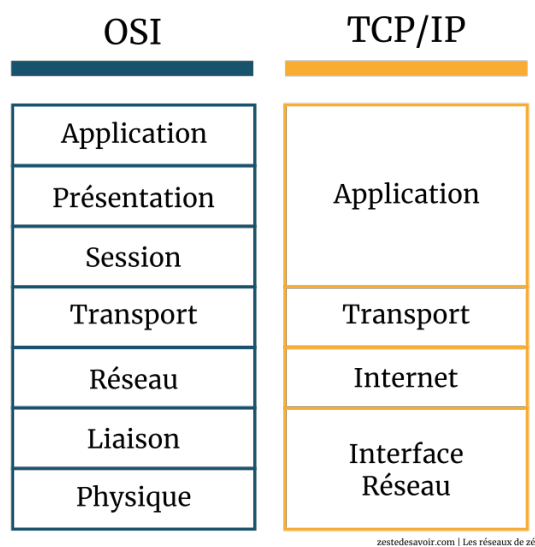


FIGURE 2 – Schéma des couches OSI et TCP

3.3 Choix de conception

Interface Web

Au début de notre projet, nous avons envisagé de développer une application web hébergée sur notre Raspberry Pi pour gérer la présence des étudiants.

Nous avons prévu pour cela de créer un site web sécurisé avec des identifiants pour les étudiants et les enseignants, qui leur permettraient de signaler leur présence à chaque demi-journée. Le professeur aurait ainsi validé la présence avant l'envoi des différentes fiche à l'administration.

L'hébergement en local sur le Raspberry Pi aurait permis de limiter les connexions à la présence sur site, mais après une analyse approfondie des exigences du projet, nous avons constaté que cette solution était trop complexe et nécessitait une maintenance continue.

En effet, l'application web aurait nécessité une mise à jour régulière pour assurer sa sécurité et sa compatibilité avec les différents navigateurs web. De plus, la mise en place d'un système d'authentification sécurisé aurait nécessité du temps et des compétences supplémentaires.

La validation par les professeurs aurait aussi demandé plus de travail étant donné que la majorité des cours dispensés en dernières années sont donné par des intervenants extérieurs. Il aurait alors fallu leur créer des identifiants à chaque fois, augmentant ainsi le temps de gestion de la BDD.

De plus, la gestion des identifiants et des droits d'accès chaque année, la sécurité du site, la validation des présences, la communication avec l'administration, et la maintenance du serveur auraient demandé beaucoup de temps et de ressources.

Connexion par bluetooth

Après avoir écarté l'idée de créer une application web pour gérer la présence des étudiants, nous avons exploré la possibilité d'utiliser une connexion Bluetooth.

L'idée était de connecter le Raspberry Pi avec les PC des étudiants et d'utiliser cette connexion pour enregistrer leur présence. Cependant, après avoir étudié de plus près cette option, nous avons réalisé que cela posait plusieurs problèmes.

Tout d'abord, il aurait été difficile de garantir que la personne qui signalait la présence ne le faisait qu'une seule fois et ne signalait pas pour plusieurs élèves.

Nous avons également constaté que cette solution nécessitait une configuration complexe pour chaque machine étudiante, ce qui aurait augmenté la charge de travail et rendu la mise en place plus difficile. En outre, nous aurions eu besoin de renommer chaque machine de chaque étudiant pour pouvoir récupérer leurs informations, ce qui aurait été fastidieux et source d'erreurs.

Enfin, il y avait également des problèmes de sécurité à considérer. La connexion Bluetooth aurait pu être interceptée et piratée, ce qui aurait pu compromettre la sécurité des données de présence des étudiants.

En fin de compte, nous avons conclu que cette solution n'était pas viable et avons poursuivi notre recherche d'une alternative pour enregistrer la présence des étudiants de manière efficace et fiable.

Socket et protocole TCP

Après avoir exploré plusieurs options pour enregistrer la présence des étudiants de manière efficace et fiable, nous avons finalement décidé d'utiliser des sockets avec le protocole TCP pour permettre à l'application Raspberry Pi de communiquer avec un serveur distant.

Dans un premier temps, nous avons envisagé de développer l'application en utilisant le langage de programmation C en raison de ses performances et de sa gestion bas-niveau des connexions réseau.

