

PELLEGRIN Léo

ROLAND Salomé

GARCION Loïs

(LOPEZ Damien FERREIRA Mélia)

FIE 4

**Rapport de Projet Tutoré :**

**Logiciel de comptage pour la Maison Campus**

**Remerciements**

Nous exprimons nos sincères remerciements à toutes les personnes qui ont contribué à la réalisation de ce projet, ainsi qu'à l'élaboration de ce rapport.

En premier lieu, nous souhaitons adresser nos remerciements à notre tuteur, en particulier à Monsieur Rémi BASTIDE, dont l'accompagnement et les conseils ont été précieux tout au long du projet. Ses recommandations éclairées ont grandement enrichi chaque étape, de la conception initiale jusqu'au développement final. De même, nous exprimons notre profonde gratitude envers notre client, Monsieur Rémy GINESTET, dont la collaboration et la compréhension ont été essentielles à la réussite de notre projet. Sa disponibilité et son soutien nous ont permis de relever les défis avec confiance et détermination.

Nous tenons également à remercier Madame Raja NEJJAR pour son implication dans le financement des capteurs indispensables à la réalisation de notre projet. Sa contribution financière a été d'une importance capitale et nous sommes reconnaissants de son soutien.

Enfin, nous exprimons notre reconnaissance envers certains de nos professeurs, notamment Monsieur Alexis COMBES et Monsieur Adrien PEYROUTY, pour leur enseignement précieux et leurs conseils avisés dans divers domaines, qui ont enrichi nos connaissances et ont été directement applicables à notre projet.

L'ensemble de ces contributions a été d'une importance cruciale pour la réussite de notre projet, et nous tenons à exprimer notre gratitude envers toutes les personnes impliquées.

**Sommaire**

[**1. Présentation du sujet et analyse des besoins 5**](#_k27ley4cldyr)

[1.1 Contexte 5](#_he4ytjzg03os)

[1.2 Périmètre 5](#_96g5w95s514d)

[1.3 Les besoins 5](#_78u4u22xh5t5)

[**2. Phase de conception et de recherche 7**](#_r7cvorj25k2g)

[2.1 Modèle de donnée et Diagrammes 7](#_aj6rubiyk3na)

[2.1.1 Le MCD\* 7](#_bd8phno94uh0)

[2.1.2 Définition des utilisateurs 9](#_u535nscwbhc)

[2.3 Maquettes des Interfaces 10](#_qmh0v03rnb0i)

[**3. Réalisation de la solution matérielle 13**](#_fc5fwd5ke53f)

[3.1 Choix des capteurs 13](#_72t1sb46l038)

[3.2 Financements du projet Tutoré 14](#_7tpyqh23cs37)

[3.3 Test sur les capteurs 15](#_jp56j2kedhe)

[**4. Architecture de l'application 18**](#_f51vdjd0k18)

[4.1 Création de la base de données 18](#_z7t89a3m8x63)

[4.1.1 Réalisation de la base de données 18](#_2krnem4zr8mm)

[4.1.2 Création de procédures 19](#_uwaocvd725un)

[4.2 Réalisation du backend\* 20](#_5ohxr856kmqx)

[4.2.1 Mise en place de l’API\* 20](#_2ihvbpa9ujnp)

[4.2.2 Mise en place de la gestion utilisateur 22](#_nnsw9i3izmhh)

[4.2.3 Création de fausses données 24](#_qx51t7g798um)

[4.3 Réalisation du frontend\* 25](#_u5eazl8yq95y)

[4.3.1 L’architecture de l’interface utilisateur 25](#_x69k3ttuwxnb)

[Réalisation des pages principales 26](#_r7kc73l08lrb)

[Liaison avec l’API\* 30](#_2hwx1j2ziy2k)

[4.3.2 Gestion des utilisateurs 31](#_wn0bea7vrm5i)

[La connexion 31](#_eu0y6ejz34md)

[Récupération des droits des utilisateurs 33](#_qa354xai7xbc)

[**5. Outils et technologies utilisés 34**](#_5af2yh6o1ccm)

[5.1 Structure logicielle 34](#_4ty37ml8sfhv)

[5.1.1 La base de données 34](#_ds2r8d3oxxvz)

[5.1.2 Le backend\* 34](#_nkj9v2s5frtd)

[5.1.3 Le frontend\* 35](#_sb8qlwbo74is)

[5.2 Logiciels et outils 36](#_lyhdh3pvrwek)

[5.3 Organisation et gestion du dépôt Git 41](#_mdbhfp7vhepu)

[**7. Gestion de projet 43**](#_trk6ph8k9pj4)

[7.1 Planification 43](#_xgf0gyhmk8ed)

[7.2 Organisation et Mise en place 44](#_bk85w1imlwpc)

[7.3 Implication du client dans le projet 44](#_jyix0w4meq9q)

[**8. Bilan et perspectives 45**](#_s8zfrqeiumeg)

[8.1 État d’achèvement du projet 45](#_2lqpsut6wvo)

[8.2 Difficultés rencontrées et solutions 45](#_mylzbbntbbxw)

[8.3 Prolongements et ambition du projet 47](#_78dicbedf68h)

[**9. Bibliographie 48**](#_ms73ov5rhps4)

[**10. Annexes 49**](#_m4eodyvmnn8)

[11. Table des figures 55](#_yn0qu2ln5jyd)

[12. Glossaire 56](#_5hik3l6hsrkq)

# 1. Présentation du sujet et analyse des besoins

## 1.1 Contexte

Ouverte en 2019, la Maison Campus regroupe plusieurs équipements : une bibliothèque, une salle multimodale, une salle informatique, des salles de travail en groupe, un espace détente, un Fablab et les bureaux du SMIX\*. Ce Learning center a été lancé par le SMIX\* et est porté par l'INU Champollion, qui en est le propriétaire, et l'IUT Castres. La Maison Campus accueille les étudiants du campus de la Borde Basse : ISIS, IUT, BTS et licences pro du lycée Borde Basse, et a pour but de permettre différentes phases du travail comme la recherche de sources, l'apprentissage, la révision, la pédagogie, la fabrication, la culture et la découverte.

À ce jour, pour connaître et gérer l’affluence de la Maison Campus, le directeur de celle-ci, Rémy Ginestet, passe toutes les deux heures dans chaque salle de la Maison Campus pour compter combien d’étudiants s’y trouvent. Ainsi, la mise en place d’un système de comptage permettra l’automatisation de ce processus.

## 1.2 Périmètre

Après plusieurs réunions avec les différents acteurs, il a été convenu que le projet se concentrera initialement sur l’installation de capteur au niveau des doubles portes d'entrée de la Maison Campus. Cependant, notre approche globale vise à concevoir une solution qui puisse être étendue à l'ensemble de la Maison Campus, tout en étant suffisamment adaptable pour être déployée dans d'autres bâtiments tels que l'IUT Paul Sabatier et le Syndicat Mixte.

En envisageant une perspective d'extension, nous prenons en compte les spécificités des différentes zones de la Maison Campus, tout en veillant à concevoir un projet modulaire. Cette modularité garantira une flexibilité optimale permettant une adaptation facile aux caractéristiques uniques de chaque espace, par exemple la distance de captation des personnes est assez importante et peut-être variable en fonction de la largeur de la porte.

## 1.3 Les besoins

Le projet vise à mettre en place un système de comptage dans les différents espaces de la Maison Campus afin de fournir des statistiques précises sur son affluence. Il se divise en deux parties principales :

**Installation et Configuration des Capteurs Physiques :**

* Installer des capteurs au niveau des doubles portes d'entrée de la Maison Campus.
* Tester les capteurs dans le hall d'entrée pour évaluer la réussite du projet.
* Envisager un déploiement sur l'ensemble de la Maison Campus après évaluation.

**Création d'un Site Web de Statistiques :**

* Développer un site web permettant d'afficher en temps réel le nombre de personnes dans chaque partie de la Maison Campus.
* Fournir des statistiques en fonction de dates et plages horaires personnalisables.
* Permettre l'exportation des données vers des fichiers Excel ou PDF.
* Respecter la charte graphique de la Maison Campus pour maintenir une cohérence et un professionnalisme dans l'utilisation.

Les parties prenantes impliquées dans le projet comprennent :

* L'équipe de PTUT, responsable du développement du projet.
* L’École d'ingénieur ISIS, en charge de la gestion budgétaire du projet.
* La Maison Campus, qui joue le rôle de client et fournit des directives et des exigences spécifiques pour le projet.

Ces objectifs définissent le cadre du projet et servent de guide pour sa planification, son exécution et son évaluation.

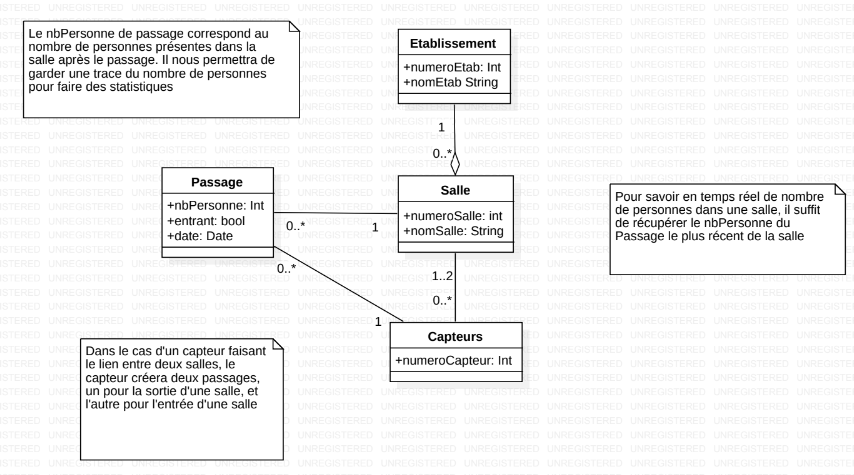
# 2. Phase de conception et de recherche

## 2.1 Modèle de donnée et Diagrammes

### 2.1.1 Le MCD\*

Avant de parler de la solution côté Maison Campus et côté solution web, nous avons essayé de définir le format de la donnée que nous allons utiliser. Cette donnée doit être pratique aussi bien du côté de nos capteurs, que du côté de notre site web.

Pour représenter cette donnée, nous avons réalisé un premier jet d’un diagramme de classe représentant : Les établissements, les salles, les capteurs et leurs passages.



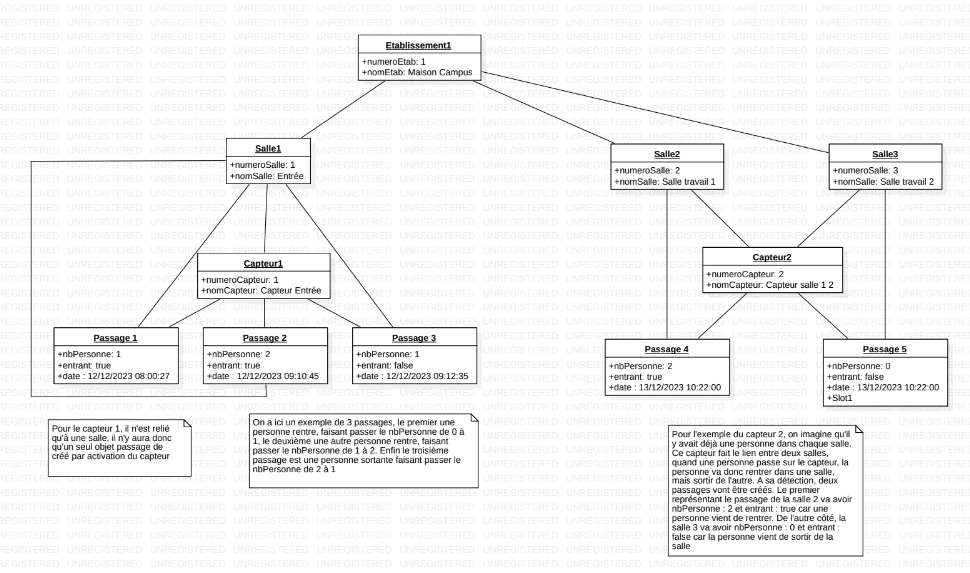
##### Figure 1 : Modèle conceptuel de données

Dans ce diagramme, l’établissement représente la zone entière où seront installés nos capteurs. Les salles sont les zones où nous voulons compter nos personnes. Les capteurs représentent le capteur physique, mais il faut faire attention, car ce capteur peut représenter le lien entre plusieurs salles. Et enfin, un passage qui est lié à la fois à notre capteur et à une salle.

Lorsqu’un capteur faisant le lien entre plusieurs salles va s’activer, deux passages vont être créés. Le premier sera le passage entrant dans une salle, et le deuxième le passage sortant dans une salle.

Pour savoir en temps réel combien de personnes sont présentes dans une salle, il suffira de récupérer le dernier passage lié à cette salle.

Afin de représenter concrètement notre diagramme de classe pour la Maison Campus, nous avons réalisé un diagramme objet.



##### Figure 2 : Exemple d’application du modèle conceptuel de données

Dans ce diagramme objet, nous représentons deux cas de salles. Dans un premier temps, un capteur qui n’est lié qu’à une seule salle, comme le capteur de l’entrée de la Maison Campus. Dans notre exemple, ce capteur génère trois passages, le premier est un passage entrant qui va faire passer le nombre de 0 à 1. Le deuxième est un passage entrant aussi, faisant passer cette fois-ci le nombre de personnes de 1 à 2. Enfin, le dernier passage est un passage sortant, faisant ainsi passer le nombre de personnes de 2 à 1.

Ensuite, nous avons un exemple pour un capteur lié à deux salles différentes. Pour ces deux salles, on part du principe qu’il y a déjà une personne dans chaque salle.

On ne représente ici qu’une seule détection du capteur qui va donc créer deux passages. Le premier passage est un passage entrant pour la salle numéro 2 qui va faire passer le nombre de personnes de 1 à 2. Le deuxième passage créé est lui le passage de la personne sortant de la salle 3 et va ainsi faire passer le nombre de personnes de 1 à 0.

### 2.1.2 Définition des utilisateurs

Dans notre analyse, nous avons besoin d’identifier les différents utilisateurs de notre solution ainsi que les différentes actions qu’ils peuvent réaliser. C’est pourquoi nous avons utilisé le diagramme de cas d’utilisation qui est très utile pour définir simplement et efficacement les utilisateurs et leurs droits.

##### Figure 3 : Use Case

Dans ce diagramme de cas d’utilisation, nous avons défini trois utilisateurs, l’utilisateur non connecté qui n’a accès qu'aux actions de se connecter, ainsi qu'à une option pour changer de mot de passe / retrouver son compte.

Notre deuxième utilisateur est l’utilisateur connecté à un établissement qui pourra consulter les données de l’établissement. Pour consulter ces données, il pourra sélectionner la pièce en question ou sélectionner une plage horaire. Mais cet utilisateur n’aura un rôle que de consultation.

Enfin, nous avons identifié un troisième type d’utilisateur, l’administrateur qui aura comme actions supplémentaires la possibilité de créer des utilisateurs et de sélectionner quel établissement consulter.

Nous avons aussi fait la conception de notre utilisateur en base de données pour définir ses champs.



##### Figure 4 : Diagramme de la classe ApiUser

Dans cette table, le apiUsername correspond au mail de l’utilisateur, le champ apiPasswordHash contient le mot de passe crypté avec un algorithme, apiRole sera soit “admin” soit le numéro d’un établissement. En fonction, l’utilisateur aura accès à toutes les fonctionnalités, ou uniquement celle liée à son établissement associé. Et enfin le booléen passwordChanged, indique si le mot de passe a déjà été modifié ou non. Si le mot de passe n’a pas encore été changé, il devra le faire lors de sa première connexion.

## 2.3 Maquettes des Interfaces

Afin de présenter une interface utilisateur répondant aux besoins des utilisateurs finaux, nous avons tout d'abord étudié le site de la solution déjà en place à la Maison Campus, comme indiqué par M. Ginestet. L'objectif était d'éviter de proposer un site similaire qui pourrait ne pas être apprécié par les utilisateurs. Cette analyse nous a permis de comprendre les erreurs commises dans le passé en termes d'expérience utilisateur, jugée comme trop complexe.

