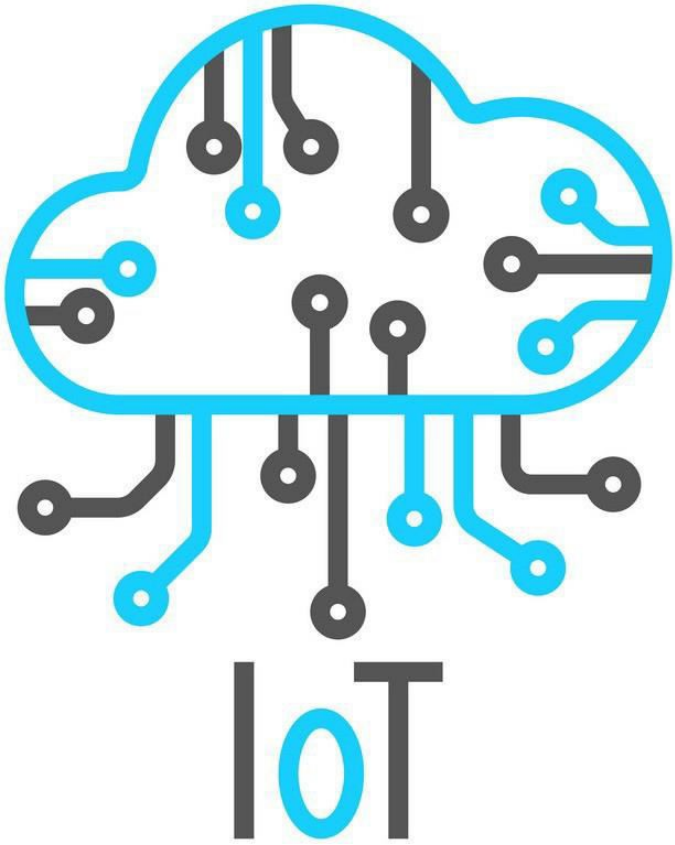


Rapport du projet

GROUPE NASTA

Nassim MESSAOUDI – Tarek NAIT SAADA – Marc JEAN PIERRE

Internet des Objets | 3 Novembre 2019

Encadrants : Aomar OUSMANI – Massinissa HAMIDI

SOMMAIRE

INTRODUCTION

Vous êtes étudiant ? Vous galérez pour trouver votre salle de cours ?
Alors ce projet est fait pour vous !

Dans le cadre du cours d'internet des objets, nous avons réfléchi à concevoir un objet connecté répondant à une problématique précise celle d'améliorer l'expérience de l'étudiant au sein de l'Université Paris 13.

Plus précisément, notre solution technique va permettre de centraliser l'accès à l'information afin de permettre à l'étudiant un accès plus facile et user-friendly à ses informations pédagogiques et scolaires.

En d'autres termes, nous allons concevoir une borne connectée permettant à l'étudiant d'afficher son planning de la journée, son compte Izly et d'autres informations relatives à sa scolarité.

Cahier de charges

Dans le but de créer une borne connectée nous devons établir au préalable les objectifs principaux de cet objet connecté, ainsi que les objectifs secondaires de ce dernier.

- Objectifs principaux :
    - Borne connecté menu d'un écran tactile permettant d'afficher le planning de l'étudiant.
    - Connexion à l'interface par le biais de la carte étudiant (tag RFID).
    - Incorporation de l'espace ENT dans l'application (Information pédagogique).
    - Récupérer des informations à partir d'une base de données (formation de l'étudiant) et de l'hyper planning et de l'ENT.
    - Regrouper les applications de l'université en une seule application.

- Besoins Fonctionnels :

Ce sont les fonctionnalités du système qui permettent en premier lieu à la borne de :

    - Reconnaitre l'étudiant via lecteur RFID .
    - Afficher le planning de l'étudiant.
    - Afficher des informations concernant les salles de TP libres.
    - Afficher les informations pédagogiques concernant l'étudiant.
    - Signaler une carte perdue.
    - Récupérer des informations depuis le sercal, l'ent et l'hyperplanning.
    - Afficher le stockage disponible pour l'étudiant dans les serveurs de l'université.
    - Avoir accès au calendrier annuel.
    - Avoir accès au plan de l'université.
    - Permettre à l'étudiant de recharger sa carte izly.
    - Afficher a l'étudiant son quota d'impression restant.
    - Afficher des informations concernant la météo.
    - Interactions avec l'utilisateur sur écran tactile.
    - Suppression de toute donnée de l'utilisateur qui est dans la mémoire de la borne à la fin de son utilisation.