Cependant, nous avons finalement opté pour Python, car il offre une grande variété de bibliothèques et d'outils pour le développement rapide d'applications réseau, ainsi que pour sa simplicité de syntaxe et sa facilité d'utilisation.

Avec Python, nous avons pu mettre en place rapidement une application capable d'établir des connexions TCP avec un serveur distant et de transmettre des données de manière sécurisée et fiable.

Nous avons également pu intégrer facilement des fonctionnalités supplémentaires telles que la gestion des erreurs et la récupération des données de présence.

En outre, Python nous a permis de développer rapidement une interface utilisateur graphique (GUI) pour la partie client, ce qui la rendait plus facile à utiliser pour les professeurs et les étudiants.

En résumé, grâce à l'utilisation de sockets avec le protocole TCP et au choix de Python comme langage de programmation, nous avons pu développer rapidement et efficacement une application de gestion de présence pour notre projet.

4 Implémentation et résultats expérimentaux

4.1 Implémentation

Serveur.py

Ce script Python est un serveur qui écoute sur le port 8080 et qui reçoit des données de présence de la part des étudiants qui se connectent.

Le serveur crée un nom de fichier CSV pour chaque demi-journée (matin et après-midi) et pour chaque option (INEM, Cyber, IA, Fintech, Visual, BI).

Si le fichier CSV n'existe pas encore, le serveur crée un nouveau fichier CSV avec un en-tête contenant les informations qui seront saisies par la suite.

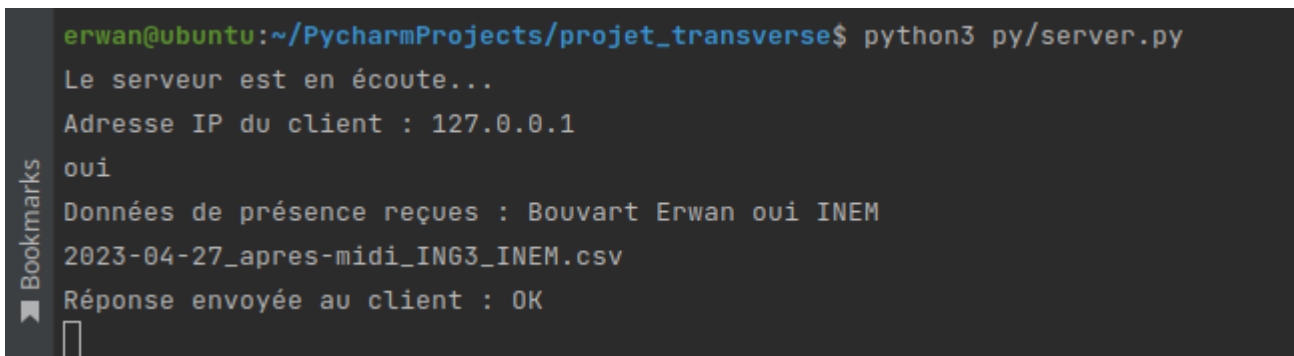
Après cette première étape le serveur vérifie dans un premier temps si l'adresse IP du client a déjà été utilisée pour signer une présence.

Si c'est le cas, le serveur envoie un message au client indiquant que cette adresse IP a déjà été utilisée. Sinon, le serveur écrit les informations de présence dans le fichier CSV.

Le script comporte également une fonction `isWeekEnd()` qui vérifie si le jour courant est un week-end (samedi ou dimanche) ou non, ainsi si le serveur reçoit des données durant le week-end il ne se passe rien.

Le serveur utilise le module `csv` pour écrire les informations de présence dans les fichiers CSV, le module `socket` pour créer un socket serveur et le module `os` pour créer des répertoires si nécessaire et vérifier l'existence des fichiers.

Enfin, le script utilise la fonction `start-server()` pour lancer le serveur. Cette fonction ouvre le socket serveur, attend les connexions entrantes et traite les données de présence envoyées par les clients. Le serveur traite les données de présence seulement si le jour courant n'est pas un week-end.



```
erwan@ubuntu:~/PycharmProjects/projet_transverse$ python3 py/server.py
Le serveur est en écoute...
Adresse IP du client : 127.0.0.1
oui
Données de présence reçues : Bouvard Erwan oui INEM
2023-04-27_apres-midi_ING3_INEM.csv
Réponse envoyée au client : OK
```

FIGURE 3 – Terminal du server.py après réceptions des infos

Client.py

Le script client.py est un script Python qui utilise la bibliothèque d'interface graphique tkinter pour créer à son démarrage une fenêtre canvas.