La première difficulté a été de réfléchir à la page d’accueil du site de statistiques afin d’avoir un maximum d’informations sans qu’elles soient trop complexes ou dénuées de sens. Dans un premier temps, nous avons défini les informations les plus importantes selon nous :

* Le nombre de personnes :
  + Dans la globalité
  + Dans la bibliothèque
  + Somme des salles de travail
* Une vision plus ludique avec une carte interactive de la Maison Campus
* Des graphiques sur la journée en fonction des horaires
  + Occupation globale
  + Occupation détaillée

##### Figure 5 : Maquette de la page d'accueil

Une deuxième page a été imaginée pour présenter des informations personnalisables en fonction de la saisie de l’utilisateur en fonction des salles, de la date et des horaires.

##### Figure 6 : Maquette de la page d’historique personnalisable

La dernière partie a été de réfléchir sur l’interface utilisateur d’export des informations du site vers un fichier Excel ou PDF. Nous avons décidé d’ajouter cette fonctionnalité dans la page d’informations personnalisables afin de pouvoir à la fois visionner les informations, mais aussi les exporter sous une autre forme pour diverses utilisations.

Pour conclure, cette maquette n’a pas encore été présentée au client, M. Ginestet. Elle est donc sujette à de nombreux changements en fonction des différents retours. De plus, notre client nous a confirmé qu’il la montrerait à différentes personnes afin d’avoir plus de retour et améliorer l’expérience utilisateur au maximum. Il faut aussi préciser que le logiciel utilisé pour réaliser la maquette interactive est une version gratuite qui ne permet pas de mettre en œuvre la totalité de l’expérience utilisateur comme voulu. Par conséquent, des fonctionnalités pourront être présentes dans la solution finale qui ne le sont pas dans la maquette

Vous pourrez trouver la maquette réalisée ci-dessous : (Il faut prendre en compte qu’elle peut évoluer en fonction de la date de consultation).

Lien:[https://www.figma.com/proto/tjBo8nlBvx3ds6JICuGWyK/Maquette-Site-Stats-PTUT?page-id=0%3A1&type=design&node-id=1-2&viewport=-608%2C1144%2C0.6&t=gV9n71t8RoTVPSgL-1&scaling=min-zoom&starting-point-node-id=1%3A2&mode=desig](https://www.figma.com/proto/tjBo8nlBvx3ds6JICuGWyK/Maquette-Site-Stats-PTUT?page-id=0%3A1&type=design&node-id=1-2&viewport=-608%2C1144%2C0.6&t=gV9n71t8RoTVPSgL-1&scaling=min-zoom&starting-point-node-id=1%3A2&mode=design)

# 3. Réalisation de la solution matérielle

## 3.1 Choix des capteurs

Il existe sur le marché différentes solutions de comptage proposées par des entreprises, ces entreprises utilisent différentes technologies. Nous en avons trouvé cinq existantes et adaptables à notre échelle.

Premièrement le capteur infrarouge passif et actif. Les passifs sont des capteurs électroniques qui mesurent la lumière infrarouge. Ils sont généralement utilisés dans des caméras de détection de mouvement, car ils permettent de détecter la présence de tout objet émettant de la lumière infrarouge passant devant le capteur. Les capteurs infrarouge actif, eux, sont des capteurs similaires, mais ils vont aussi émettre un rayonnement infrarouge, et capter le rayonnement infrarouge réfléchi par l’objet. Ces capteurs ont une faible consommation *(≤ 11 µA)* et ont une portée maximale de 5 mètres. Ils sont peu coûteux et sont simples d’utilisation. Cependant, ces capteurs infrarouges ne s’activent que lors de la présence d’un objet émettant ou réfléchissant la lumière infrarouge, il n'est donc pas assez précis dans le cas où plusieurs personnes entrent en même temps. Il ne permet pas non plus de détecter le sens du mouvement, excepté si nous utilisons deux capteurs, mais la précision sera encore réduite.

Ensuite, il existe le capteur à faisceau de rupture *(ou break beam sensor)\** fonctionnant via un émetteur et un récepteur. L’émetteur va continuellement émettre un faisceau qui va être récupéré par le récepteur, si jamais ce faisceau n’atteint plus le récepteur, alors c’est qu’il a été coupé et qu’une personne est passée. Ce type de capteur est précis et localisé, il détecte sur un espace très fin, on peut donc en poser plusieurs à une distance réduite pour détecter le sens de la personne.

Une technologie utilisable est l’onde millimétrique. En effet, les ondes millimétriques sont les ondes radioélectriques couvrant les fréquences de 30 à 300 GHz. Ces capteurs ont une précision accrue et ont une capacité de détection grande portée. Notons également qu’ils sont capables de détecter lorsque plusieurs personnes entrent simultanément. Cependant, ces capteurs utilisent un phénomène plus complexe que les autres, entraînant une complexité d’installation et d’utilisation. Un temps plus conséquent sera donné au niveau du code afin d’avoir une précision acceptable. Malgré sa grande précision, les ondes millimétriques peuvent être perturbées en fonction de la densité de bruits ou d’autres ondes radios présentes dans la Maison Campus, ce qui entraînerait quelques erreurs de mesures. De plus, sa consommation est d’environ 90 mA, il faudra donc prévoir une optimisation de la batterie au moment du montage du capteur.

Par la suite de nos investigations, nous avons exploré le concept du Time of Flight\*, exploitant des zones d’ondes infrarouges. Ce principe novateur est mis en œuvre au moyen de capteurs spécifiques qui discernent avec précision le passage d’une personne à travers chaque zone. La variation dans la détection au sein de chaque zone permet alors de déterminer de manière astucieuse le sens de la circulation. Cette approche offre une méthode sophistiquée et technologiquement avancée pour la surveillance des déplacements, s'avérant particulièrement adaptée à notre objectif d'analyse du flux de personnes dans l'environnement de la Maison Campus.

Finalement, une technologie un peu différente des autres est l’utilisation de caméras. Celles-ci peuvent compter précisément le flux de personnes sans aucun ajout de carte électronique supplémentaire. Les caméras sont très efficaces, fiables sur le long terme et sont plus difficiles à tromper, les résultats seront donc plus précis. Leur capacité de collecte de données est bien plus avancée que pour les autres capteurs en termes de vitesse et de sens. La surveillance à distance reste également une possibilité. Par ailleurs, l’utilisation de ce type de technologie nécessite une importante alimentation électrique, une importante bande passante. Il faudrait donc qu’elles soient directement reliées au réseau électrique de l’établissement et que nous ayons un serveur dédié pour pouvoir réaliser et analyser les vidéos en tant réel. Ces deux contraintes peuvent engager des travaux au sein de la Maison Campus.

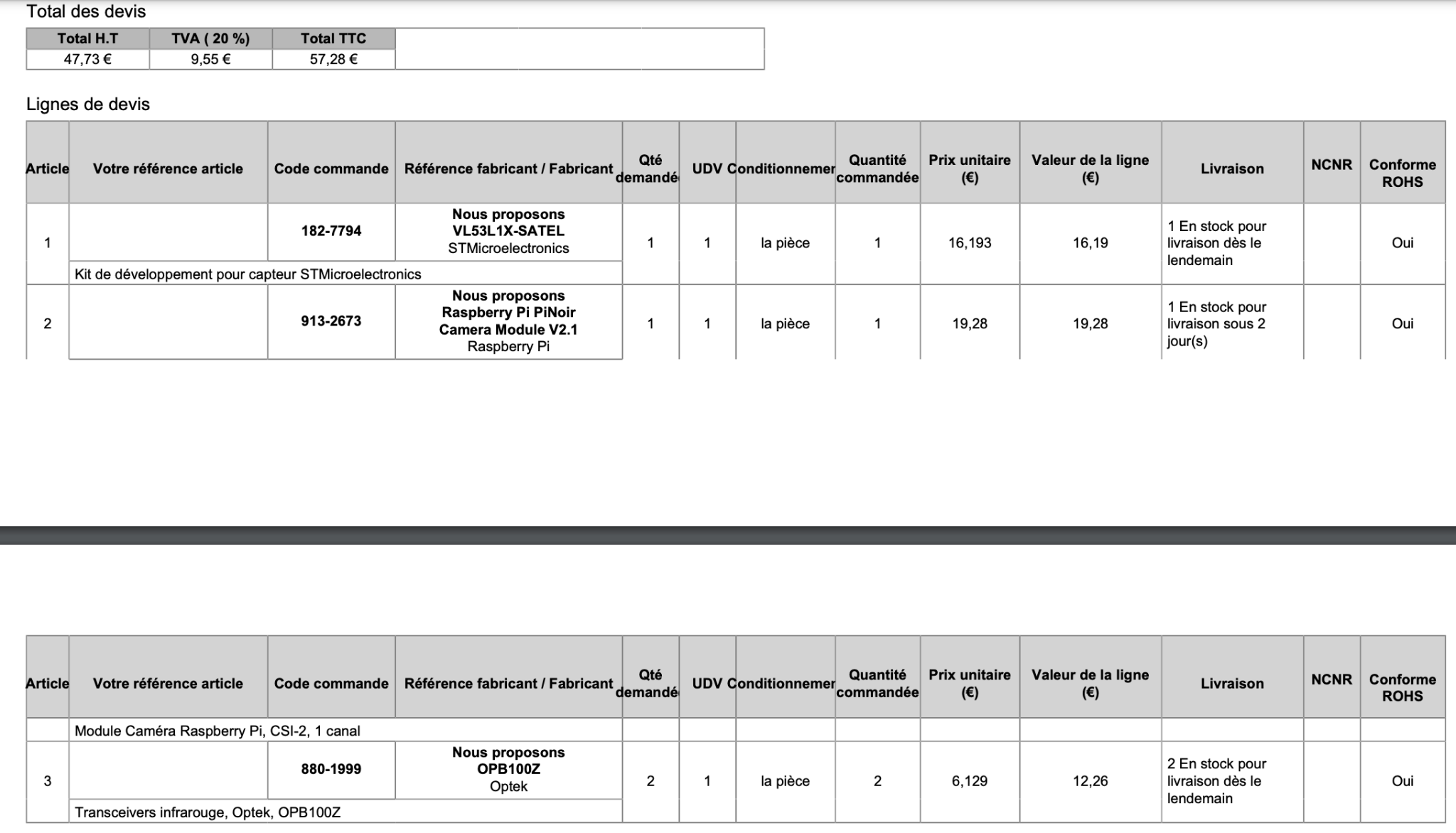
Pour conclure, pour chaque type de capteur, nous avons réalisé une analyse financière en plus de celle technique. Cette dernière ne prend pas en compte uniquement le prix du capteur, car il doit souvent être branché sur une carte électronique. De plus, des modules supplémentaires afin de connecter la carte avec un câble Ethernet peuvent être nécessaire en fonction des modèles. Cela nous a donc permis de comparer les avantages et les inconvénients avec une notion financière de chaque capteur.

## 3.2 Financements du projet Tutoré

Suite à notre réunion avec Mme NEJJAR, la responsable financière de l'école ISIS, notre objectif était d'obtenir des informations cruciales sur le budget attribué à notre projet. Initialement, nous étions dépourvus de directives claires concernant la démarche à suivre pour soumettre une demande de financement, et l'existence même d'un budget dédié à notre initiative était floue. Ajoutant à la complexité, nous avions une contrainte de temps pressante, car l'école était en plein processus de clôture budgétaire annuelle.

Une autre exigence importante était que le devis devait être établi auprès d'un fournisseur partenaire de l'école, simplifiant ainsi les transactions. Notre enveloppe budgétaire pour la phase de test était fixée à 200€, et nous avons pris des mesures minutieuses pour nous assurer que le devis proposé respectait scrupuleusement cette limite.

Suite à des recherches approfondies, nous avons minutieusement sélectionné les capteurs qui semblaient être les plus adaptés à notre problématique. Par la suite, nous avons présenté notre devis à Mme NEJJAR, dans l'espoir de recevoir son aval et de progresser dans la concrétisation de notre projet.



##### Figure 7 : Devis des capteurs pour la phase de test

## 3.3 Test sur les capteurs

La suite logique de cette partie du projet a donc été de mettre en place et de tester les capteurs. Comme il est indiqué dans le devis (Figure 6), nous avons choisi d’en commander trois : une caméra, des capteurs à faisceau de ruptures\* et un capteur de type Time of Flight\*.

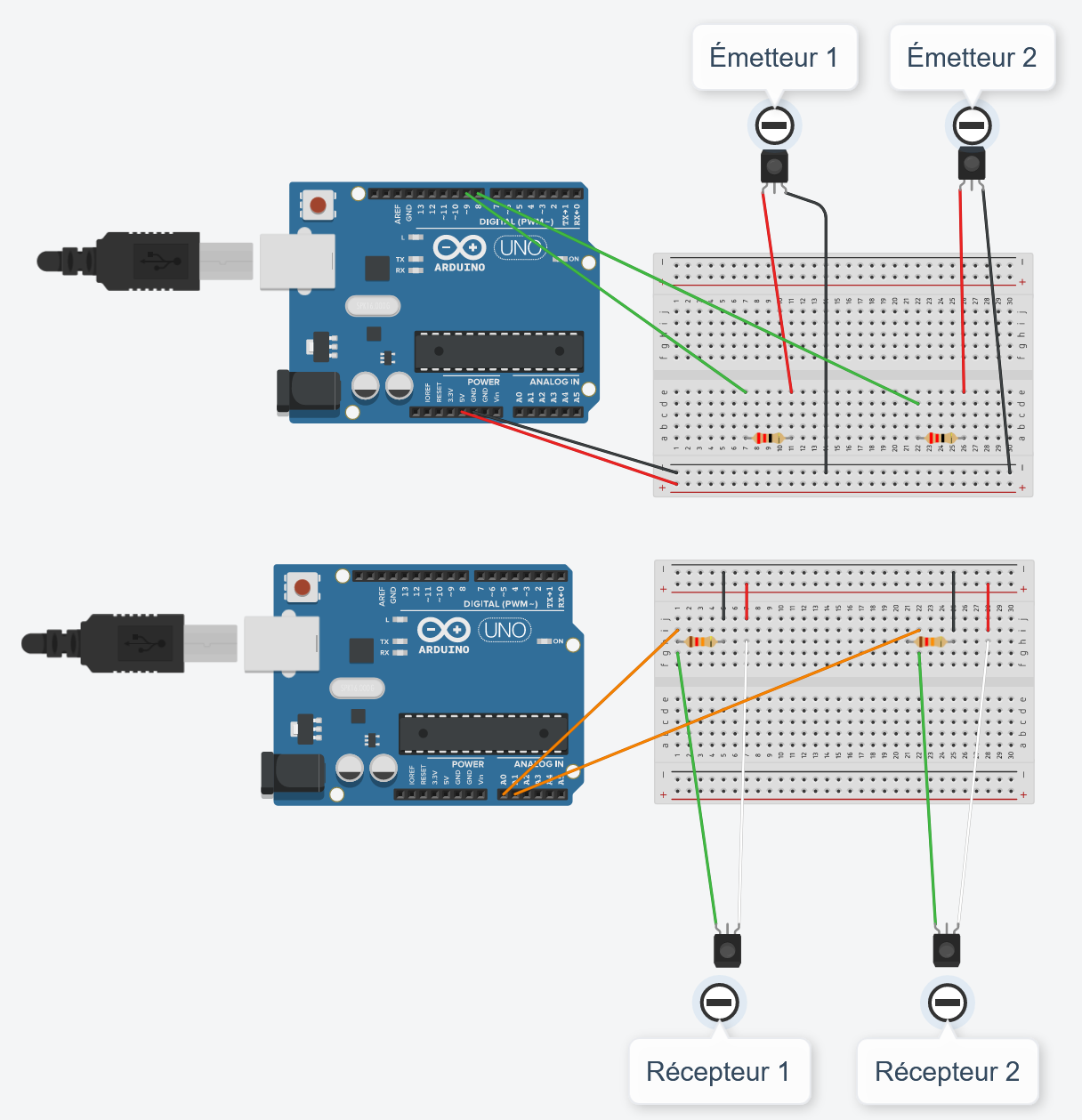
Pour réaliser un tel montage et pouvoir exploiter ses capteurs, il est nécessaire d’utiliser un micro-contrôleur\*. Au début du projet, nous pensions utiliser une carte de type “Raspberry Pi\*” mais nous avons finalement pris la décision de changer et avons choisi une carte “Arduino Uno\*” car c’est le modèle avec lequel nous avions l’habitude de travailler.