Les fonctionnalités du système permettent aussi à l'utilisateur de :

- De poser sa carte sur le lecteur.
- D'interagir avec la borne en utilisant l'écran tactile.
- De récupéré différentes informations concernant l'utilisateur.

- Les besoins Non-fonctionnels :

La borne est conçue pour répondre aux exigences suivantes :

Fiabilité : La Borne doit fonctionner sans erreurs et sans faille de sécurité toute information concerne un seul et unique étudiant. La borne doit donner un accès rapide à l'information.

Les erreurs : D'abord minimiser les erreurs en offrant à l'utilisateur un accès limité aux données et en cas d'erreur ou mauvaise manipulation, cette dernière doit renvoyer un message d'erreur.

Ergonomie et bonne interface : L'utilisation de la borne doit être simple, intuitive et facile.

Régulations et normes

Manipulation des données :

Dans le contexte de ce projet nous allons créer une base de données contenant deux utilisateurs (moi et mon binôme). On crée deux utilisateurs seulement pour nous protéger juridiquement conformément à la réglementation RGPD (GDPR). Donc pour ce présent rapport je donne, moi, MESSAOUDI Nassim, ainsi que mon binôme ; Tarek NAIT SAADA la permission au software et hardware conçu par NASTA de lire et manipuler mes données dans un but éducatif.

Suite au refus d'accès au serveur LDAP de l'université Paris 13, on devait réfléchir à une autre façon pour faire marcher notre idée. On a donc décidé d'héberger notre propre base de données sur la machine host ou sur une machine distante.

Normes/sécurité :

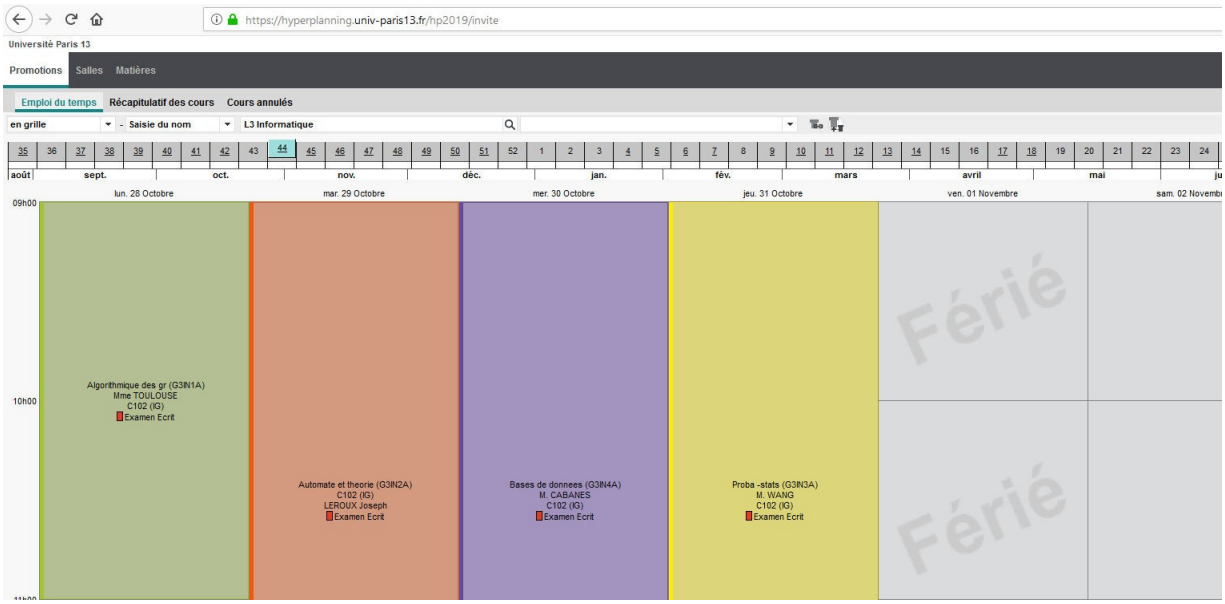
- Toutes les communications de données dans notre solution se font par ssl. Dans la majorité des cas on utilise le service d'authentification de l'université pour afficher les informations personnelles (stockage, quota, infos pédagogiques). Donc la borne sera sécurisée des attaques MITM (Man In The Middle Attacks).
- En cas d'échec de la vérification le logiciel retentera 2 fois puis enverra un mail à la personne détentrice de la carte en signalant que la carte a été perdue.
- Au retrait de la carte toutes les données de connexion seront remises à zéro. Rien ne sera sauvegardé sur le serveur distant ou sur la borne.
- La borne sera bien protégée physiquement de manière à ce qu'une tierce personne n'ait pas accès au microcontrôleur, En cas de modifications hardware, la borne se met en mode sécurisé empêchant tout entrée ou sortie et donc n'interagira plus jusqu'à intervention de l'admin (il en sera averti).