Cette fenêtre comporte trois champs pour entrer des informations sur la personne qui utilise l'application, à savoir le nom, le prénom et l'option.

Le choix de l'option se fait à partir d'un menu déroulant créé avec la bibliothèque ttk.

Un bouton est également présent pour envoyer ces informations au serveur.

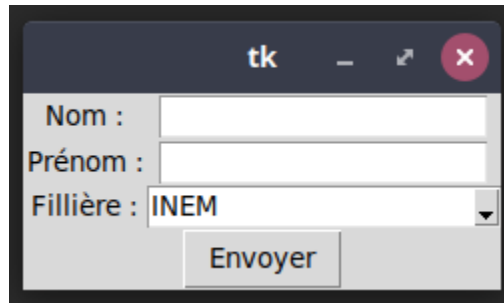


FIGURE 4 – fenêtre Tkinter pour rentrer nos informations

Le script utilise le module socket pour créer un socket client et établir une connexion avec le serveur, qui est identifié par son adresse IP et son port.

Il envoie ensuite les données de présence encodées en utf-8 au serveur en utilisant la méthode `sendall()` du socket.

Le script attend ensuite une réponse du serveur en utilisant la méthode `recv()` du socket.

Enfin, le socket est fermé et la fenêtre Tkinter est détruite.

En résumé, le script client.py permet à l'utilisateur de saisir ses informations personnelles et de les envoyer au serveur via un socket client.

```
erwan@ubuntu:~/PycharmProjects/projet_transverse$ python3 py/client.py
Données de présence envoyées : Bouvart Erwan oui INEM
Réponse du serveur : Déjà signé
erwan@ubuntu:~/PycharmProjects/projet_transverse$
```

FIGURE 5 – Terminal du client.py après envoi des infos vers server.py et réceptions de la réponse

SendMail.py

Le script python SendMail.py est un script qui nous permet d'envoyer les fiches de présences par mail.

Le code commence par importer les modules nécessaires pour le script, notamment "schedule" pour planifier l'envoi d'e-mails à une heure spécifique, "time" pour faire des pauses entre les vérifications de l'heure, "smtpplib" pour envoyer des e-mails, "datetime" pour obtenir la date et l'heure actuelles, "MIMEText", "MIMEMultipart" et "MIMEApplication" pour la construction du message e-mail, et "os" pour accéder aux fichiers CSV.

Le script configure ensuite les paramètres nécessaires à l'envoi d'e-mails, tels que le serveur SMTP, le port SMTP, l'adresse e-mail de l'expéditeur et le mot de passe de l'expéditeur.

```
# configuration de l'adresse mail d'envoi et du server smtp
smtp_server = "smtp.gmail.com"
smtp_port = 587
email_addr = "sendmailtransverse@gmail.com"
email_password = "mjxcpgwxzqkiepuq"
```

FIGURE 6 – Configuration des paramètres SMTP

Le script utilise la bibliothèque "schedule" à travers la fonction "Schedule()" pour planifier l'envoi de l'e-mail à une heure précise. Dans notre cas, l'envoi est programmé à 12h30 et 18h30 tous les jours.

La boucle "while True" permet de continuer à vérifier si l'heure planifiée est atteinte, et si oui, elle exécute la fonction "send_email()".

```
def Schedule():
    schedule.every().day.at("12:30").do(send_email)
    schedule.every().day.at("18:30").do(send_email)
    while True:
        schedule.run_pending()
        time.sleep(1)
```

FIGURE 7 – Fncion Schedule

La fonction "send_email()" est définie pour créer le message e-mail à envoyer et l'envoyer via le serveur SMTP.

La fonction commence par obtenir la partie de la journée (matin ou après-midi) en fonction de l'heure actuelle, puis elle crée les noms des fichier CSV à chercher et à joindre en utilisant la date et la partie de la journée.

Ensuite, elle construit le message e-mail en utilisant les modules MIME pour ajouter un texte et un ou plusieurs fichiers CSV en pièces jointes. Enfin, la fonction utilise le serveur SMTP pour envoyer le message e-mail.