De ce fait, nous n’avons pas pu utiliser la caméra parce que cette dernière n’est compatible qu’avec une carte Raspberry pi\*.

Les capteurs de type time of flight\* ont été testés, mais nous avons rencontré des problèmes expliqués plus en détails dans la partie [3.3.1 Problèmes rencontrés](#_555d05d1nqca).

Nous avons donc choisi d’approfondir nos tests sur les capteurs à faisceau de ruptures (break beam\*).

Il a fallu, en premier lieu, réaliser le montage des capteurs comme représenté ci-dessous.



##### Figure 8 : Schéma du montage des capteurs à faisceau de rupture\*

Nous avons fait le choix d’utiliser deux capteurs afin de reconnaître s’il s’agit d’une entrée ou d’une sortie selon le principe suivant :

* Une entrée est enregistrée si le faisceau un est rompu en premier puis le faisceau 2 ensuite
* Une sortie est enregistrée si c’est le faisceau deux qui est rompu en premier, puis le faisceau 1.

Cette configuration nous permet donc de connaître le sens dans lequel la personne a franchi la porte.

Au niveau de l’alimentation, les émetteurs nécessitent d’être branchés uniquement sur le secteur, car ils émettent un faisceau infrarouge en continu et ne renvoie pas d'informations.

Les récepteurs sont, quant à eux, reliés à l'ordinateur afin que l’algorithme puisse traiter les informations qu’ils transmettent.

### 

# 4. Architecture de l'application

L'application est structurée en trois parties principales : la base de données, le frontend\* et le backend\*. La base de données définit le modèle de données stockées en son sein, avec ici un système basé sur Postgres SQL\*. Le backend\*, développé avec le framework\* Node.js, fournit une API\* pour établir le lien entre la base de données et l'interface utilisateur. La dernière partie, destinée à l'utilisateur final, est le frontend\*. Nous avons opté pour le framework\* Vue.js pour sa capacité à créer des interfaces utilisateur Web. Cette section est connectée au backend\* pour permettre l'interrogation des données requises.

## 4.1 Création de la base de données

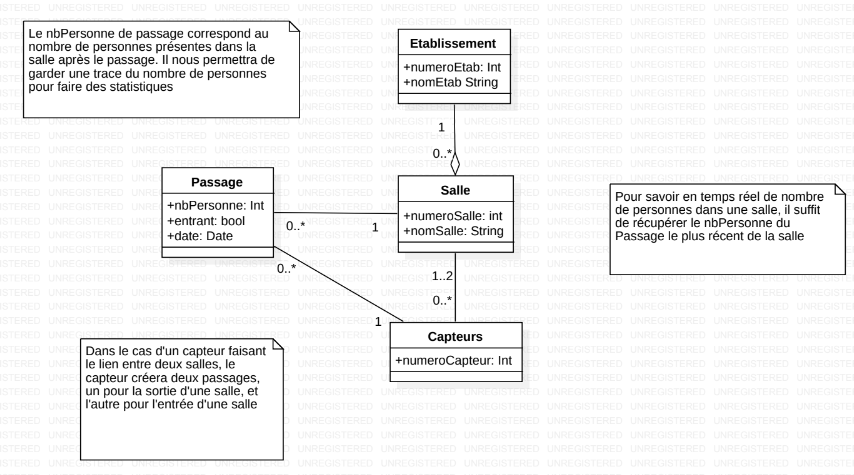
Dans notre projet, nous devons stocker des informations sur les passages dans la maison Campus. Afin de stocker ces informations, nous utilisons une base de données relationnelle dont nous allons expliquer le résultat.

Dans une base de données, on utilise des tables qui représentent des objets de données. Ces tables possèdent des variables que l’on va remplir lors d’ajouts de lignes dans cette base de données.

Ces lignes pourront ensuite être récupérées grâce à des requêtes dans le langage SQL\*, qui nous permettra de sélectionner quelles données doivent être renvoyées, et en fonction de quels critères.

### 4.1.1 Réalisation de la base de données

Dans la partie de conception de la base de données, nous avions défini le schéma suivant :



##### Figure 9 : Modèle conceptuel de données

La suite consiste donc en la création de ces tables et de leurs liens dans la base de données.

Le code SQL\* permettant de créer la donnée est assez long, vous pourrez donc le retrouver en annexe.

Il faut ensuite initialiser la donnée de ces différentes tables, pour ce faire, nous devons créer un établissement, lui associer ses salles, ses capteurs, et enfin ses passages qui vont initialiser le nombre de personnes dans la salle à 0.

Encore une fois, ce code de génération de la donnée est assez long, et sera, lui aussi, présent en annexe.

### 4.1.2 Création de procédures

Dans notre schéma de données, nous avons réfléchi à un système particulier, lorsqu’une personne rentre dans une salle, un passage est généré avec comme valeur pour le nbPersonne, la valeur du nbPersonne précédent + 1. De même, quand une personne sort d’une salle, la valeur du nouveau nbPersonne doit être égal à la valeur du nbPersonne précédent - 1. Dans le cas où deux salles sont connectées, la sortie d’une personne est liée à l’entrée d’une personne.

Pour gérer tous ces cas, nous avons pris la décision d’utiliser une procédure stockée PL/pgSQL\*. Ce langage de script permet de créer des fonctions ou des procédures qui vont exécuter un code selon des paramètres, et que nous pouvons ensuite appeler très simplement.

Ce script étant assez long, vous pourrez le retrouver en annexe. Nous allons tout de même ici présenter son fonctionnement :

On retrouve en paramètre deux identifiants, l’identifiant de la salle de sortie, et l’identifiant de la salle d’entrée. Dans le cas où l’un des deux est vide, nous ne créerons qu’un passage, mais dans le cas où les deux sont remplis, nous allons créer deux passages.

Nous allons ensuite récupérer la valeur nbPersonne du dernier passage de ces salles, et créer un nouveau nbPassage dont la valeur du nbPersonne sera modifié en fonction de l’entrée ou la sortie. La date de ce nouveau passage sera sa date au moment de sa création.

## 4.2 Réalisation du backend\*

La partie backend\* de ce projet est très importante, elle permet de faire le lien entre la base de données, les capteurs et l’application web. Elle doit envoyer à l’application web de la donnée dans un format adapté et elle doit permettre la création de passage depuis les capteurs, le tout de manière sécurisé.

### 4.2.1 Mise en place de l’API\*

Une API\* c’est une interface logicielle qui permet de connecter un logiciel ou un service à un autre logiciel ou service pour transmettre de la donnée. Dans le cadre de notre projet, tout ce qui doit être récupéré ou crée dans la base de données passera par l’API\*.

Cet API\* a deux objectifs principaux :

* La création de routes spécifiques doivent permettre de récupérer de la donnée spécifique, dans une forme pratique, mais aussi de ne permettre d’accéder qu’aux données que nous avons choisies.
* Cette API\* est le seul accès disponible à la base de données, et nous permet donc de choisir qui peut accéder à quelles informations.

Pour créer cet API\*, il faut d’abord définir un port sur lequel notre API\* va écouter. Nous avons choisi le port 8080, car il est souvent disponible par défaut et est moins susceptible d'être bloqué par les pare-feu que d'autres ports comme le 80 (HTTP\*).

Il faut ensuite définir les CORS\* de notre API\*. Les CORS\* (Cross-Origin Resource Sharing) sont une mesure de sécurité qui permet de contrôler les demandes HTTP\*. Le principe est qu’en cas de requête sur notre API\* depuis une autre application web, le navigateur va bloquer cette requête pour des raisons de sécurité. C’est donc dans le code que nous devons spécifier quelles adresses sont autorisés. Ici, c'est assez simple, nous autorisons les deux adresses que nous utilisons pour le site web : l’adresse locale et l’adresse publique.

const corsOptions = {

origin: ['https://ptut-front-leocorp.koyeb.app', 'http://localhost:8000'],

optionsSuccessStatus: 200

};

##### Figure 10 : Autorisations CORS\* API\*

Une fois toutes les options définies, nous pouvons maintenant passer à l’écriture des routes.

Afin de réaliser des requêtes vers notre base de données, nous utilisons la librairie Postgres de Node.js, il nous suffit ensuite de créer un lien vers la base de données, de réaliser notre requête, de récupérer le résultat ou l’erreur.

Nous devons ensuite imaginer le plan de notre API\*, nous avons décidé ici d’avoir deux types de routes :

* Des routes de type GET pour récupérer de la donnée, dans ces requêtes, nous pourrons définir dans l’URL des informations comme le numéro de bâtiment, de salle…
* Des routes de type POST pour créer de la donnée, ces routes serviront surtout pour les capteurs pour ajouter des passages.

On peut ensuite réfléchir et imaginer ce que le site web va utiliser comme données, et sous quel format.

Les données principales que le site web va utiliser sont :

* Les informations d’un établissement.
* Les informations des salles d’un établissement.
* Les informations des passages des salles.
* Le dernier passage d’une salle.

Les routes GET principales créées sont donc les suivantes :

* "/etablissement" : permet de récupérer les informations de tous les établissements.
* "/etablissement/:idEtab" : permet de récupérer les informations de l’établissement au numéro “:idEtab”.
* "/etablissement/:idEtab/salle" : permet de récupérer les informations de toutes les salles de l’établissement au numéro “:idEtab”.
* "/etablissement/:idEtab/passage" : permet de récupérer tous les passages de l’établissement au numéro “:idEtab”.
* "/etablissement/:idEtab/lastpassage” : permet de récupérer le dernier passage de l’établissement au numéro “:idEtab”.
* "/etablissement/:idEtab/passage/periode" : permet de récupérer la liste des passages dont la date se situe entre deux dates.

Pour cette dernière requête, voila un exemple d’appel de cette route : etablissement/1/passage/periode?dateDebut=2024-02-10T16:56:05.368Z&dateFin=2024-03-20T16:57:30.820Z

Pour les requêtes POST, nous n’en avons qu’une seule :

* '/createpassage' : cette route va appeler la procédure que nous avons présentée précédemment, il suffit de définir le numéro de la salle d’entrée et le numéro de la salle de sortie dans le corps de la requête.

L’API\* maintenant crée, notre site web pourra récupérer les informations qui lui sont nécessaires et les capteurs pourront créer les passages.

### 4.2.2 Mise en place de la gestion utilisateur

Notre API\* est fonctionnelle, mais il faut désormais la sécuriser pour empêcher n’importe qui d’accéder à nos données.

Afin de spécifier qui peut accéder à quelle donnée, nous allons ajouter la notion d’utilisateur.

Pour cela, nous allons créer une table pour les utilisateurs dans notre base de données selon la conception précédente.



##### Figure 11 : Modèle conceptuel de données des utilisateurs

Une fois cette table crée, il faut définir le fonctionnement de notre utilisateur, comment le créer, comment le connecter, comment vérifier sa connexion, etc.

Tout d’abord, nous avons décidé d’avoir deux rôles, soit l’utilisateur est un administrateur, et il aura donc accès à toutes les fonctionnalités et à tous les établissements. Soit l’utilisateur possède comme rôle un numéro d’établissement

Nous avons pris la décision d’utiliser le format suivant. Un utilisateur ne peut être créé que par un administrateur, l’administrateur entre son mail, et le nouvel utilisateur recevra par mail un mot de passe temporaire pour sa première connexion. Lors de sa première connexion, l’utilisateur devra définir un nouveau mail qu’il utilisera par la suite. En cas d’oubli de mot de passe, l’utilisateur pourra réinitialiser son mot de passe, il recevra alors un mail avec un nouveau mot de passe temporaire, et il devra encore une fois changer son mot de passe lors de sa première connexion.

Pour l’envoi de mail, nous utilisons la librairie nodemailer, ainsi qu’un compte Google avec un mot de passe d’application. Cela nous permettra de directement envoyer des mails depuis l’application sans passer par un serveur mail.

Avant de réaliser les routes POST dans l’API, il faut d’abord mettre en place les différentes options de sécurité que nous allons utiliser.

Pour le cas des mots de passe, une des règles de sécurité importante est de ne jamais stocker les mots de passe en dur dans la base. En cas de fuite de données, ou d’une intrusion, les mots de passe ne doivent pas pouvoir être récupérés et utilisés à de mauvaises intentions. Pour pallier ce problème, nous allons utiliser un algorithme de chiffrement. Le fonctionnement est le suivant. Avant de stocker les mots de passe dans la base de données, nous allons appliquer un hachage sur ce mot de passe à l’aide d’un algorithme du nom de Bcrypt. Bcrypt, lorsqu’il est utilisé sur la même chaine de caractère, renverra la même chaine hachée. Lors d’une connexion, on vérifie donc la concordance du mot de passe entrée avec celui dans la base de données.

Pour gérer les sessions utilisateurs du côté du site web, on utilise des jetons d’identifications, le fonctionnement est le suivant. Lors de la connexion de l’utilisateur, on va générer un jeton jwt\* (json web token) à partir des informations de l’utilisateur, d’une clé secrète et d’une durée de vie. Lorsque l’utilisateur se connecte, on lui envoie ce jeton, qu’il va ensuite utiliser lors de ses requêtes. Sur nos routes API\*, on va ensuite vérifier ce jeton et ses informations comme le rôle pour notre réponse.

On peut maintenant générer nos routes POST pour chacune de nos fonctionnalités :

* '/user/register' : cette route permet pour un utilisateur administrateur de créer un compte à un mail donné, elle va insérer en base un nouveau Apiuser avec un mot de passe généré et lui envoyer un mail.
* '/user/login' : cette route permet à un utilisateur en entrant son username et son mot de passe de se connecter. En cas de première connexion, l’API\* ne va pas renvoyer de jeton de connexion, mais va informer que l’utilisateur doit changer de mot de passe. Dans le cas contraire, l’API\* envoie un jeton d’identification avec les informations de l’utilisateur.
* '/user/resetpassword' : cette route va réinitialiser le mot de passe en base pour un username donné, et envoyer un mail avec le nouveau de passe généré. Lors de la première connexion, ce mot de passe devra être changé.
* '/user/changepassword' : cette route est ce que l’on va utiliser lors de la première connexion, ou après une réinitialisation de password. On donne dans le corps l’ancien mot de passe et le nouveau mot de passe. Si le mot de passe concorde avec celui en base, on insère en base le nouveau mot de passe haché.

### 4.2.3 Création de fausses données

Nous n’avons pas eu l’occasion de tester toute notre application avec les capteurs, il nous a donc fallu générer de la fausse donnée. La donnée que nous avons générée ne pouvait pas être aussi fidèle que dans la réalité, mais nous avons quand même essayé de respecter un certain nombre de contraintes :

* Les dates doivent se situer entre 8 heures du matin et 19 heures le soir
* Le soir à 19 heures, tout le monde doit sortir pour n’avoir personne la nuit
* Les passages doivent être sur une grande période de temps

Afin de réaliser cette création de données, nous avons dû modifier la procédure d’ajout de passage en base de données. Pour l’instant cette procédure ne permet de spécifier que le passage d’entrée et le passage de sortie, mais pas la date. Nous n’allons pas faire un script qui génère de la donnée sur trois mois en le laissant lancer pendant trois mois. Nous avons donc créé une nouvelle procédure basée sur celle déjà existante et qui nous permet de spécifier la date.

Il suffit donc de réaliser un script qui va sur une longue période, générer des données de 8 heures à 19 heures, avant de faire des passages de sortie pour tout le monde.

Vous pourrez retrouver ce script sur GitHub avec le backend\*.

## 4.3 Réalisation du frontend\*

Avant de comprendre l’architecture de l’interface, il faut comprendre comment s’organise un fichier de programmation avec le framework\* Vue. Il se décompose en deux parties : La première est le template\* qui permet d’afficher des éléments graphiques, appelée “template\*” en fonction du besoin et des informations. La deuxième partie est un programme appelé “script” permettant de définir les informations à afficher et appliquer un traitement sur ces dernières.

### 4.3.1 L’architecture de l’interface utilisateur

​​L'architecture d'une application Vue.js avec Vue Router est conçue pour rendre la navigation entre différentes parties de l'application simple et fluide. De plus, nous avons utilisé une librairie complémentaire à ce framework\* nommé Vuetify permet d’avoir des composants graphiques comme des formulaires ou des entêtes d’application déjà définis et prêts à l’usage.