État de l'art

En répondant à un problème réel que les étudiants font face tous les jours, le nombre de solutions qui proposent la même solution que notre projet est malheureusement faible ou constitue une partie seulement du projet

Tout d'abord intéressons-nous aux solutions existantes qui permettent à l'étudiant d'accéder à son emploi du temps.

1. Hyperplanning :



- Type de solution : Application mobile et/ou Application web.
- Description de la solution :

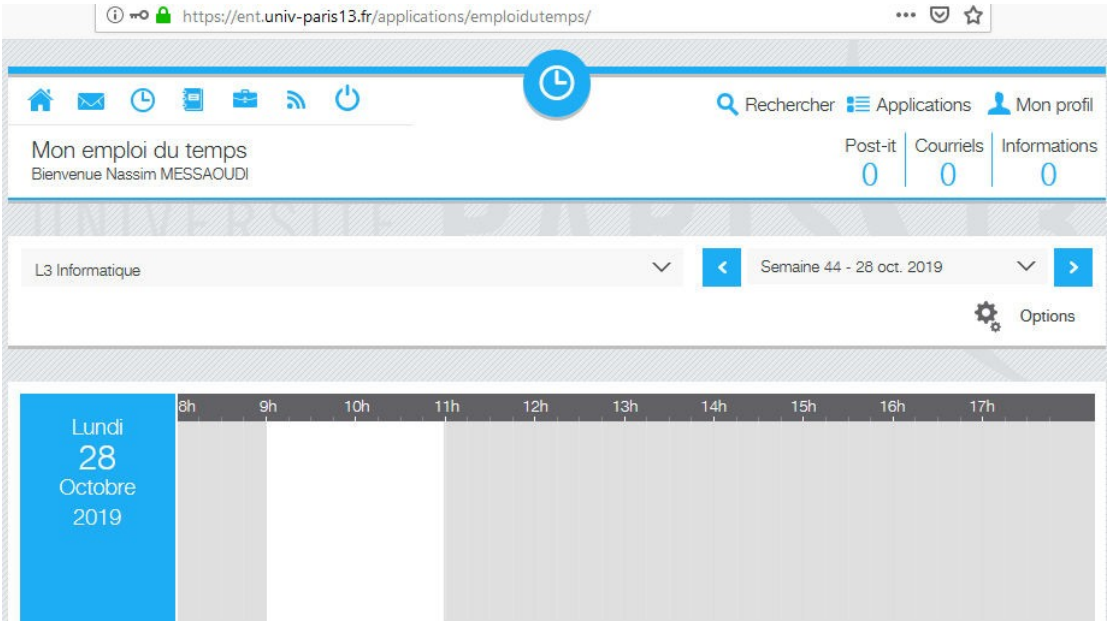
    - Il est possible d'accéder au planning de l'étudiant via le portail (sous-domaine) hyperplanning de l'université (https://hyperplanning.univ-parIS13.fr/hp2019/invite?fd=1), puis il faut chercher la formation de l'étudiant puis sélectionner la semaine correspondante.

- Quant à la solution mobile, elle nécessite un lien hyperplanning propre à l'établissement ou un QR code affiché par l'établissement. Or, on a fait notre enquête et le seul panneau affichant hyperplanning ne dispose pas de QR code.

Donc il est impossible à l'étudiant de se connecter à hyperplanning avec son téléphone et donc de visualiser son planning.

- Critique : En plus de l'interface graphique non user-friendly, l'université ne dispose pas assez de « panneaux planning » manipulant hyperplanning pour permettre une visualisation du

2. Application web Emploi du temps de l'ENT :



- Type de solution : Application web.
- Description de la solution :
  - Après avoir accéder à son espace ENT, l'étudiant peut consulter son emploi du temps grâce à l'application emploi du temps (http://ent.univ-paris13.fr/applications/emploidutemps).
- Critique : L'interface web sur ordinateur est convenable et sert à afficher de manière efficace l'emploi du temps, cependant l'étudiant étant généralement en déplacement, et donc utilise très certainement un appareil mobile or l'interface du site sur mobile ne permet pas une utilisation et un affichage convenable de l'emploi du te

Intéressons-nous désormais à l'affichage des quotas des étudiants.