En résumé, ce fichier Python configure un serveur d'envoi d'e-mails et envoie un e-mail tous les jours à 12h30 et 18h30 avec des fichiers CSV joints.

Projet_IHM.pde

Ce programme sous processing permet de créer une IHM (Interface Homme-Machine) qui va afficher des graphiques permettant aux responsables d'option et à l'administration d'avoir une vision plus globale des absences. Ce code est écrit en Java et utilise la bibliothèque Apache POI pour lire et écrire des fichiers Excel.

La classe principale de ce code est Tableau, qui représente un tableau Excel. Il a tout d'abord, un constructeur qui prend le nom du fichier Excel et le nom de la feuille à lire en paramètres. À partir de là, il lit les données du fichier Excel et les stocke dans un tableau bidimensionnel. Les en-têtes de colonnes sont stockés dans un tableau à une dimension et les données sont stockées dans un tableau à deux dimensions. La classe Tableau a également des méthodes pour enregistrer les données dans un nouveau fichier Excel et pour récupérer les en-têtes de colonnes et les données du tableau.

Ensuite, il y a la classe GraphApplet, qui hérite de la classe PApplet de la bibliothèque Processing. Cette classe prend en entrée un HashMap qui contient des données de comptage de dates et génère un graphique à barres en utilisant Processing. La méthode settings() est utilisée pour définir la taille de la fenêtre graphique, et la méthode setup() est utilisée pour dessiner les barres sur le graphique. La méthode draw() n'est pas utilisée dans ce cas.

Enfin, il y a la méthode main() qui crée une instance de la classe Tableau pour lire un fichier Excel, traite les données pour obtenir un HashMap de comptage de dates, puis crée une instance de la classe GraphApplet pour générer un graphique à barres à partir de ces données.

En somme, ce code utilise la bibliothèque Apache POI pour lire et écrire des fichiers Excel, la bibliothèque Processing pour générer des graphiques à barres, et Java pour faire le lien entre ces deux bibliothèques. Le résultat final est une application Java qui peut lire un fichier Excel, traiter les données et générer un graphique à barres pour les représenter.

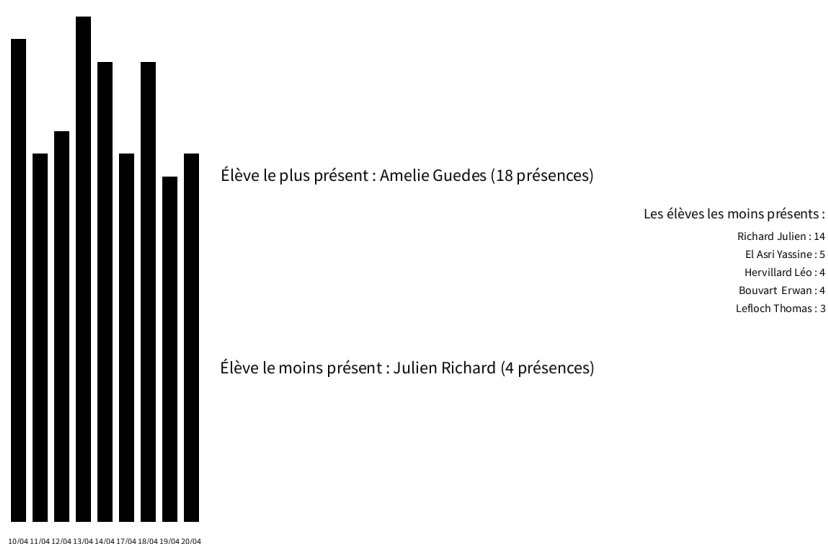


FIGURE 8 – Résultat de l'IHM processing

4.2 Résultats

Notre projet de système de présence a permis d'obtenir des résultats satisfaisants.

En premier lieu, notre script client.py s'exécute correctement et envoie les informations données vers le serveur.

Pour cela, nous avons une interface humain/machine sous forme d'un formulaire à remplir (voir ci-dessous).

Il y a, en plus des cases à remplir, un menu déroulant contenant les différentes options de l'école. Cela correspond à la seule interaction qu'il y'a entre l'étudiant et le projet, le reste est automatique.