La structure de base est contenue dans le dossier “src” :

* C'est là que se trouve tout le code de votre application Vue.js. Vous trouverez principalement :
  + Composants : Ces fichiers contiennent des morceaux de code réutilisables, comme des boutons.
  + Views : Ces fichiers définissent les différentes pages de notre application, comme la page d'accueil ou la page historique dans notre cas.
  + Router : Dans ce dossier, vous configurez la navigation de votre application.
  + App.vue : Ce fichier est comme la maison de votre application, c'est le point de départ.
  + main.js : C'est ici que votre application Vue.js est initialisée.

Le Vue routeur est une bibliothèque qui gère la navigation entre les différentes vues de votre application. On configure les différentes routes de votre application dans le fichier index.js situé dans le dossier dédié au routeur. Voici un aperçu des routes présentes dans notre application :



##### Figure 12 : Chemins définis dans le router Vue

Chaque route correspond à une vue spécifique. Ici, on retrouve la page de connexion comme vue par défaut, mais aussi l’affichage de la page d’accueil, de l’historique ou encore du profil de l’utilisateur connecté.

#### Réalisation des pages principales

Les deux pages principales contiennent principalement des graphiques permettant d’afficher les données de manière plus lisible. Pour cela, nous avons utilisé une librairie extérieure appelée ApexChart. Cependant, à ce moment du développement, nous ne pouvions pas encore nous connecter à l’API\* afin de récupérer les informations. Nous avons donc simulé des données en fonction des attentes des graphiques de la librairie précédemment citée.

Pour la page d’accueil, nous avions besoin d’afficher dans un premier temps l’occupation actuelle.

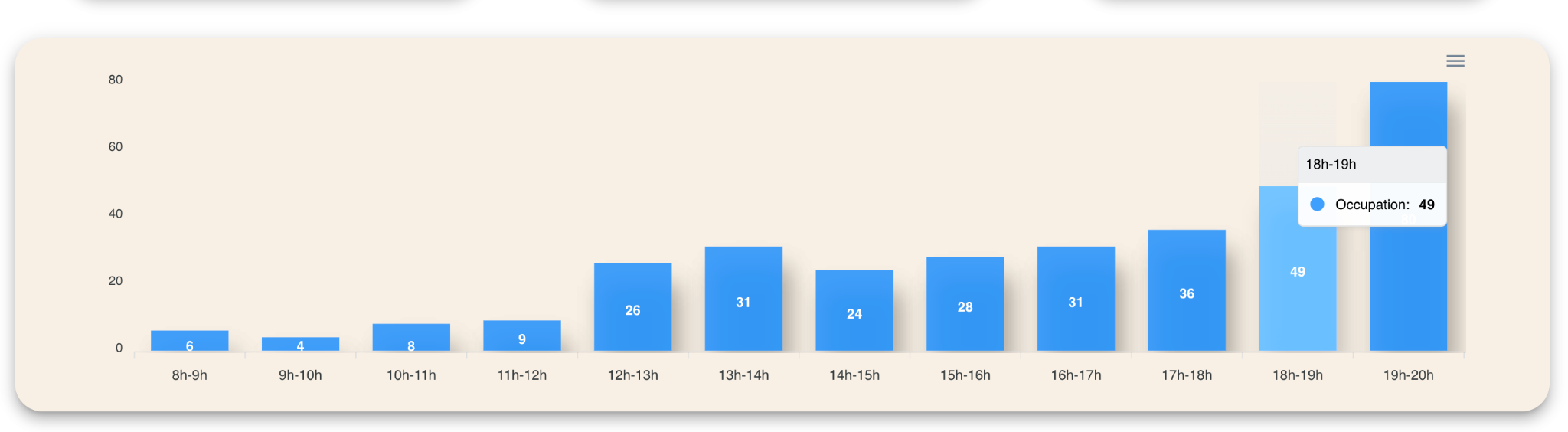


##### Figure 13 : Affichage des occupations globales

Après consultation avec le client, nous avons décidé de séparer cette occupation en deux parties correspondant aux différentes espaces de la Maison Campus :

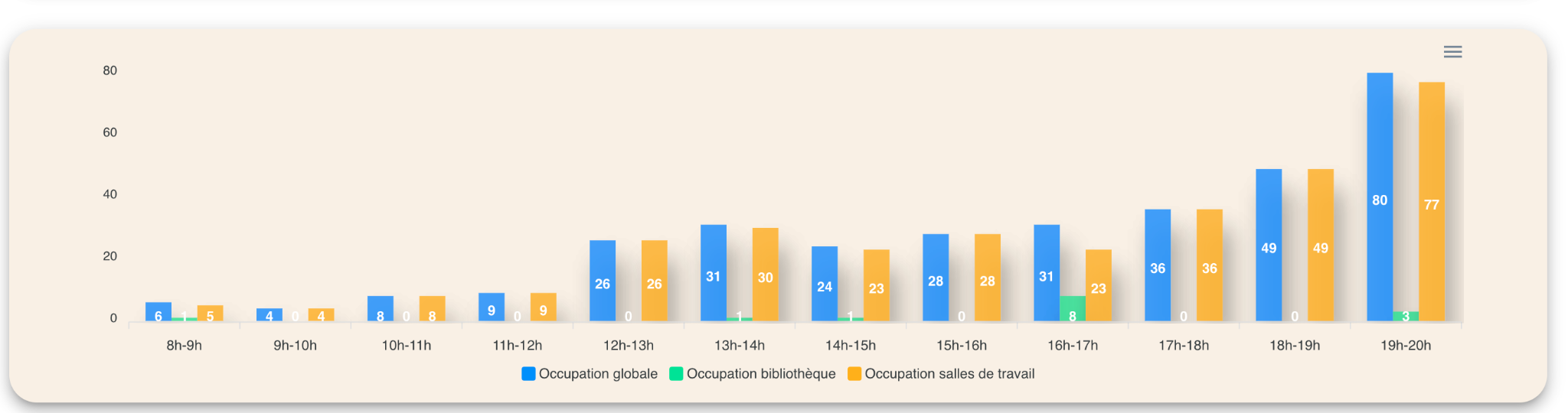
* La bibliothèque
* Les salles de travail

Un premier graphique affiche l’évolution de la fréquentation globale par tranche horaire de 8h à 20h :



##### Figure 14 : Graphique d’occupation globale

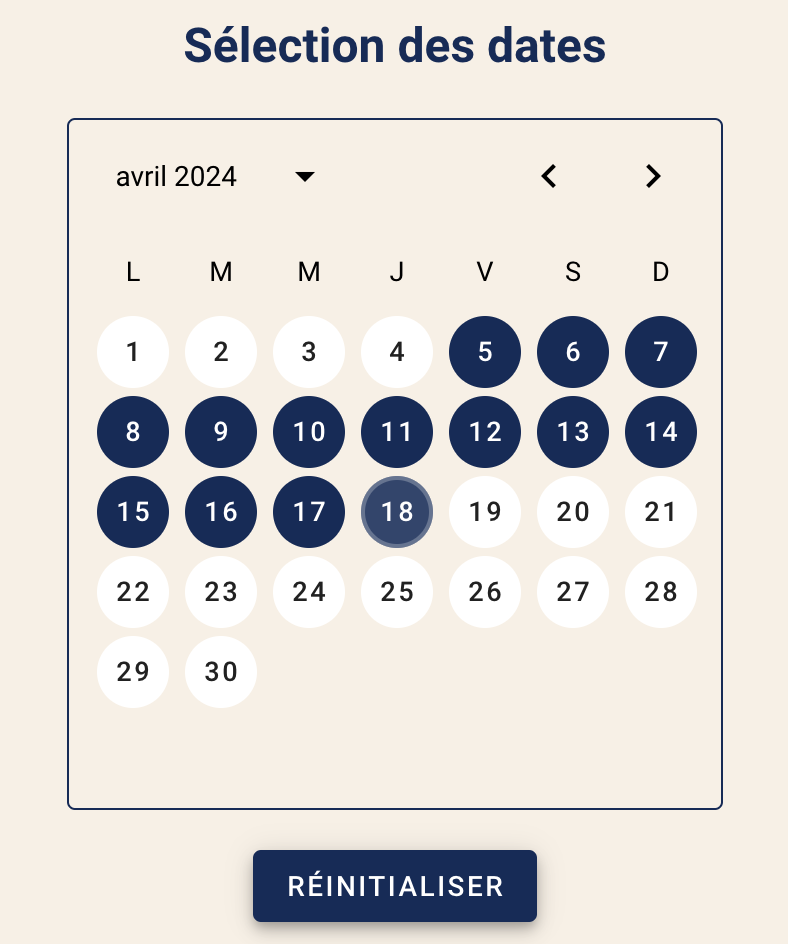
Le deuxième graphique permet d’obtenir les mêmes informations, mais permet de consulter l’évolution en fonction des différents espaces :



##### Figure 15 : Graphique occupation bibliothèque et salles de travail

En résumé, la page d'accueil fournit à l'utilisateur toutes les informations essentielles concernant la fréquentation quotidienne et son évolution tout au long de la journée.

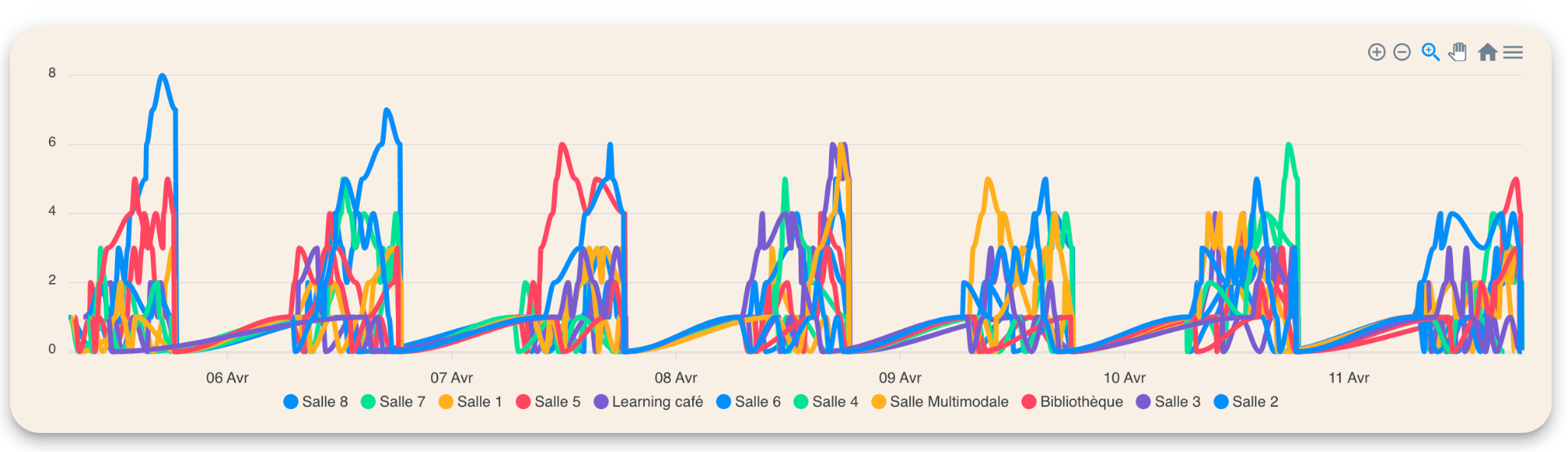
Pour la page de consultation de l’historique, le choix de personnalisation d’affichage des données est plus large pour l’utilisateur. Comme son nom l’indique, un aspect temporel peut être sélectionné à travers le menu de sélection de date :



##### Figure 16 : Menu de sélection de la plage de date

Dans cette interface, l'utilisateur a la possibilité de choisir une date spécifique ou une plage de dates pour consulter les données. De plus, deux graphiques sont proposés pour afficher l'historique des données.

Le premier graphique, illustré par une courbe, offre une visualisation de l'évolution des données sur la période choisie. Il est également interactif, offrant à l'utilisateur la possibilité de zoomer et de dézoomer sur les plages de temps désirées. Cette fonctionnalité ajoute une dimension ludique et rend la consultation des données plus intuitive.

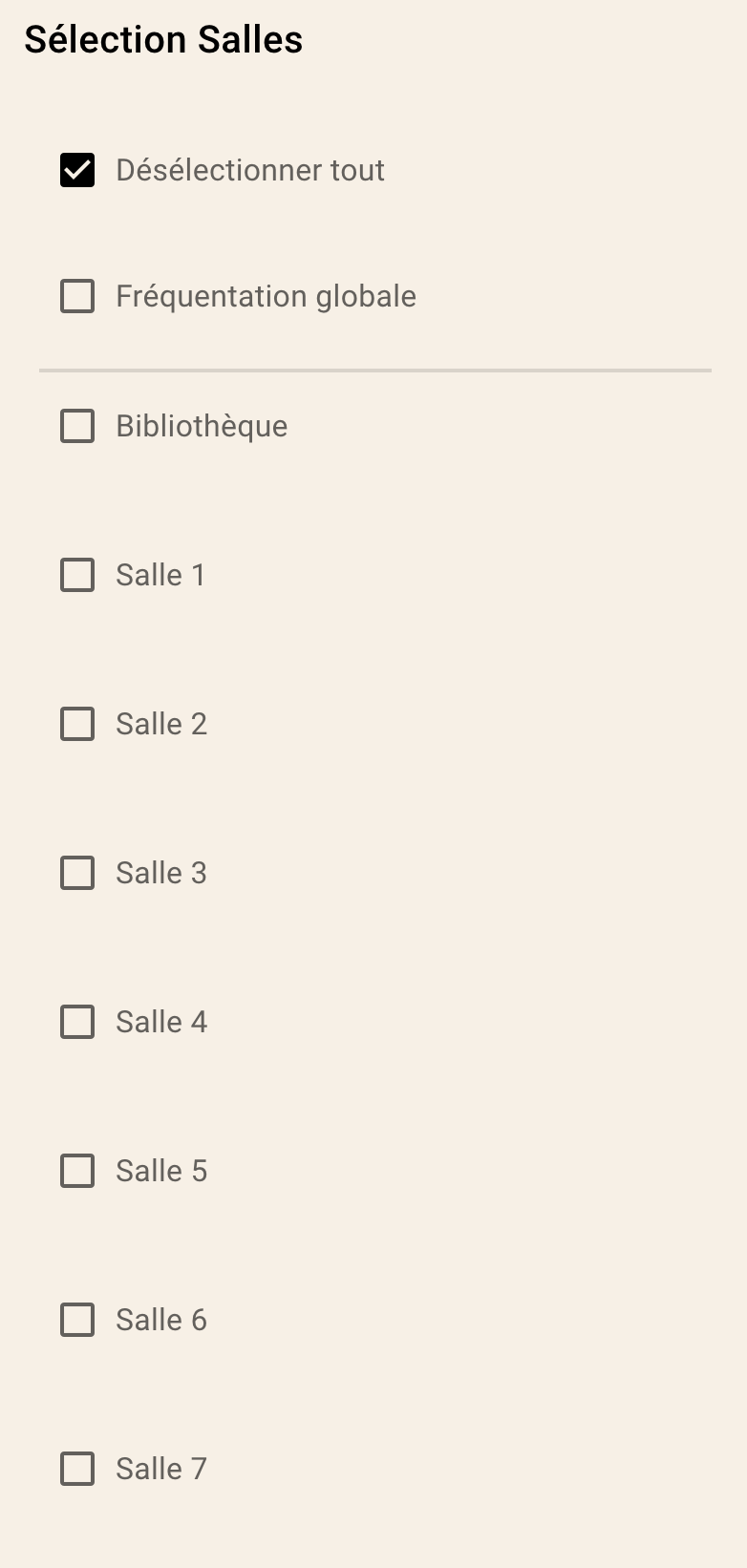


##### Figure 17 : Graphique à courbe de l’évolution de l’occupation

On peut voir sur la [figure 17](#_hzcpfda1oox2) que plusieurs courbes se superposent. Elles représentent chacune une partie plus détaillée de la maison Campus :

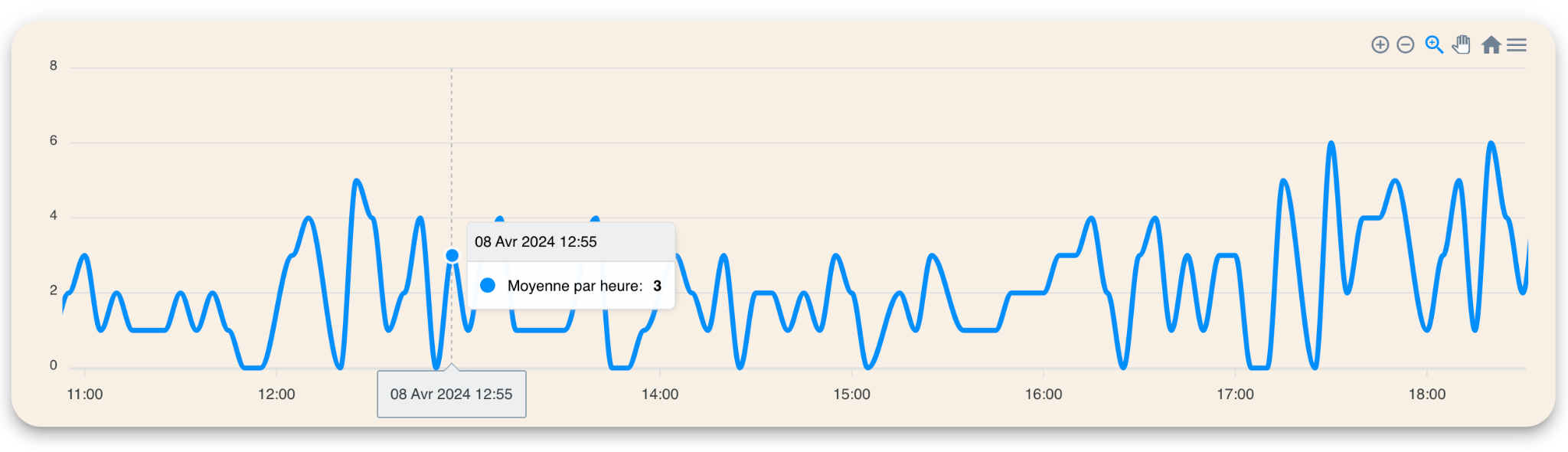
* Les salles numérotées de un à huit
* La salle multimodale
* La bibliothèque
* Le learning café

Une interface permet à l'utilisateur de choisir les salles qu’il souhaite afficher.