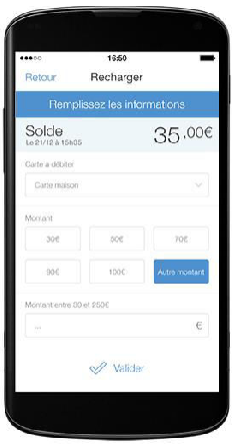
Pour l'instant il n'existe pas de solution effective qui permet d'afficher le quota d'impression restant pour un étudiant donné.

Pour ce qui est du quota de disque sur les serveurs de l'université, il est possible de récupérer le stockage restant en passant par ssh et en saisissant la commande correspondante.

Regardons maintenant la partie Izly de notre projet. Pour ce qui est de la partie izly, on utilisera évidemment l'application Izly directement au lieu d'implémenter une autre solution qui communique en SOAP avec les serveurs d'Izly.

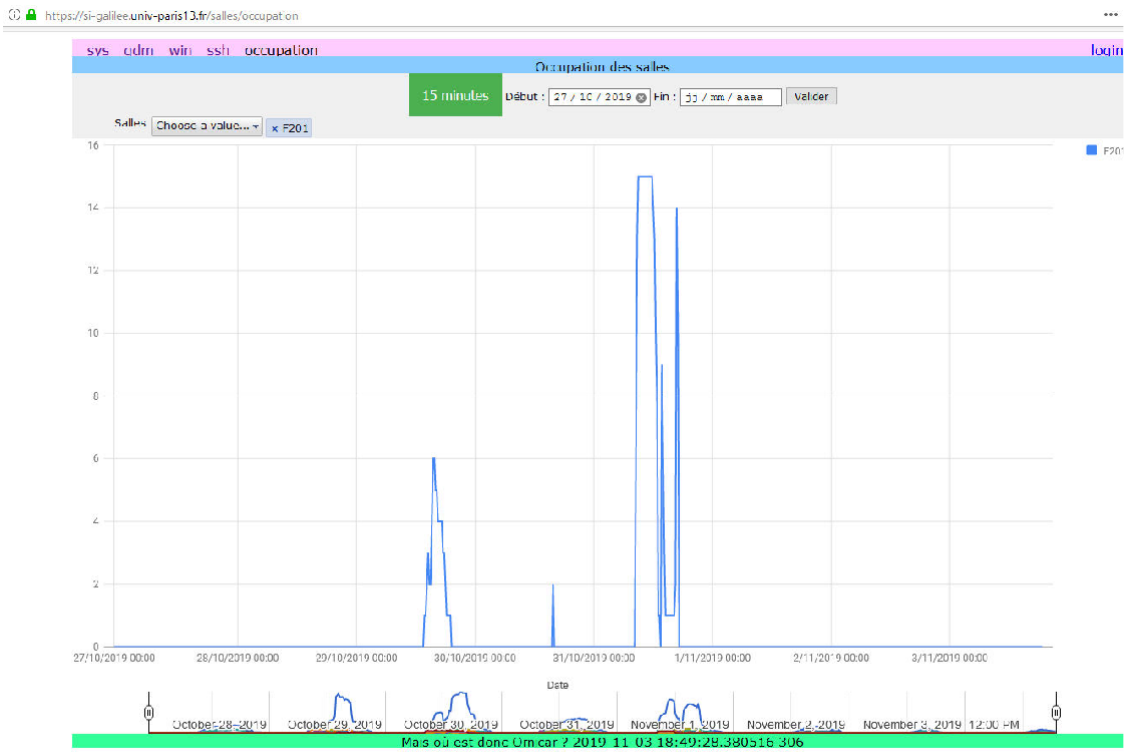
3. Izly :

- Type de solution : Application mobile ou web.
- Description de la solution :
  Application permettant d'accéder, consulter, et recharger son compte Izly.
- Critique : Pas d'éléments de critique.



Notre borne permettra aussi d'afficher les salles TP libres au niveau de l'institut Galilée. Il existe un sous-domaine hébergé par l'université qui permet d'avoir une liste d'informations exhaustive sur les salles de TP comme le nombre de personnes connectés, ce qui permet donc de savoir le nombre de place libre

4. Salles TP libres :



- Type de solution : Application web.
- Description de la solution : Portail appartenant au domaine univ-paris13 qui permet de recueillir beaucoup de données sur l'utilisation des salles de TP (https://si-galilee.univ-paris13.fr/salles/occupation).
- Critique : Le nombre de personnes qui connaissent l'existence de ce service (https://si-galilee.univ-paris13.fr/salles/occupation) peut être compté sur le bout des doigts et n'as pas été conçu pour répondre à la même problématique que celle que notre projet aborde. Cette application a été conçue pour une analyse plus profonde des tendances d'utilisation et heures d'utilisation des salles machines.