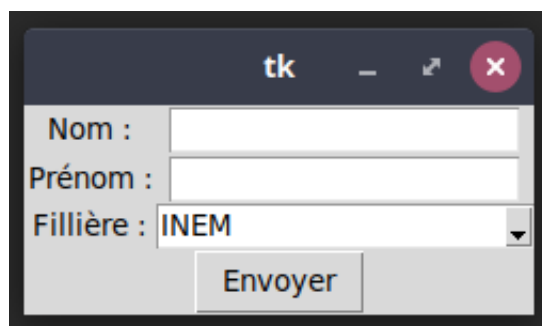


FIGURE 9 – Aperçus de l'IHM coté élève

En ce qui concerne la génération des fiches de présence au format CSV, cela a permis une organisation claire et précise des données.

Les fichiers CSV contiennent toutes les informations nécessaires pour la gestion des présences : l'adresse IP de l'élève, son nom, prénom, sa présence, son option et sa classe.

En effet, l'utilisation de l'adresse IP a été d'une grande utilité pour éviter toute fraude à la présence.

Grâce à la récupération de l'adresse IP, nous avons pu vérifier si un élève avait déjà envoyé sa présence. Ainsi, si un élève tente de renvoyer sa présence ou celle d'un autre, le serveur détecte cette tentative de fraude et ne prend pas en compte cette nouvelle demande de présence.

Les fichiers sont générés automatiquement en fonction de la date et de l'option de l'élève, ce qui facilite la recherche et la vérification des données.

	A	B	C	D	E	F
1	Adresse IP	Nom	Prénom	Présence	Filière	Classe
2	127.0.0.1	Erwan	Bouvar	oui	INEM	ING 3
3						
4						
5						
6						

FIGURE 10 – CSV créé par le Raspberry

En utilisant le protocole SMTP en Python, nous avons pu automatiser l'envoi des fiches de présence par mail.

Cela a permis aux enseignants de recevoir les fiches de présence dès la fin des cours, sans avoir besoin de les récupérer manuellement auprès du Raspberry Pi.

De plus, en envoyant les fiches de présence par mail, nous avons offert une traçabilité des présences, permettant aux enseignants de disposer d'un historique complet des présences des élèves pour chaque séance.

Enfin, en cas de perte de connexion Internet, les fiches de présence sont stockées sur le Raspberry Pi et peuvent être récupérées ultérieurement grâce à une connexion câblée.

Cette fonctionnalité offre une grande sécurité en garantissant que les fiches de présence sont toujours accessibles, même en cas de problème de connexion.