##### Figure 18 : Menu de sélection d’affichage des salles

De plus, cette interface propose également un graphique global affichant la moyenne de la fréquentation en fonction du temps.



##### Figure 19 : Graphique à courbe de l’occupation globale

Les autres pages présentes sur l’application sont liées à l’utilisateur et sa connexion et seront détaillées dans la partie [4.3.2 Gestion des utilisateurs.](#_2oeyzgl37vrq)

#### Liaison avec l’API\*

Une fois les pages principales mises en place dans l’interface utilisateur, il a fallu établir la connexion entre cette dernière et l’interface de programmation permettant de récupérer les données.

Cette phase n'a pas nécessité beaucoup de temps, mais elle a impliqué des ajustements dans les autorisations d'accès à l'API\* pour filtrer les requêtes entrantes. Cette mesure renforce la sécurité de l'API\* en restreignant l'accès aux informations uniquement aux utilisateurs autorisés, ce qui limite les points d'entrée potentiels.

Une fois cette étape franchie, nous avons procédé à l'implémentation des requêtes pour obtenir les données nécessaires à l'affichage des différents graphiques présents dans l'interface utilisateur. Les routes utilisées exigent obligatoirement une période de temps afin de limiter les données renvoyées. À cet effet, nous avons défini les intervalles de temps suivants :

* Pour la page d'accueil : intervalle quotidien.
* Pour l'historique : sur une semaine si aucune date n'est sélectionnée, ou sur la plage de temps choisie par l'utilisateur.

Cependant, les données renvoyées par l'API\* ne sont pas directement utilisables dans les graphiques. Comme mentionné dans la section précédente, ces données sont formatées pour représenter chaque passage en fonction d'un capteur, d'une horaire et d'un nombre de personnes.

Pour créer les graphiques de la page d'accueil, nous avons dû segmenter les informations à la fois par tranche horaire et par espace (bibliothèque ou salles de travail). Une fois cette tâche achevée, les données sont organisées et il ne reste plus qu'à calculer la moyenne pour chaque tranche horaire et à l'intégrer dans le graphique.

Pour générer les graphiques de la page d'historique, nous avons dû organiser les données par salle et les classer par ordre chronologique, permettant ainsi à l'utilisateur de sélectionner les salles souhaitées. De plus, pour chaque requête, il est nécessaire de calculer à la fois la moyenne par salle sur la période de temps choisie et par tranche horaire, fournissant ainsi une estimation de la fréquentation quotidienne pour ces dates. Cependant, aucune gestion n'a été ajoutée pour la sélection des dates ; une nouvelle requête est envoyée à chaque changement de sélection, et le processus décrit ci-dessus est appliqué de nouveau aux données.

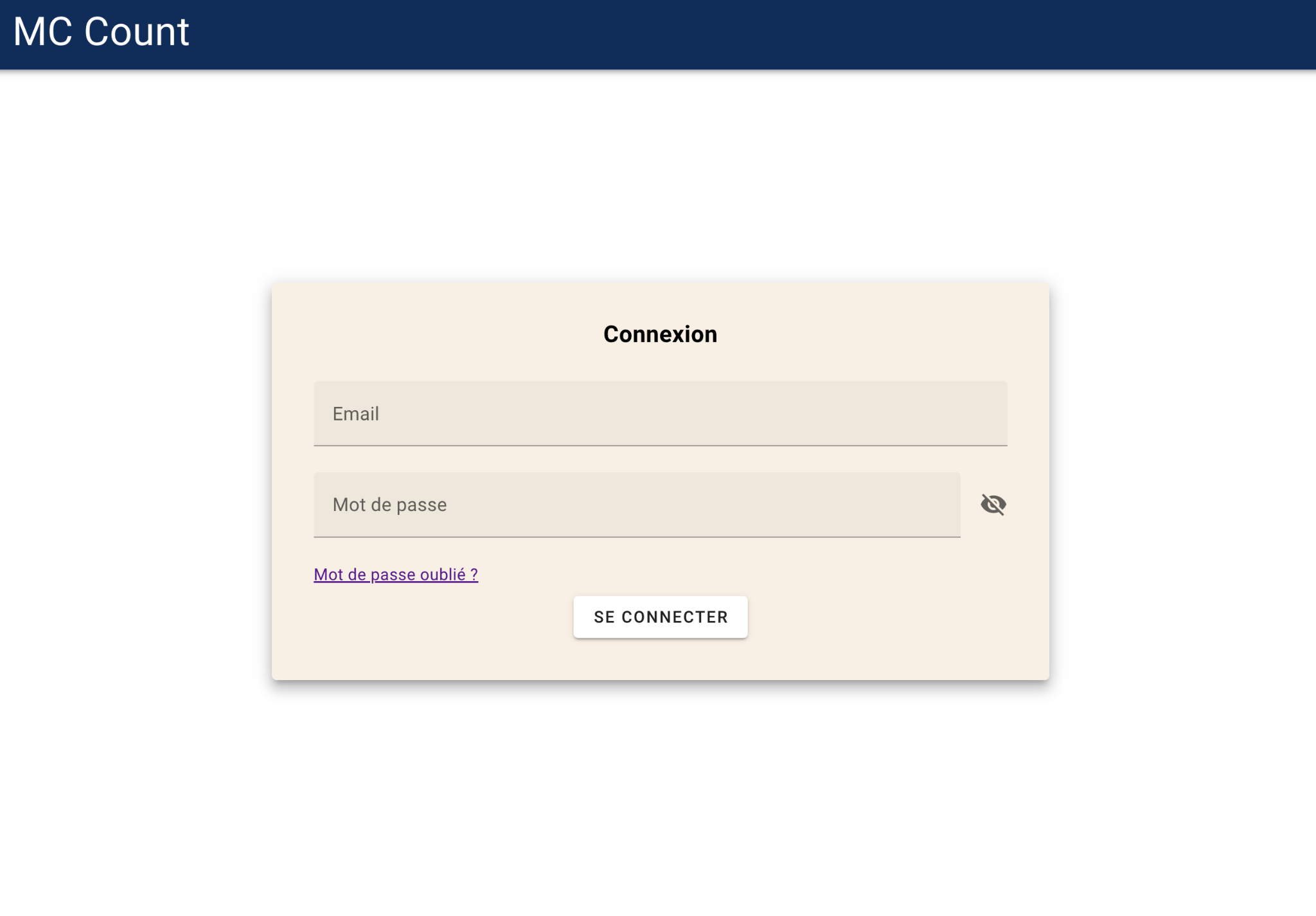
Une fois l’implémentation de l’API\* effectuée pour récupérer les informations d’occupation et de passage de la Maison Campus, nous l’avons aussi utilisé pour gérer les utilisateurs de la plateforme web.

### 4.3.2 Gestion des utilisateurs

#### La connexion

Lors de l'utilisation d'une solution web, l'étape initiale pour la plupart des utilisateurs est l'authentification. Cependant, il est crucial de contrôler l'accès à cette application pour éviter que n'importe qui puisse créer un compte et accéder aux informations. Pour garantir cette sécurité, nous avons mis en place un système d'administration qui permet à des administrateurs désignés d'envoyer des invitations pour la création de compte. Ce privilège peut être attribué à des personnes responsables de l'espace concerné, comme notre client Rémy Ginestet, responsable de la Maison Campus. Une fois invité, l’utilisateur reçoit un mail avec un mot de passe aléatoire qu’il sera amené à changer lors de sa première connexion sur le site.

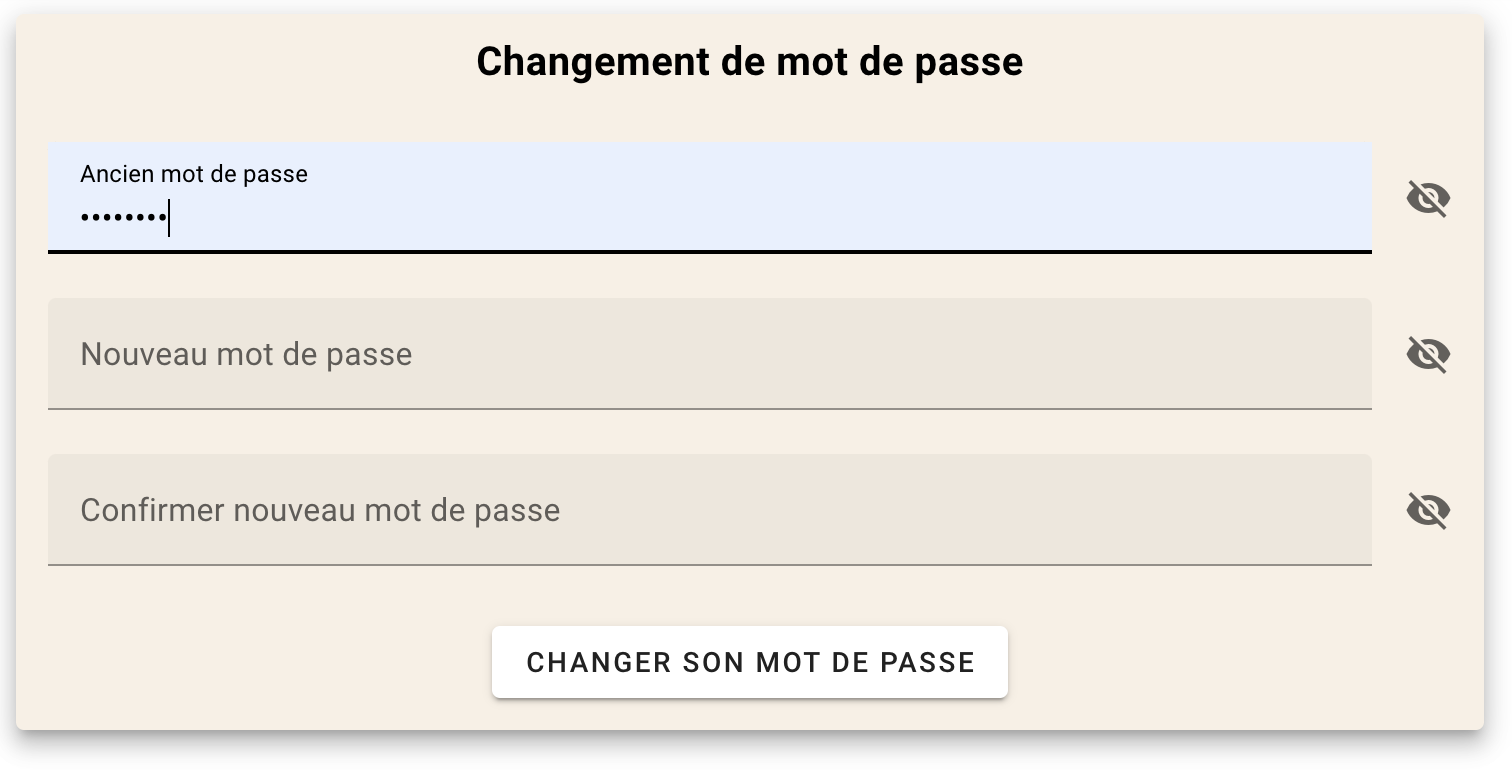
Comme nous l’avons vu dans [L’architecture de l’interface utilisateur](#_x69k3ttuwxnb) avec le router, la première page affichée lors du chargement est la page de connexion telle que :



##### Figure 20 : Page de connexion de la solution web

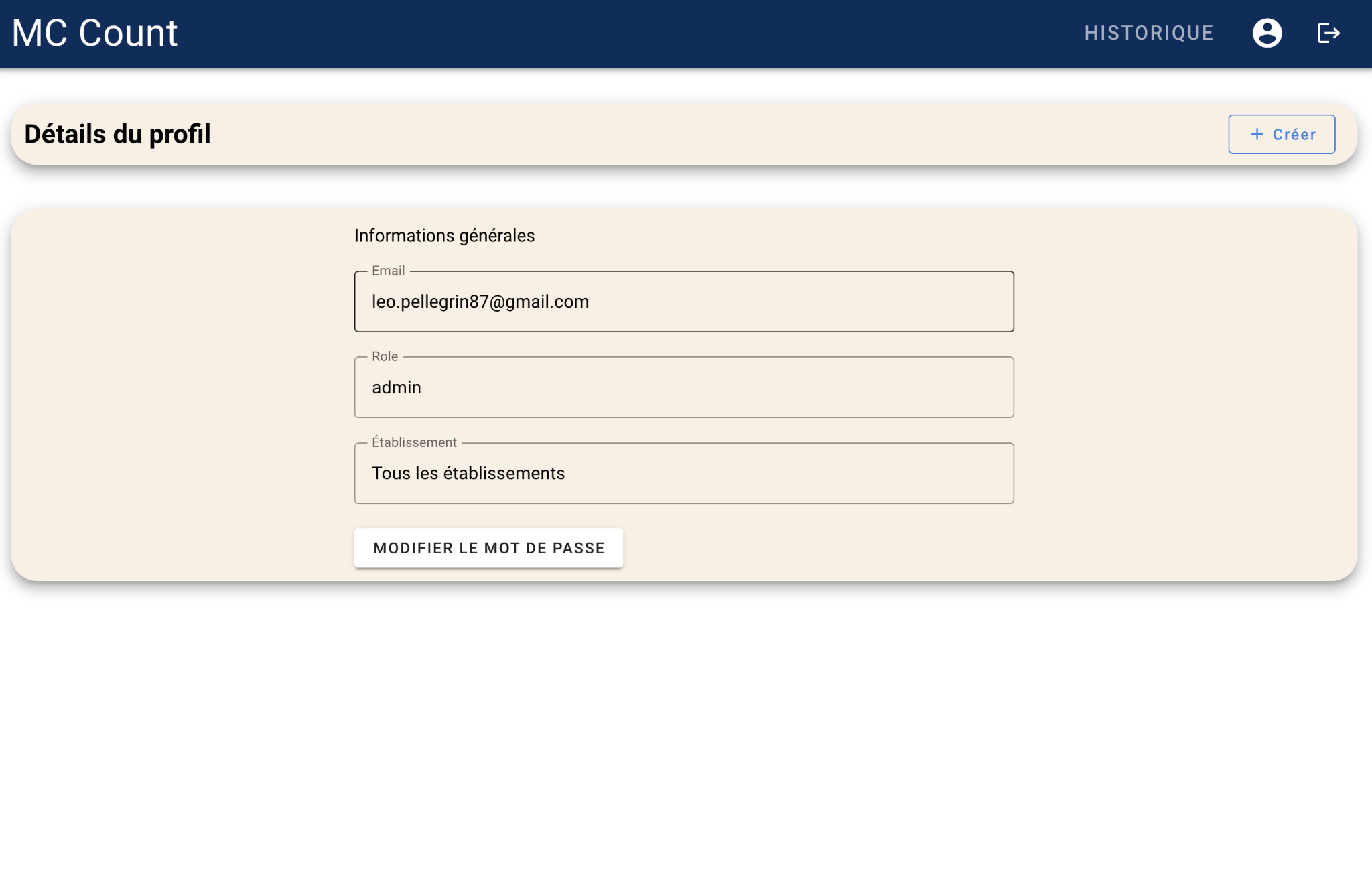
Cette page permet à la fois de se connecter, mais aussi permet d’envoyer un mot de passe généré aléatoirement à l’adresse mail renseigné dans le formulaire de connexion s'il a été oublié. Comme dit précédemment, il existe deux chemins possibles lors de la connexion :