Comparaison de notre solution avec les autres :

Le point fort de notre solution est en effet la centralisation des informations de l'étudiant par une SEULE et UNIQUE interface simple et intuitive.

L'interface tactile est plus user-friendly, l'accès à l'emploi du temps et aux informations concernant l'étudiant est simplifié et immédiat.

Schéma simplifié de la borne connecté :

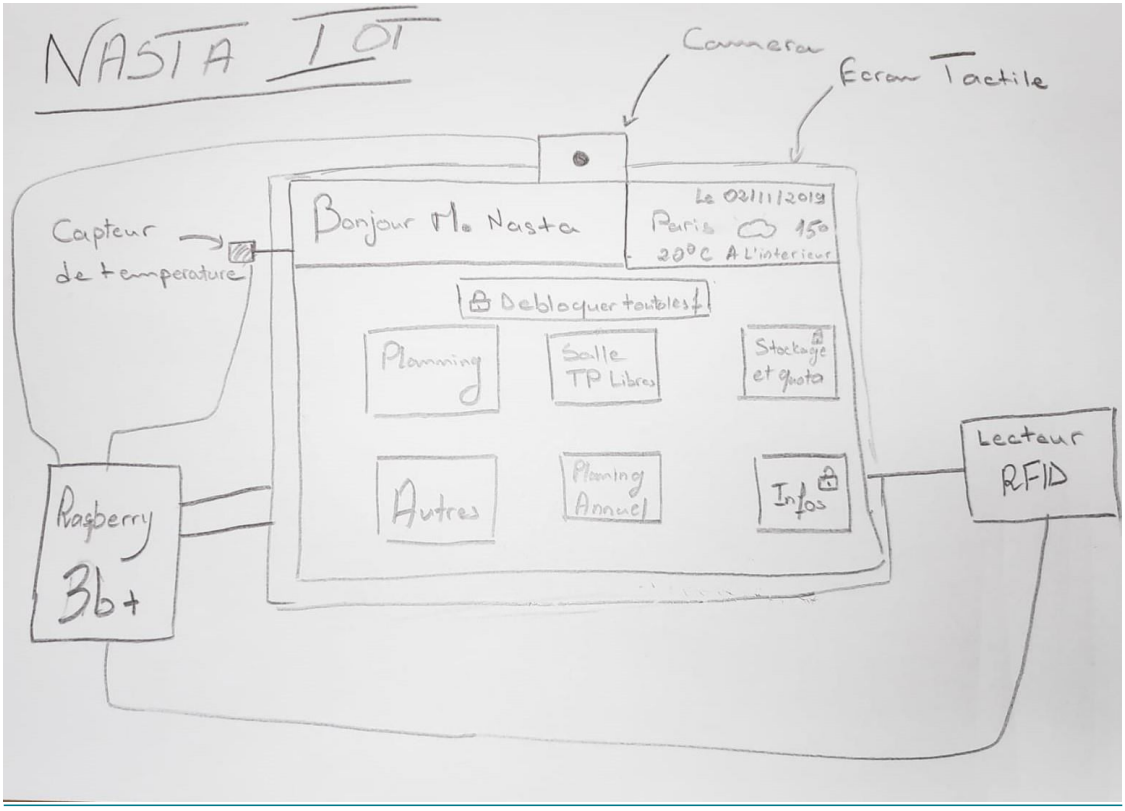
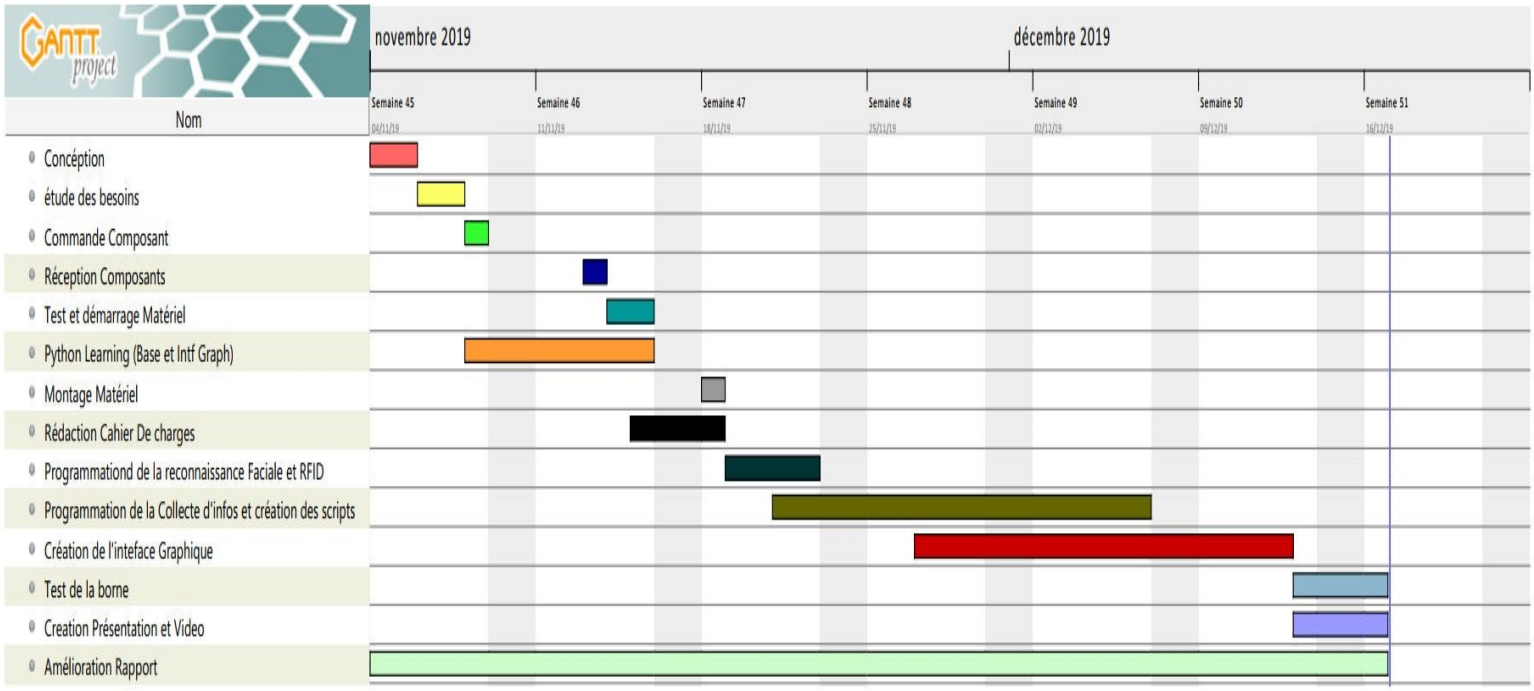