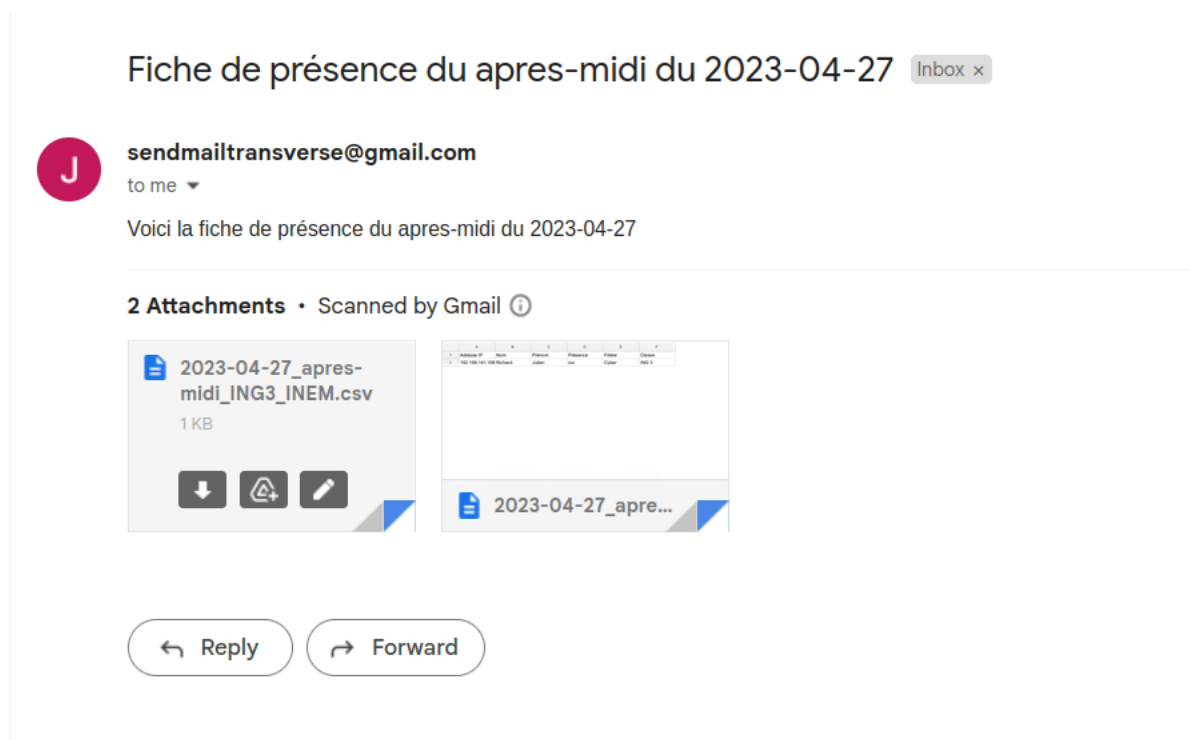


FIGURE 11 – Mail envoyé par le Raspberry

En somme, notre système de présence basé sur l'utilisation de sockets TCP, de fichiers Excel et de protocole SMTP a permis d'obtenir des résultats concluants et a permis une gestion plus efficace des présences.

5 Mise en place d'un protocole

Dans cette partie nous allons voir quel est le protocole pour la mise en place du système de gestion des absences qui pourrait être mis en place par l'école sur le campus de Saint Martin si ce projet devait se concrétiser :

Matériel nécessaire :

- Raspberry Pi 4
- Service client installé sur les ordinateurs des élèves
- Accès Internet

Configuration du Raspberry Pi 4 :

1. Télécharger le système d'exploitation (OS) sur le Raspberry Pi 4
2. Installer le script du serveur et le script d'envoi de mails sur le Raspberry Pi 4
3. Configurer le Raspberry Pi 4 pour qu'il démarre automatiquement au démarrage

Installation du service client :

1. Avant la distribution des ordinateurs ou en début d'année, présenter aux élèves le service client et expliquer son fonctionnement
2. Installer le service client sur les ordinateurs des élèves

Utilisation du système de gestion des absences :

1. Les élèves lancent le service client sur leur ordinateur en arrivant le matin et en début d'après-midi depuis leur salle de classe
2. Les élèves s'authentifient en utilisant leur Nom, Prénom et option (à choisir parmi la sélection)
3. Le système enregistre automatiquement la présence des élèves dans chaque option (INEM, CyberSécurité, Fintech, Visual Computing, IA et BI)
4. A la fin de chaque demi-journée, l'administration reçoit les fiches de présence de chaque option par mail

Utilisation de la visualisation des absences :

- Lorsque l'administration le souhaite elle peut lancer le programme `Projet_IHM.pde` depuis son pc pour afficher le graphique de synthèse des présences et avoir un meilleur aperçu du taux d'absentéisme.

En cas de problème :

- En cas de non-envoi des mails, dû par exemple à un problème de wifi, il sera possible de se connecter par câble au Raspberry pour récupérer les fiches de présence localement
- Les élèves peuvent contacter le support technique pour obtenir de l'aide avec le service client
- L'administration peut contacter l'équipe technique pour résoudre les problèmes liés au Raspberry Pi 4 ou aux scripts de serveur et d'envoi de mails.

Il est important de noter que ce protocole doit être communiqué clairement aux élèves et à l'administration pour une utilisation efficace du système de gestion des absences.

6 Conclusion

Problèmes rencontrés

L'un des principal problème rencontré lors du travail sur ce projet transverse à été la connexion wifi du site de Saint-Martin.

En effet, la connexion internet sur ce site est difficile à utiliser lors d'un projet de ce genre.

D'une part la connexion filaire au réseau ne fonctionne pas dans toutes les salles (notre salle d'option par exemple), il n'a donc pas été envisageable de l'utiliser.

De plus la connexion sans fil est difficile d'accès, il faut avoir un compte pour s'y connecter et la connexion ne dure que 5h avant de devoir saisir à nouveau nos informations d'authentification. Cela n'était donc pas envisageable car le serveur à besoin de tourner et d'être connecté en continu. Pour finaliser le projet, nous avons décider d'utiliser nos partages de connexion sur les téléphones portables afin que cela fonctionne correctement.