* Première connexion de l’utilisateur : il est redirigé vers un formulaire permettant de modifier son mot de passe



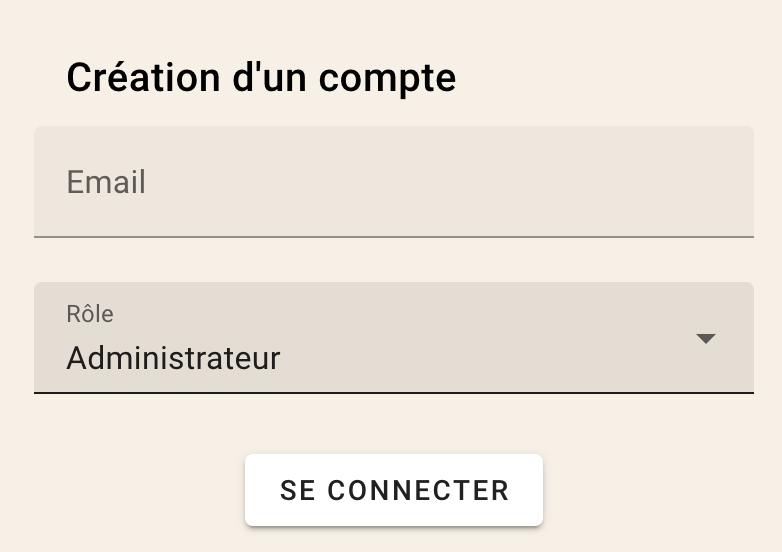
##### Figure 21 : Formulaire de changement de mot de passe

* Connexion habituelle d'un utilisateur : lorsqu'il accède à l'application, il est dirigé vers son espace utilisateur personnel. Une fois connecté, un jeton d'authentification est conservé dans le navigateur jusqu'à la fermeture du site, garantissant ainsi l'authentification de toutes les requêtes effectuées vers l'API.



##### Figure 22 : Page de profil

À travers cette interface, l’utilisateur peut consulter son email, son rôle ainsi que l’établissement auquel il est affecté. De plus, le rôle administrateur est assigné à tous les établissements, mais cela pourra être modifié pour n’avoir qu’un seul administrateur par établissement afin d’organiser l’accès à la plateforme. Ce rôle permet aussi de créer un compte à travers le bouton en haut à droite en renseignant une adresse mail ainsi que le rôle de ce dernier.



##### Figure 23 : Formulaire de création de compte utilisateur

#### Récupération des droits des utilisateurs

Comme vu dans la partie [gestion des utilisateurs](#_nnsw9i3izmhh) cotée backend\*, les informations liées à l’utilisateur comme son adresse mail ou son rôle sont stockées directement dans le jeton d’authentification. Cependant, elles sont codées d’une certaine manière. Afin de pouvoir les lire, nous avons utilisé une fonction native au langage JavaScript\* permettant de décoder une chaine de caractère codé sur une base 64. Cette méthode utilise une grande variété de caractère le rendant très utile dans la communication. De plus, il est l’une des bases de codage les plus utilisées dans le monde informatique à ce jour.

# 5. Outils et technologies utilisés

## 5.1 Structure logicielle

### 5.1.1 La base de données

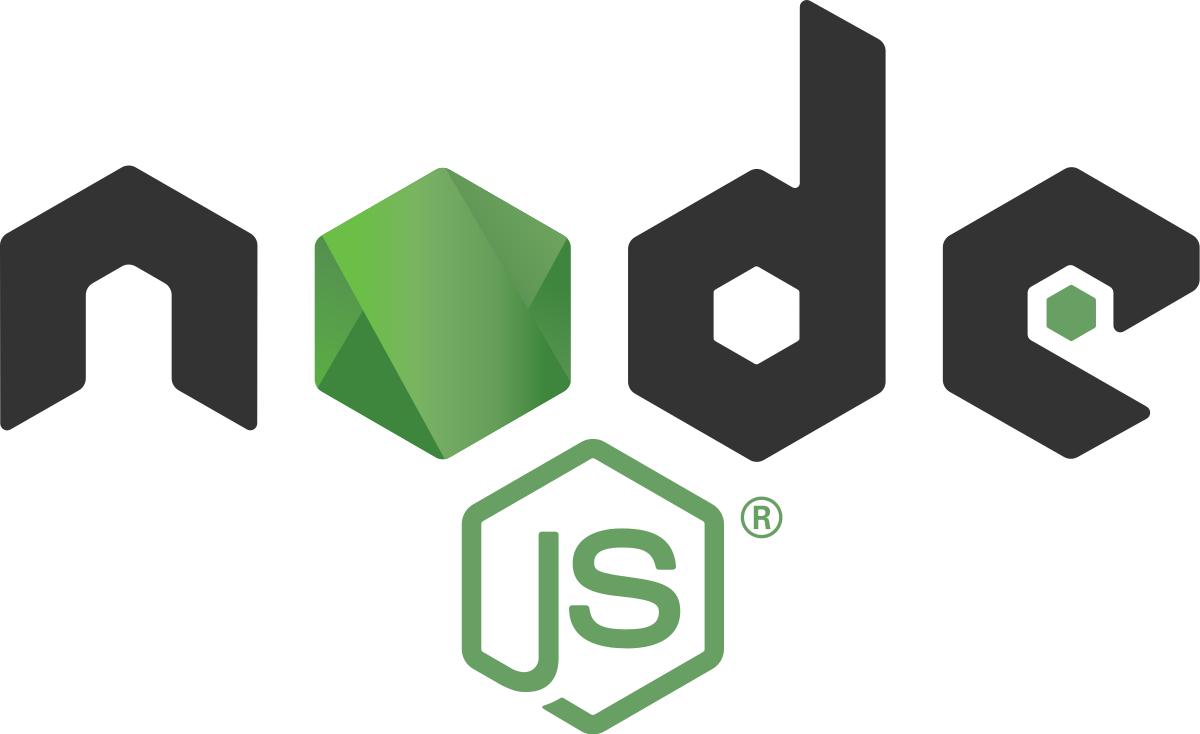
Pour ce projet, nous avons utilisé une base de données SQL\* PostGres. C’est une base de données que nous avons eu l’occasion d’utiliser lors de nos années d’études à ISIS et elle présente de nombreux avantages. C’est une base de données gratuite et open source, elle est fiable et surtout stable, et permet une sécurité renforcée de la donnée.



##### Figure 24 : Logo PostgreSQL\*

### 5.1.2 Le backend\*

Pour la gestion de l’API\*, nous avons décidé d’utiliser la technologie Node.js. Node.js est un environnement d’exécution open source coté serveur. Il est basé sur la technologie JavaScript\*, ce qui nous permet d’utiliser le JavaScript\* pour le front end\*, mais aussi pour le backend\*.



##### Figure 25 : Logo Node.JS

Pour réaliser l’API\*, nous avons utilisé le framework\* Express. Express est un framework\* pour construire des applications web basées sur Node.js, et il est en quelque sorte le framework\* standard pour le développement d’application et API\* Node.js. Il permet une gestion très simple des différentes URL et méthodes HTTP\* (GET, POST, PUT, DELETE...). Il permet aussi une gestion simple et sécurisé des droits utilisateurs, le tout de manière très intuitive.



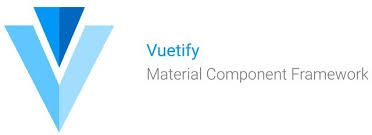
##### Figure 26 : Logo Express

### 5.1.3 Le frontend\*

Notre application doit présenter une interface graphique simple et intuitive dans le but d’aider le travail de personnes comme notre client dans leur analyse et leur demande de financement. Pour cela, nous utiliserons le framework\* VueJs ainsi que sa librairie Vuetify.

VueJs est un framework\* de développement d’interface graphique et utilisateur. Ce framework\* est très puissant, car il intègre de nombreuses fonctionnalités comme un compilateur très puissant et rapide, ce qui nous permet un gain de temps énorme. De plus, des librairies sont disponibles au sein du framework\*. Celui qui est intéressant pour notre projet est Vuetify, ce dernier ajoute des éléments d’interface graphique comme des formulaires, des barres de navigation ou encore des menus adaptatifs.

Cette partie sera responsable de l’affichage coté utilisateur, et pourra récupérer des informations via des appels API\* en direction du backend\*.



##### Figure 27 & 28 : Logo VueJs et Vuetify

En outre, pour répondre aux besoins spécifiques d'affichage des données dans notre application, nous avons intégré une librairie externe de premier ordre appelée ApexChart. Cette librairie offre une gamme diversifiée de graphiques pré-conçus, flexibles et hautement paramétrables, correspondant parfaitement à nos exigences.

Étant donné que les graphiques constituent une majeure partie de l’identité visuelle de notre application, le choix d'une librairie fiable et parfaitement interconnectable avec le framework\* utilisé est d’une importance cruciale. ApexChart s'est révélée être la solution idéale, offrant une intégration fluide avec notre environnement de développement tout en garantissant une performance optimale et une esthétique impeccable pour la visualisation de nos données. L'adoption d'ApexChart a non seulement simplifié le processus de création de graphiques complexes, mais a également enrichi l'expérience utilisateur en offrant une variété d'options de personnalisation.



##### Figure 29 : Logo ApexCharts

## 5.2 Logiciels et outils

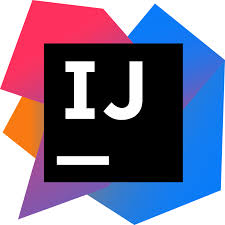
Au cours de ce projet, nous avons eu l’occasion d’utiliser divers outils et logiciels pour faciliter nos phases de maquettages, de tests, mais aussi pour l’hébergement de notre projet. Nous allons définir ces différents outils dans cette partie.

* Les outils de développements :

Afin de réaliser et développer nos projets, nous utilisons généralement des logiciels et des extensions de développement, plus communément appelés IDE\* (Environnement de Développement Intégré).

Le choix de ses IDE\* se fait en fonction des préférences personnelles, mais aussi des langages utilisés.

Pour le développement du backend\*, les IDE\* que nous avons utilisés sont WebStorm et IntelliJ. Ces deux outils de la gamme JetBrains sont très performants et offrent une large gamme d’outils et extension pour faciliter le développement en Node.js. WebStorm est plus spécialisé pour le JavaScript\*, mais nous sommes plus habitués à utiliser IntelliJ dans notre formation.



##### Figure 30 & 31 : Logo IntelliJ et WebStorm

Pour le développement du code relatif au fonctionnement des capteurs, nous avons utilisé l’IDE\* plugin de [VSCode](#_cyddybd0zfei) : PlatformIO. Elle offre un environnement de développement qui nous donne accès à plusieurs fonctionnalités telles que la gestion des bibliothèques, la compilation et le téléversement de code, ainsi que la prise en charge de plusieurs plateformes matérielles.



##### Figure 32 : Logo PlateformIO

Pour le développement de la partie frontend\*, nous avons opté pour l'utilisation de VS Code comme environnement de développement. Cette décision s'est fondée non seulement sur notre familiarité avec l'outil, mais aussi sur ses capacités avancées qui répondent parfaitement à nos besoins. En effet, VS Code offre une interface conviviale et une multitude d'extensions qui facilitent le développement, la gestion de projets et la collaboration au sein de l'équipe. Grâce à ses fonctionnalités de débogage intégrées et son support étendu pour les langages de programmation frontend\* tels que HTML\*, CSS\* et JavaScript\*, VS Code s'est imposé comme un choix judicieux pour garantir une productivité optimale tout au long du processus de développement.



##### Figure 33 : Logo VSCode

* Les outils de conceptions :

Lors de la phase de conception, nous avons dû réaliser des modèles et maquettes afin d’imaginer et conceptualiser notre projet. Cette conception s’est réalisée sur plusieurs plans, dans un premier temps la conception logicielle, puis la conception web et enfin la conception physique des capteurs.

Pour la conception des maquettes logicielles et de base de données, nous avons utilisé l’outil Star UML\*. C’est un outil de génie logiciel dédié à la modélisation UML\* qui permet la création simple de diagramme et schéma UML\*. Pour ce projet, nous l’avons utilisé pour réaliser des diagrammes de classes, diagramme de la base de données et le diagramme de cas d’utilisation.

Dans le cadre du processus de développement de notre application, j'ai choisi d'utiliser Figma, un outil de conception d'interfaces utilisateur interactives, pour créer des maquettes visuelles détaillées. Figma offre une plateforme collaborative qui permet une conception fluide et une communication efficace avec les parties prenantes, notamment notre client.



##### Figure 34 : Logo Figma

Afin de concevoir les schémas de montage des capteurs et de cartes Arduino\*, nous avons utilisé le logiciel TinkerCad. Cette plateforme offre un large choix de composants électroniques, facile d’utilisation, permettant de concevoir rapidement des circuits. De plus, ce logiciel permet la simulation des circuits, nous permettant de tester leur fonctionnement avant de passer à la réalisation physique.



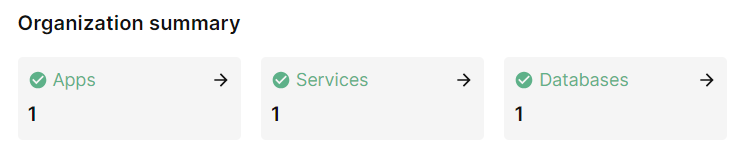
##### Figure 35 : Logo TinkerCad

* L’hébergement :

Lorsque nous développons les différentes parties de notre projet, nous le faisons de manière générale en locale. Cependant, afin d’avoir une adresse publique, qui permettra à tout le monde de pouvoir se connecter au site web, il faut l’héberger sur un serveur et lui donner une adresse publique. Dans le cadre de notre projet, nous n’avons pas pu avoir accès à un serveur, mais nous avons pu utiliser une solution d’hébergement en ligne.

Cet outil se nomme Koyeb, et il permet, via une connexion avec un compte Github, de pouvoir déployer une base de données et un service web pour une durée, et un usage limité.

Il nous permet ainsi d’obtenir une adresse publique pour notre base de données, notre backend\* et enfin notre front end.