Diagramme de Gantt & Organisation

Pour la réalisation du Projet dans les délais nous avons mis en place un planning qui définit les étapes à suivre et ce jusqu'à la fin du Projet. Nous avons estimé le temps pour chacune de ces étapes. La réalisation de ce Diagramme a été faite avec le logiciel Gantt Project 2.8 qui permet de représenter les différentes étapes et établir par la suite le diagramme de GANTT ci-dessous :



I/ Front-end

A/ Interface graphique GUI

L'interface graphique a été faite en Qml et en C++.
Elle est organisée en trois window, la première est celle utilisé pour l'accueil, la deuxième est celle qui gère l'authentification elle s'ouvre et se ferme après authentification.
La troisième window est utilisé pour le menu principal, pour la suite des sous menus et options, nous avons préféré utiliser des banners pour optimiser les ressources et la vitesse d'exécution sur le raspberry.
La mise en place de nos propres bibliothèques, nous permet la communication entre QML et C++.
Pour le planning on utilise les méthodes misent à disposition par Qt et le sqlite pour récupérer les events dans le calendar, celui-ci est proposé sous la forme de listView.

L'ensemble des images sont insérer grâce aux méthodes de Qml. Lors du signalement le mail est envoyé grâce à un script python.
Concernant le keyboard, l'utilisation de inputPanel permet l'exécution du keyboard dans un banner.
Les informations pédagogiques sont récupérées grâce à une commande curl, elles sont insérées dans un fichier json puis celui-ci est parsé avec du C++.

Le bash a également été utilisé pour la récupération du stockage et du quotas.
La météo est gérée par une api openWeather. Les valeurs d'un banner est sauvegardé permettant le changement entre banner sans devoir recommencer toutes les requêtes.
La window d'authentification quant à elle, se ferme et reviens à la window d'accueil après un délai de 10 secondes.
Les buttons eux, sont définie comme mouseArrea pour les interactions.