D'autre part, le projet transverse est un projet qui se déroule sur 1 an, malheureusement, le temps qui nous était alloué dans l'emploi du temps était vraiment faible.

Comme nous sommes tous les 4 en contrat de professionnalisation ou en alternance, en semaine nous ne pouvions pas vraiment avancer sur le projet. Il ne nous restait que les week ends mais les autres projets prenaient souvent le pas sur le projet transverse.

Améliorations futures

Dans cette partie nous allons aborder les axes d'améliorations possible concernant la suite du projet.

En effet, notre idée au départ était de commencer ce projet de système de gestion des présences pour l'école, sachant que nous n'arriverions pas à la fin de l'année à un système assez performant pour être utilisé. Cependant nous aimerions que ce projet puisse se transmettre d'une année sur l'autre aux futures promotions d'INEM afin de continuer à développer ce projet et arriver à un résultat opérationnel qui pourrait être utilisé par CyTech et maintenu soit par le service entreprise soit par la classe INEM elle même.

- Avoir un système plus performant :
 - Récupérer au début d'année une liste des élèves par option pour que tout les élèves de la classe soient inscrit dans le csv. Ainsi l'authentification SMTP servirait à ajouter à la ligne de l'élève s'il est présent ou non et permettre d'avoir un aperçu plus rapide des absents.
 - Trouver une solution pour la gestion du wifi sur le site de Saint-Martin
- Mailling :
 - Créer un service qui permet d'envoyer les mails au cas où le raspberry pi s'éteint
- IHM Processing :
 - Proposer une IHM avec une fonction de recherche dans les dates.
 - Avoir une vision moins globales des absences avec une vue permettant de voir les absences de chaque élève.
 - Faire une fonction qui permet d'envoyer un rapport général par mail en plus des fichiers excels envoyés.

- Sécurité :
 - Créer un firewall (ex : IPTABLE) pour empêcher un attaquant de pirater le serveur.
 - Améliorer la sécurité lié aux mails en améliorant le chiffrement du mot de passe
- Changement du système pour une utilisation plus simple :
 - Système de badgeage
 - Système de Qr code sur le portable des professeur ou une tablette
 - Application mobile

En conclusion, la création d'un serveur Raspberry Pi pour la gestion des absences est un projet transverse complexe qui demande une analyse minutieuse des différentes options disponibles pour atteindre l'objectif de manière efficace et fiable.

Nous avons dû faire face à plusieurs défis et problèmes lors de la conception de ce système, mais avons réussi à surmonter ces obstacles grâce à notre collaboration et notre créativité.

Au cours de notre projet, nous avons examiné plusieurs options pour enregistrer la présence des étudiants, telles que la création d'une application web et l'utilisation de la connexion Bluetooth. Cependant, après avoir examiné de plus près ces options, nous avons réalisé qu'elles étaient trop complexes ou peu fiables pour répondre aux exigences de notre projet.

Finalement, nous avons opté pour l'utilisation de sockets avec le protocole TCP pour permettre à l'application Raspberry Pi de communiquer avec un serveur distant. Nous avons également utilisé la bibliothèque Tkinter pour créer une interface utilisateur graphique (GUI) conviviale pour l'utilisateur final.

En utilisant des outils tels que Python, Tkinter et Processing, nous avons pu développer une interface utilisateur intuitive et facile à utiliser pour les professeurs, ce qui a facilité la gestion des absences et la génération de graphiques rendant le tout plus visuel.

Grâce à notre travail acharné et à notre détermination, nous avons réussi à créer un système efficace pour la gestion des absences des étudiants, qui répondait aux exigences de notre projet. Ce projet transverse nous a permis de développer nos compétences en programmation, en collaboration et en résolution de problèmes, tout en nous donnant une expérience précieuse de travail sur un projet concret.

Dans l'ensemble, la création d'un serveur Raspberry Pi pour la gestion des absences peut être un projet enrichissant et stimulant pour les étudiants, qui permet de développer des compétences en programmation, en réseau et en conception d'interfaces utilisateur. Nous espérons que ce projet pourra se transmettre aux générations suivantes et continuer à se développer.