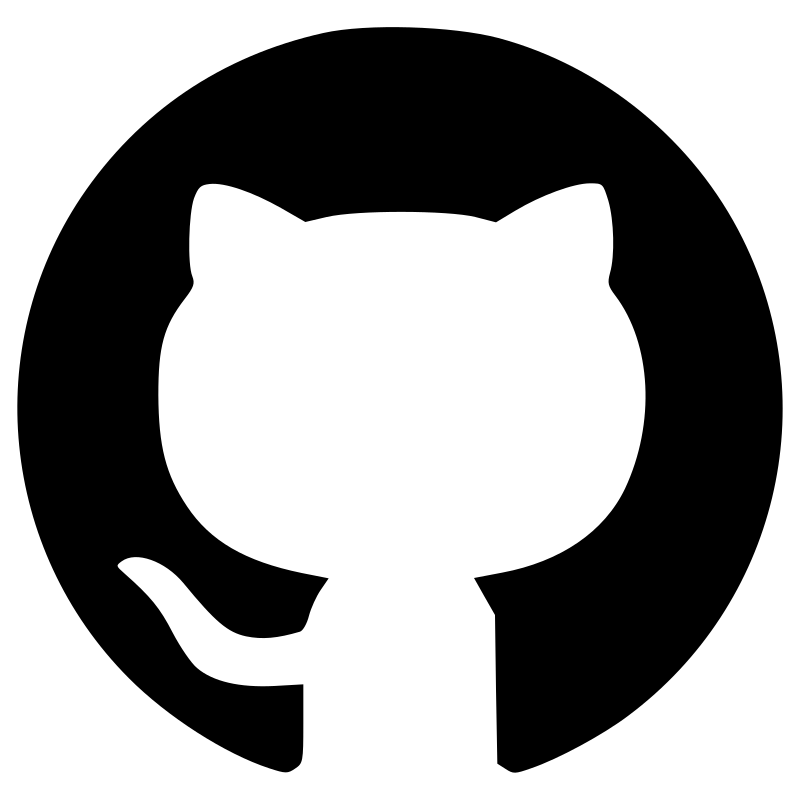
##### Figure 36 : Dashboard\* Koyeb

##### Figure 37 : Logo Koyeb

* Outils divers :

Au cours de ce projet, nous avons aussi utilisé un grand nombre d’outils qui ne rentrent pas dans une catégorie particulière et que nous allons définir ici.

Lors de la phase de développement, toute l’équipe a dû travailler sur plusieurs projets à la fois, et sur plusieurs parties différentes. Afin de faciliter le partage des fichiers et des modifications, nous avons utilisé l’outil de versionning\* GitHub. Github est une plateforme web, permettant créer, de stocker, et de partager du code en ligne. Pour ce projet, nous avons créé plusieurs projets GitHub, que nous avons cloné sur nos pc. En cas de modification, nous devons utiliser l’action de commit et de push, qui permet de publier sur la plateforme nos modifications locales. Et en cas de modification, les autres doivent pull, qui signifie récupérer les modifications cotées GitHub.



##### Figure 38 : Logo Github

Lors de la création d’une API\*, il peut être compliqué de tester les routes HTTP\* qui ont été créées. Postman est une application facilitant le test et l’envoi de requêtes HTTP\* vers une API\*. Son interface permet de stocker et de renseigner différentes informations et de visualiser simplement la réponse obtenue



##### Figure 39 : Logo PostMan

En cas de problème lors du développement, ou lors du besoin de générer du code standard, nous avons utilisé l’intelligence artificielle ChatGPT. C’est un outil très performant qui permet de générer des réponses cohérentes et utiles, mais il n’est pas non plus sans faille, il faut bien évidemment vérifier ses réponses, et ne pas tout prendre pour garanti. Dans ce projet, il a surtout été utile pour des questions simples, ou pour analyser des erreurs d’exécution de code.



##### Figure 40 : Logo ChatGPT

Un autre outil d’intelligence artificielle que nous avons eu l’occasion d’utiliser au cours de ce projet est GitHub Copilot. C’est une intelligence artificielle directement dans l’environnement de développement et spécialisé dans le développement. Il utilise ses connaissances basées sur un grand nombre de GitHub publique pour générer du code adapté. Il est très pratique, car il analyse le code déjà existant dans le fichier, pour proposer des suggestions de code que l’on peut accepter ou refuser. Pour ce projet, GitHub Copilot a notamment été très utile pour générer du code redondant ou pour donner des suggestions sur la suite. Et encore une fois, comme pour ChatGPT, ses résultats ne sont pas infaillibles, et il ne faut pas l’utiliser sans le comprendre, ou le vérifier.



##### Figure 41 : Logo GitHub Copilot

## 5.3 Organisation et gestion du dépôt Git

L'organisation du versionnage du code s'est effectuée à travers la création de plusieurs dépôts distincts. Cette approche a été adoptée afin de ne pas surcharger un unique dépôt et de garantir un espace de travail propre à chaque personne travaillant sur les différentes composantes du projet. Toutefois, cette stratégie ne se limite pas à une simple segmentation du code. Elle offre également la possibilité à chaque membre de l'équipe de compléter ou de consulter le code des autres collaborateurs impliqués dans le projet.

Cette structure en dépôts distincts présente de nombreux avantages. Elle favorise une meilleure gestion des versions en permettant de se concentrer sur des portions spécifiques du projet, tout en facilitant la collaboration et la communication entre les membres de l'équipe. De plus, elle garantit une plus grande souplesse dans le processus de développement, permettant à chacun de travailler de manière autonome tout en assurant une cohérence globale du code.

En somme, cette organisation du versionnage du code, basée sur des dépôts distincts, a permis d'optimiser la productivité, la qualité et la collaboration au sein de l'équipe de développement.

# 

# 7. Gestion de projet

## 7.1 Planification

La partie gestion de projet est essentielle afin de définir les objectifs du projet ainsi que les acteurs de chaque tâche de ce dernier.

Pour mener à bien cette phase, nous avons commencé par établir une définition claire des objectifs ainsi que la répartition des tâches entre les membres de l'équipe, mais aussi l'établissement d'un calendrier réaliste.

Une fois les premières étapes terminées, la création d'un plan de travail détaillé avec des jalons clés et des échéances qui vont nous servir de base tout au long du déroulement du projet.

La gestion de projet inclut également la coordination des efforts de l'équipe, la surveillance des progrès réalisés et l'identification proactive des éventuels écarts par rapport au plan initial. Pour mener à bien toutes ces étapes, nous avons adopté une approche avec la méthode Kanban\*, qui nous offre une excellente gestion pour un projet avec une équipe réduite comme la nôtre.

La communication régulière et transparente au sein de l'équipe et avec le tuteur est essentielle pour garantir une compréhension partagée des objectifs et des étapes à franchir.

Nous avons réalisé un diagramme de Gantt\* afin d’organiser de manière systématique les différentes tâches que nous devons accomplir tout au long du projet tutoré. Ce diagramme présente une vue d'ensemble claire et chronologique de nos activités, nous permettant de mieux gérer notre temps et d'identifier les dépendances entre les différentes étapes du projet.

Chaque tâche est soigneusement répertoriée avec une estimation réaliste du temps nécessaire à son accomplissement. En outre, des marges de manœuvre ont été intégrées dans le planning pour anticiper d'éventuels retards ou ajustements nécessaires.

Notre objectif principal en élaborant ce diagramme de Gantt\* est de mettre en évidence les tâches critiques, celles qui ont un impact significatif sur le bon déroulement du projet. En identifiant ces points sensibles, nous sommes en mesure de concentrer nos efforts et nos ressources là où ils sont les plus nécessaires, assurant ainsi le succès global du projet tutoré.

## 7.2 Organisation et Mise en place

Pour l’organisation de la gestion de projet, nous avons utilisé l’application Notion. Cette dernière nous a permis de nous organiser et de collaborer tout au long du projet. Le travail sur ce projet et notamment la répartition des tâches a été un petit peu particulier, car l’équipe a changé entre les deux semestres.

Au premier semestre, lors de la phase de conception, les tâches été réparties de la manière suivante :

* Damien s’est occupé des recherches sur les capteurs
* Mélia a géré la gestion de projet ainsi que les demandes de financements
* Léo réalisait les maquettes de l’application
* Loïs s’est occupé de la conception de la base de données et des diagrammes UML\*

Au second semestre, lors de la phase de développement de l’application, les tâches ont été réparties de la façon suivante :

* Léo : Développement du frontend\* de la solution logicielle
* Loïs s’est principalement occupé de la partie backend\*. Cette partie se compose de la réalisation de la base de données avec la création des tables, mais aussi de la création de l’API\* et de la gestion des utilisateurs.

## 7.3 Implication du client dans le projet

Tout au long du développement du projet, nous avons maintenu un contact étroit avec notre client, M. Ginestet. Pour ce faire, nous avons planifié des réunions environ une fois par mois afin de discuter de l'avancement du projet. Bien que nous reconnaissions que des réunions plus fréquentes auraient été souhaitables pendant la phase de développement, cela n'a pas été réalisable en raison de contraintes de temps.

Pour compenser cette limitation, nous avons maintenu une communication régulière par courrier électronique. L'objectif était de tenir notre client informé des progrès réalisés et de lui permettre de nous fournir des retours sur l'application. Ces retours nous ont été précieux pour adapter l'interface aux besoins de M. Ginestet, garantissant ainsi un résultat final de qualité qui répond efficacement à sa problématique.

# 8. Bilan et perspectives

## 8.1 État d’achèvement du projet

Les objectifs initiaux du projet étaient les suivants :

* Concevoir un système électronique permettant de surveiller et de compter l'occupation d'un espace donné.
* Développer un logiciel capable d'analyser les données collectées et de les présenter sous forme de statistiques sur l'occupation de cet espace.
  + Mettre en place une base de données pour stocker les données collectées.
  + Développer une partie backend\* chargée de la mise en forme des données.
  + Créer une interface frontend\* pour l'affichage convivial des statistiques sur l'occupation.

Actuellement, le projet a atteint un stade d'avancement significatif. La partie logicielle est entièrement fonctionnelle, bien que quelques ajustements restent nécessaires pour perfectionner certains aspects. Néanmoins, en raison de retards induits par une charge de travail accrue dans d'autres matières, certains retours et suggestions du client n'ont pas pu être pleinement pris en compte. Il est également essentiel de noter que le code développé est conçu pour être réutilisable. Par conséquent, une attention particulière a été accordée à sa clarté et à ses commentaires. Malgré cela, il subsiste potentiellement des améliorations à apporter. Il est impératif de continuer à optimiser le code afin de garantir son efficacité et sa maintenabilité à long terme.

Toutefois, le développement du système électronique n'a pas progressé aussi rapidement que prévu. Cette situation découle en partie d'une certaine méconnaissance du domaine des objets connectés malgré les cours dispensés en parallèle. De plus, l'utilisation de capteurs moins courants a posé un défi supplémentaire, car il était difficile de trouver des ressources en ligne pour nous guider dans leur intégration.

## 8.2 Difficultés rencontrées et solutions

Les principales difficultés rencontrées se divisent en deux parties : la partie logicielle et la partie matérielle de la solution.

Dans la première partie, nous avons rencontré plusieurs obstacles. Tout d'abord, la création de données simulées s'est avérée complexe. Nous devions générer des données réalistes en tenant compte de plages horaires spécifiques où les passages de personnes pouvaient être ajoutés. Créer un schéma aléatoire pour simuler l'entrée des individus tout en respectant le cheminement typique à travers la Maison Campus s'est avéré être un défi. Toutefois, en raison de contraintes de temps, nous avons dû nous limiter à la création de données basées uniquement sur des horaires fixes et avec des salles déliées.

Lors de la création de la base de données et de l’API\*, nous avons aussi eu beaucoup de difficultés sur la gestion des dates. Sur le serveur Koyeb, nous n’avions pas directement accès à la console et aux paramètres serveur, les données stockées en base pouvaient donc avoir des dates non conformes et qui pouvait se réinitialiser lors de maintenance. Nous avons donc pris la décision de rajouter une deuxième vérification et modification sur le backend\* dans l’API\* via un traitement sur le résultat renvoyé.

Ensuite, la mise en forme des données pour les graphiques de l'interface utilisateur s'est révélée complexe. Le traitement des données était particulièrement difficile à mettre en œuvre, nécessitant une attention minutieuse. De plus, nous devions effectuer des tris sur différents critères pour chaque graphique, ce qui a entraîné une redondance de code puisque nous devions traiter les données de manière répétée pour chaque graphique. D’autre part, la gestion des dates dans les données s'est avérée délicate en raison des différences de fuseaux horaires entre les différentes parties prenantes (backend\*, librairie extérieure, client), ce qui a ajouté une complexité supplémentaire à notre tâche.

La dernière difficulté rencontrée au niveau logiciel s'est située lors du déploiement de la solution. Étant donné que nous ne disposions pas d'un serveur privé pour effectuer ce déploiement, nous avons dû recourir à une plateforme en ligne, Koyeb, avec laquelle nous n'étions pas familiers. La prise en main s'est révélée assez complexe, d'autant plus que nous n'étions pas très à l'aise de base dans le domaine du DevOps\*. Cependant, grâce aux tutoriels disponibles et à la documentation fournie par le site lui-même, nous avons réussi à atteindre nos objectifs de déploiement.

Dans la seconde partie, lors des tests de la solution physique, nous avons essayé d’utiliser les capteurs de type “time of flight\*” mais ces derniers nous ont posé quelques problèmes.

En effet, le fonctionnement de ces derniers repose sur la définition de deux zones dans le champ de détection du capteur. Le principe est qu’un passage est enregistré si toutes ces conditions sont validées dans l’ordre :

* personne n’est détecté
* une personne est détectée dans la zone 1
* personne n’est détecté
* une personne est détectée dans la zone 2
* personne n’est détecté.

Or, la calibration de ces deux zones s’est avérée être très complexe. De plus, on trouve très peu de ressources concernant l’utilisation de ces capteurs sur internet et la calibration de ces derniers n’est jamais détaillée.

En conséquence, nous avons fait le choix de nous concentrer sur la mise en place des capteurs à faisceau de ruptures afin d’avoir un prototype fonctionnel à présenter à notre client.

## 8.3 Prolongements et ambition du projet

Le prolongement du projet pourra se concentrer sur la finalisation du système électronique et la réalisation de tests approfondis en conditions réelles. Il est primordial de résoudre des défis tels que la connexion à Internet pour le transfert des données collectées, ainsi que l'alimentation des cartes en fonction des branchements disponibles dans la Maison Campus. De plus, des améliorations sont envisageables sur la plateforme web, notamment l'intégration d'une fonction d'exportation des données au format Excel, conforme aux préférences du client.

Comme indiqué précédemment, le projet vise des objectifs ambitieux, dépendant étroitement de la réussite et de la fiabilité de la solution proposée. Dans un premier temps, l'installation pourrait se limiter à l'entrée de la Maison Campus afin de poursuivre les tests et de valider la conception du matériel. Par la suite, une fois, son utilité et sa fiabilité confirmées, le système pourra être déployé dans l'ensemble de l'établissement. Une fois pleinement opérationnel, il pourrait être étendu à d'autres sites tels que la bibliothèque d'Albi.