II/Back-end

A/ Serveur ldap et docker

Pourquoi un docker concernant l'installation du serveur ldap ? Pour ça portabilité son efficacité et ça rapidité.
Il suffit de lancée une ligne de commande pour déployer le server ldap avec toutes les options (adresse, mot de passe etc…)
Pour la manipulation du server ldap nous avons utilisé apach directory studio. Pour créer le schéma et remplir le ldap nous avons tout d'abords créé un fichier ldif contenant toutes les informations souhaitées, et nous avons transformé ce fichier en fichier ldap.
Concernant le docker nous avons dut faire un grand nombre de test d'images afin de choisir la plus légère qui remplit pleinement nos besoins.

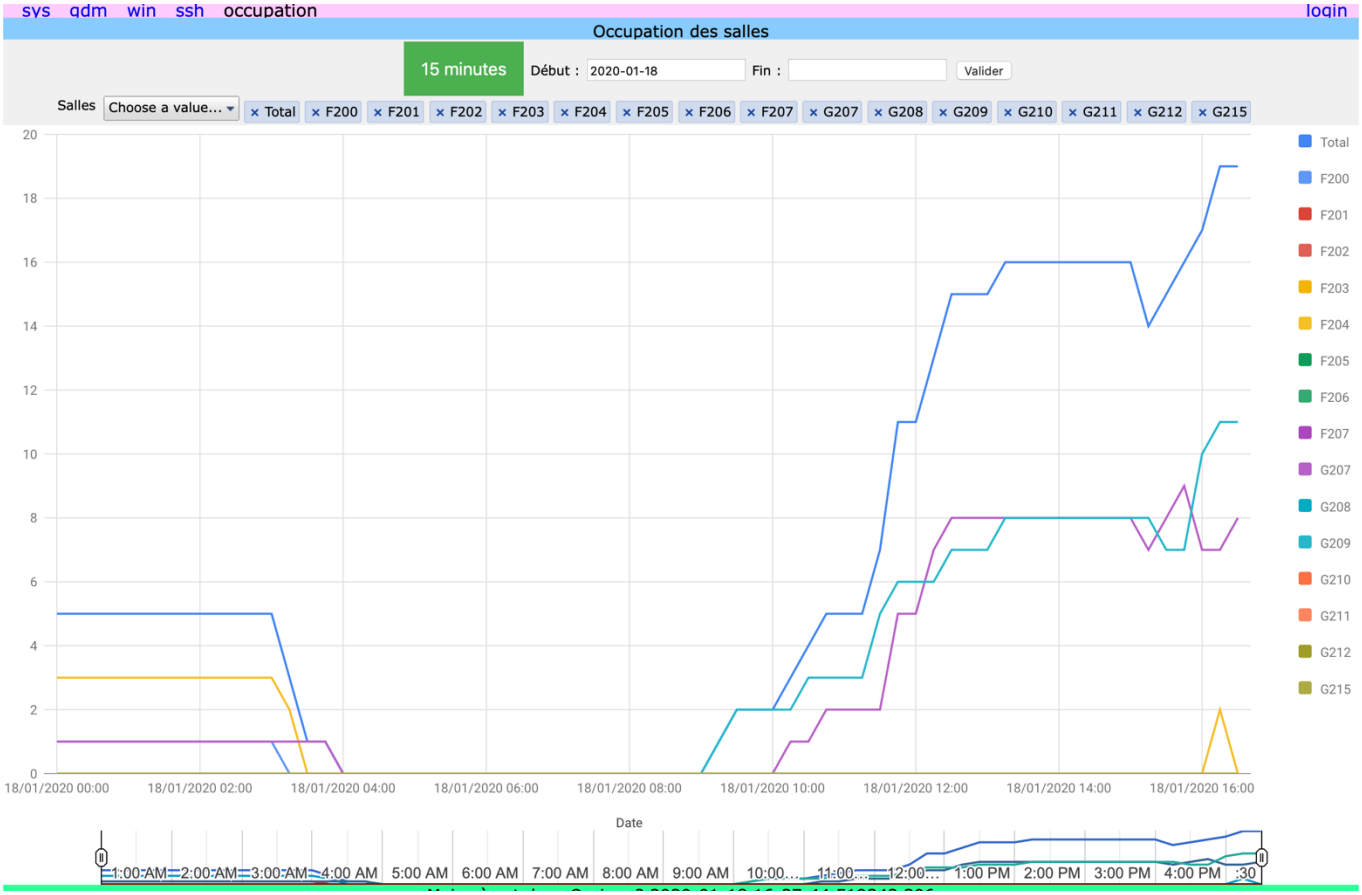
B/ Récupération des informations

Hyperplanning

L'obtention des informations liées à l'hyperplanning a été envisagé par le biais d'un scripte python, celui-ci récupère le fichier ical grâce au lien hyperplanning fournie sur le site.
Par la suite le script réorganise et construit une base de données qui sera directement utilisé par le script à charge de l'interface graphique.

Nous déterminons que la première approche sous script python a plusieurs avantages et inconvénients.
Les avantages tel que, l'exécution du scripte à une seul reprise, économise une quantité de ressource non négligeable.
De plus le faîte de dissocier cette partie du programme permet également les mise à jour et le débogage rapides.
Les inconvénients de cette approche, sont liés à l'actualisation des données de l'hyperplanning, en effet en cas de changement celui-ci sera effectué sur notre machine lors du prochain lancement uniquement, soit au prochain démarrage.
Une méthode a été envisagé pour obtenir de manière continue les informations à chaque utilisation grâce au web scraping mais nos recherches peu fructueuses nous ont amené à rester sur notre première approche.

Occupation



Les informations liées à l'occupation des salles, son récupéré grâce à un scripte python.
Celui-ci construit une url grâce aux arguments fournis, afin d'accéder à la page désirée qui rassemble les informations que l'on recherche. (Voir exemple.)
L'inconvénient de ce procédé est que nous avons une incertitude de 15 minute imposée par le site.
Nous n'avons pas trouvé de moyens plus efficaces pour rassembler cet ensemble d'informations de manière directe et continue.
Principalement les fichiers qui se chargent de récupérer les informations sont (occupation.py, getnbpeople.py).

Quotas

Le quota est récupéré dans une base de données ici fake. Le script reste fonctionnel avec une base de données existant.
Le stockage lui est récupéré grâce à un script bash qui se connecte à l'université et récupère le dernier log des stockages.

III/ Coût et composants

Pour ce projet plusieurs composants sont présents.
Nous avons en premiers lieu l'écran, d'une valeur de 55 € et d'une dimension de 7 pouces.
Le raspberry 3B+ à 25€.
Le lecteur RFID 10€.
Câble + power supply 5€.
Le plexiglace pour la construction du boîtier contenant l'ensemble du système.

Problèmes rencontrés :

Le premier problème que nous avions rencontré, était lié à la construction de la requête utilisée pour récupérer les informations de l'hyperplanning, car il nous fallait un ID, mais celui-ci changeait à chaque requête, et était fourni de manière interne par le service en charge de l'hyperplanning.
Le second problème lui était lié au choix du langage à utiliser pour construire notre interface graphique. Le projet étant à ce stade presque entièrement en python, nous étions partis pour celui-ci, mais nous avons déterminé que Qt et C++ étaient beaucoup plus intuitifs car pourvus d'une grande communauté d'entre aide, et de plus ce langage ressemble fortement à certains langages déjà visités en cours (java).
L'interface graphique est organisée en banners afin de limiter au maximum l'utilisation de plusieurs fenêtres limitant donc la consommation des ressources de notre raspberry.
La gestion des ressources de notre raspberry nous induisait vers une dynamique d'économie des ressources.
Nous avons également rencontré un problème sur la compilation de l'interface graphique, en effet on a opté pour une cross-compilation, c'est à dire que l'application marchera sur X64 (Linux) et sur arm64 (donc sur notre raspberry).

En effet la partie IZLY est affichée grâce à un web viewer qui a besoin d'un module d-bus pour compiler. Malgré un nombre important d'essais avec plusieurs versions différentes, le résultat reste inchangé. Nous avons donc décidé de mettre de côté cette aspect, rappelons-le, fonctionnel sur notre machine, car le module rencontre ce problème uniquement sur le raspberry.

Enfin nous avions rencontré un problème avec notre lecteur RFID, car celui-ci avait besoin d'être soudé à une composante, permettant le raccord à la raspberry.
Le module du keyboard dans notre interface graphique ne voulait pas compiler.
Concernant l'hyperplanning une autre méthode a été envisagé grâce au web service et aux techniques de web scraping pour récupérer les données malheureusement le site était vraiment bien protégé et tout était fait en back-end, nous avons donc préféré nous arrêter là et continuer avec la première méthode qui était fonctionnel.