# 9. Bibliographie

* Solution logicielle : <https://ptut-front-leocorp.koyeb.app/>
* GitHub API\* : https://github.com/LoisGarcion/BddApiPTUT
* Node.js : <https://nodejs.org/en>
* PostGres : <https://www.postgresql.org/>
* JWT : <https://medium.com/@sbesnier1901/s%C3%A9curiser-une-api-avec-node-js-et-jwt-15e14d9df109>
* Bcrypt : <https://blog.logrocket.com/password-hashing-node-js-bcrypt/>
* Express : <https://expressjs.com/>
* Vue js : <https://vuejs.org/>
* Vuetify : <https://vuetifyjs.com/en/>
* ApexChart : https://apexcharts.com/
* Koyeb : https://www.koyeb.com/
* Capteurs : <https://cei-lab.github.io/ece3400-2017/tutorials/sensors/VL53L0X.html>

<https://github.com/adafruit/Adafruit_VL53L0X?tab=readme-ov-file>

<https://learn.adafruit.com/adafruit-vl53l1x/arduino>

<https://www.instructables.com/DIY-Laser-People-Counter/>

<https://www.mouser.fr/datasheet/2/414/OPB100-3241415.pdf>

# 10. Annexes

Création des tables :

| DROP TABLE PASSAGE; DROP TABLE CAPTEUR; DROP TABLE SALLE; DROP TABLE ETABLISSEMENT; DROP TABLE APIUSER;  ALTER DATABASE koyebdb SET timezone TO 'Europe/Paris';  CREATE TABLE Etablissement ( numeroEtab SERIAL PRIMARY KEY, nomEtab VARCHAR(40) );  CREATE TABLE Salle ( numeroSalle SERIAL PRIMARY KEY, numeroEtab INT, nomSalle VARCHAR(40), FOREIGN KEY (numeroEtab) REFERENCES Etablissement(numeroEtab) );  CREATE TABLE Capteur ( numeroCapteur SERIAL PRIMARY KEY, numeroSalle1 INT, numeroSalle2 INT, FOREIGN KEY (numeroSalle1) REFERENCES Salle(numeroSalle), FOREIGN KEY (numeroSalle2) REFERENCES Salle(numeroSalle) );  CREATE TABLE Passage ( numeroPassage SERIAL PRIMARY KEY, nbPersonne INT, entrant BOOL, datePassage TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP, numeroSalle INT, numeroCapteur INT, FOREIGN KEY (numeroSalle) REFERENCES Salle(numeroSalle) );  CREATE TABLE ApiUser ( apiUsername VARCHAR(20) PRIMARY KEY, apiPasswordHash VARCHAR(64), apiRole VARCHAR(10), passwordChanged BOOL ); |
| --- |

Création des procédures :

| DROP PROCEDURE ADD\_PASSAGE; SET TIME ZONE 'Europe/Paris'; SHOW TIME ZONE;  CREATE OR REPLACE PROCEDURE ADD\_PASSAGE(  p\_numeroSalleEntrante SALLE.numeroSalle%TYPE DEFAULT NULL,  p\_numeroSalleSortante SALLE.numeroSalle%TYPE DEFAULT NULL ) AS $$ DECLARE  v\_nbPersonnePassageEntrant PASSAGE.nbPersonne%TYPE;  v\_nbPersonnePassageSortant PASSAGE.nbPersonne%TYPE;  v\_now TIMESTAMP WITH TIME ZONE; BEGIN  -- Set the timezone to Europe/Paris  SET TIME ZONE 'Europe/Paris';  PERFORM pg\_advisory\_lock(1); -- Choose a unique lock identifier   -- Get the current timestamp with the correct timezone  v\_now := CURRENT\_TIMESTAMP;   IF p\_numeroSalleEntrante IS NOT NULL THEN  SELECT pas.nbPersonne INTO v\_nbPersonnePassageEntrant  FROM PASSAGE pas  WHERE pas.numeroSalle = p\_numeroSalleEntrante  AND pas.numeroPassage = (SELECT MAX(numeroPassage)  FROM PASSAGE  WHERE numeroSalle = p\_numeroSalleEntrante);  INSERT INTO PASSAGE(nbPersonne, entrant, datePassage, numeroSalle)  VALUES(v\_nbPersonnePassageEntrant+1, TRUE, v\_now, p\_numeroSalleEntrante);  END IF;   IF p\_numeroSalleSortante IS NOT NULL THEN  SELECT pas.nbPersonne INTO v\_nbPersonnePassageSortant  FROM PASSAGE pas  WHERE pas.numeroSalle = p\_numeroSalleSortante  AND pas.numeroPassage = (SELECT MAX(numeroPassage)  FROM PASSAGE  WHERE numeroSalle = p\_numeroSalleSortante);  IF v\_nbPersonnePassageSortant = 0 THEN  RAISE EXCEPTION 'Le nombre de personne d''une salle ne peut pas être inférieur à 0';  END IF;  INSERT INTO PASSAGE(nbPersonne, entrant, datePassage, numeroSalle)  VALUES(v\_nbPersonnePassageSortant-1, FALSE, v\_now, p\_numeroSalleSortante);  END IF;  PERFORM pg\_advisory\_unlock(1); -- Use the same lock identifier as above END; $$ LANGUAGE plpgsql;  CREATE OR REPLACE PROCEDURE ADD\_PASSAGE\_DATE(  p\_numeroSalleEntrante SALLE.numeroSalle%TYPE DEFAULT NULL,  p\_numeroSalleSortante SALLE.numeroSalle%TYPE DEFAULT NULL,  p\_date PASSAGE.datePassage%TYPE DEFAULT NULL ) AS $$ DECLARE  v\_nbPersonnePassageEntrant PASSAGE.nbPersonne%TYPE;  v\_nbPersonnePassageSortant PASSAGE.nbPersonne%TYPE;  v\_now TIMESTAMP WITH TIME ZONE; BEGIN  -- Set the timezone to Europe/Paris  SET TIME ZONE 'Europe/Paris';  PERFORM pg\_advisory\_lock(1); -- Choose a unique lock identifier   IF p\_numeroSalleEntrante IS NOT NULL THEN  SELECT pas.nbPersonne INTO v\_nbPersonnePassageEntrant  FROM PASSAGE pas  WHERE pas.numeroSalle = p\_numeroSalleEntrante  AND pas.numeroPassage = (SELECT MAX(numeroPassage)  FROM PASSAGE  WHERE numeroSalle = p\_numeroSalleEntrante);  INSERT INTO PASSAGE(nbPersonne, entrant, datePassage, numeroSalle)  VALUES(v\_nbPersonnePassageEntrant+1, TRUE, p\_date, p\_numeroSalleEntrante);  END IF;   IF p\_numeroSalleSortante IS NOT NULL THEN  SELECT pas.nbPersonne INTO v\_nbPersonnePassageSortant  FROM PASSAGE pas  WHERE pas.numeroSalle = p\_numeroSalleSortante  AND pas.numeroPassage = (SELECT MAX(numeroPassage)  FROM PASSAGE  WHERE numeroSalle = p\_numeroSalleSortante);  IF v\_nbPersonnePassageSortant = 0 THEN  RAISE EXCEPTION 'Le nombre de personne d''une salle ne peut pas être inférieur à 0';  END IF;  INSERT INTO PASSAGE(nbPersonne, entrant, datePassage, numeroSalle)  VALUES(v\_nbPersonnePassageSortant-1, FALSE, p\_date, p\_numeroSalleSortante);  END IF;  PERFORM pg\_advisory\_unlock(1); -- Use the same lock identifier as above END; $$ LANGUAGE plpgsql; |
| --- |

Insertion des données de base :



| DELETE FROM PASSAGE;  DELETE FROM CAPTEUR;  DELETE FROM SALLE;  DELETE FROM ETABLISSEMENT;  INSERT INTO ETABLISSEMENT (nomEtab) VALUES('Etab test');  INSERT INTO SALLE (numeroEtab, nomSalle) VALUES(2, 'Bibliothèque'); INSERT INTO SALLE (numeroEtab, nomSalle) VALUES(2, 'Salle 1'); INSERT INTO SALLE (numeroEtab, nomSalle) VALUES(2, 'Salle 2'); INSERT INTO SALLE (numeroEtab, nomSalle) VALUES(2, 'Salle 3'); INSERT INTO SALLE (numeroEtab, nomSalle) VALUES(2, 'Salle 4'); INSERT INTO SALLE (numeroEtab, nomSalle) VALUES(2, 'Salle 5'); INSERT INTO SALLE (numeroEtab, nomSalle) VALUES(2, 'Salle 6'); INSERT INTO SALLE (numeroEtab, nomSalle) VALUES(2, 'Salle 7'); INSERT INTO SALLE (numeroEtab, nomSalle) VALUES(2, 'Salle 8'); INSERT INTO SALLE (numeroEtab, nomSalle) VALUES(2, 'Salle 9');  INSERT INTO CAPTEUR (numeroSalle1) VALUES(5); INSERT INTO CAPTEUR (numeroSalle1) VALUES(6); INSERT INTO CAPTEUR (numeroSalle1) VALUES(7); INSERT INTO CAPTEUR (numeroSalle1) VALUES(8); INSERT INTO CAPTEUR (numeroSalle1) VALUES(9); INSERT INTO CAPTEUR (numeroSalle1) VALUES(10); INSERT INTO CAPTEUR (numeroSalle1) VALUES(11); INSERT INTO CAPTEUR (numeroSalle1) VALUES(12); INSERT INTO CAPTEUR (numeroSalle1) VALUES(13); INSERT INTO CAPTEUR (numeroSalle1) VALUES(14);  INSERT INTO PASSAGE(nbPersonne, entrant, datePassage, numeroSalle) VALUES(0,NULL,NOW(),5); INSERT INTO PASSAGE(nbPersonne, entrant, datePassage, numeroSalle) VALUES(0,NULL,NOW(),6); INSERT INTO PASSAGE(nbPersonne, entrant, datePassage, numeroSalle) VALUES(0,NULL,NOW(),7); INSERT INTO PASSAGE(nbPersonne, entrant, datePassage, numeroSalle) VALUES(0,NULL,NOW(),8); INSERT INTO PASSAGE(nbPersonne, entrant, datePassage, numeroSalle) VALUES(0,NULL,NOW(),9); INSERT INTO PASSAGE(nbPersonne, entrant, datePassage, numeroSalle) VALUES(0,NULL,NOW(),10); INSERT INTO PASSAGE(nbPersonne, entrant, datePassage, numeroSalle) VALUES(0,NULL,NOW(),11); INSERT INTO PASSAGE(nbPersonne, entrant, datePassage, numeroSalle) VALUES(0,NULL,NOW(),12); INSERT INTO PASSAGE(nbPersonne, entrant, datePassage, numeroSalle) VALUES(0,NULL,NOW(),13); INSERT INTO PASSAGE(nbPersonne, entrant, datePassage, numeroSalle) VALUES(0,NULL,NOW(),14);  CALL ADD\_PASSAGE(1,NULL); CALL ADD\_PASSAGE(2,NULL);  CALL ADD\_PASSAGE(1,2); |
| --- |

# 

# 11. Table des figures

[Figure 1 : Modèle conceptuel de données 7](#_ttacvds65ojt)

[Figure 2 : Exemple d’application du modèle conceptuel de données 8](#_6t8w49mxolbg)

[Figure 3 : Use Case 9](#_vrthsb5tr5nb)

[Figure 4 : Diagramme de la classe ApiUser 10](#_he6pm98wczug)

[Figure 5 : Maquette de la page d'accueil 11](#_4ldr4u6ib2d1)

[Figure 6 : Maquette de la page d’historique personnalisable 12](#_s9ibyzm2ncie)

[Figure 7 : Devis des capteurs pour la phase de test 15](#_lcuqn5py6x0p)

[Figure 8 : Schéma du montage des capteurs à faisceau de rupture\* 16](#_kgrhobn0g5hc)

[Figure 9 : Modèle conceptuel de données 19](#_xk77cajz3vqx)

[Figure 10 : Autorisations CORS\* API\* 21](#_wz376hlqubrj)

[Figure 11 : Modèle conceptuel de données des utilisateurs 23](#_948yqf9gzeh2)

[Figure 12 : Chemins définis dans le router Vue 26](#_jkv4fp5zuz08)

[Figure 13 : Affichage des occupations globales 27](#_ch5o5pbwxjh)

[Figure 14 : Graphique d’occupation globale 27](#_b3r6monjlvnm)

[Figure 15 : Graphique occupation bibliothèque et salles de travail 27](#_afjnub1cwrir)

[Figure 16 : Menu de sélection de la plage de date 28](#_ovnh2bezh2w)

[Figure 17 : Graphique à courbe de l’évolution de l’occupation 28](#_hzcpfda1oox2)

[Figure 18 : Menu de sélection d’affichage des salles 29](#_dyyycwd4xlhq)

[Figure 19 : Graphique à courbe de l’occupation globale 29](#_h2409krf4v9u)

[Figure 20 : Page de connexion de la solution web 31](#_heoo2ywofysg)

[Figure 21 : Formulaire de changement de mot de passe 32](#_2ob0vy1klz29)

[Figure 22 : Page de profil 32](#_tm01nlq07h8v)

[Figure 23 : Formulaire de création de compte utilisateur 33](#_2axmy5l0vx6)

[Figure 24 : Logo PostgreSQL\* 34](#_selfj1o4e22w)

[Figure 25 : Logo Node.JS 34](#_iijcqvnzqr0j)

[Figure 26 : Logo Express 35](#_7njw3tso9bb)

[Figure 27 & 28 : Logo VueJs et Vuetify 35](#_z91qcwaonf6h)

[Figure 29 : Logo ApexCharts 36](#_ypi8ypffsa7z)

[Figure 30 & 31 : Logo IntelliJ et WebStorm 37](#_dxi6soyo8479)

[Figure 32 : Logo PlateformIO 37](#_161h0a1mrzzv)

[Figure 33 : Logo VSCode 38](#_cyddybd0zfei)

[Figure 34 : Logo Figma 38](#_cdsyz143u94j)

[Figure 35 : Logo TinkerCad 39](#_q5y78uqhkg1p)

[Figure 36 : Dashboard\* Koyeb 39](#_6diczdi2gbj1)

[Figure 37 : Logo Koyeb 39](#_eaa4essotgxr)

[Figure 38 : Logo Github 40](#_k0z3p1g9pc8q)

[Figure 39 : Logo PostMan 40](#_u6zioocvnebs)

[Figure 40 : Logo ChatGPT 40](#_qj3zcdvds7tb)

[Figure 41 : Logo GitHub Copilot 41](#_qgyxd87v9nho)

# 12. Glossaire

* **API :** Une API, ou Application Programming Interface, est une interface permettant la connexion entre plusieurs applications ou logiciels.
* **Arduino Uno :** Carte électronique permettant l’apprentissage de l’électronique.
* **Backend :** Le backend est la partie que l’utilisateur ne voit pas, c’est ce qui se passe en arrière-plan. Dans notre projet, le backend correspond au lien avec la base de données et le serveur.
* **Capteur “break-beam” ou capteur à faisceau de rupture :** Dispositif qui détecte la présence ou l'absence d'objets en mesurant l'interruption d'un faisceau lumineux entre une source lumineuse et un récepteur.
* **Capteur “time of flight” :** Dispositif qui mesure le temps nécessaire à un signal pour voyager jusqu'à un objet et revenir au capteur, permettant ainsi de calculer la distance entre le capteur et l'objet avec précision.
* **CORS :** Les Cross-origin resource sharing sont une option de vérification pour une application web pour sécuriser et contraindre les requêtes venant d’autres applications.
* **CSS :** Les feuilles de style en cascade forment un langage informatique qui décrit la présentation des documents HTML et XML.
* **Dashboard :** Le tableau de bord de gestion est un outil d'évaluation constitué de plusieurs indicateurs de performance à des moments donnés ou sur des périodes données.
* **DevOps :** Pratique technique visant à l'unification du développement logiciel et de l'administration des infrastructures informatiques, notamment l'administration système.
* **Framework :** Ensemble d'outils et de composants logiciels.
* **Frontend :** Le frontend correspond en quelque sorte à l’interface utilisateur, c’est la partie avec laquelle l’utilisateur va interagir.
* **Gantt :** Outil utilisé en ordonnancement et en gestion de projet et permettant de visualiser dans le temps les diverses tâches composant un projet.
* **HTML :** Le HyperText Markup Language, généralement abrégé HTML, est le langage de balisage conçu pour représenter les pages web.
* **IDE :** Environnement de développement.
* **JavaScript :** Langage de programmation de scripts principalement employé dans les pages web interactives.
* **JWT :** Les Json Web Token sont un format pour sécuriser des échanges d’informations. Il permet notamment d’identifier un utilisateur.
* **Kanban :** Outil de gestion de projet visuel qui permet de suivre les tâches en temps réel, de gérer les priorités et de favoriser la collaboration entre les membres de l'équipe.
* **MCD :** Modèle conceptuel de données.
* **Micro-contrôleur :** petit ordinateur sur une puce, utilisé pour contrôler des dispositifs et exécuter des tâches spécifiques dans divers appareils électroniques.
* **PL/pgSQL :** Le PL/pgSQL est le langage de script de PostGres, il permet la réalisation de procédures et de fonctions.
* **Raspberry Pi :** Nano-ordinateur pouvant se connecter à un moniteur, à un ensemble clavier/souris et disposant d'interfaces Wi-Fi, Bluetooth et Ethernet.
* **SMIX :** Syndicat Mixte.
* **SQL :** Le SQL ou Structure Query Language est un langage permettant de communiquer avec une base de données.
* **Template :** Modèle de conception de logiciel ou de présentation des données.
* **UML :** Le Langage de Modélisation Unifié conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel.
* **Versionning :** Méthode de gestion des versions d'un même produit.
* **HTTP :** Protocole de transmission permettant à l'utilisateur d'accéder à des pages web par l'intermédiaire d'un navigateur.