ECOLE SUPERIEURE D'INGENIEUR DE RENNES

UNIVERSITÉ DE RENNES 1

Projet IoT - Semestre 2

LockDoor

Rapport du projet de porte connectée

Grosos Corentin - Certenais Mathis - Delagarde Théo

24/05/2022

Sommaire

Introduction

Dans le cadre de notre deuxième année d'école d'ingénieur en **Technologie de l'Information** option "Internet of Things", nous sommes un groupe de trois étudiants qui avons eu à réaliser un projet orienté "objets connectés".

Parmi les 6 sujets proposés, nous nous sommes penchés sur le "Smart Access", soit l'ouverture ou la **porte connectée**. Ce choix a été effectué car en plus de nécessiter tous les aspects de programmation d'une application classique (interface, serveur, bdd etc.), le projet requiert également de mettre en place une connectique électronique. Issus tous 3 de DUT GEII, c'était un réel plus quant à notre choix de projet.

Ce document est un rapport de notre travail effectué tout au long de l'année sur ce projet que nous avons nommé **"LockDoor"**; la mise en place du cahier des charges, les recherches , les choix de technologies, l'avancement actuel du projet, ses améliorations possibles, ainsi qu'une prise de recul plus personnelle sur le travail effectué.

Contexte

Même lors d'un projet d'école, et même si le sujet n'est pas complètement ouvert, il reste très important de se demander **en quoi** c'est un projet qui pourrait réellement voir le jour; quelles plus-values apporte-t-il :

Au quotidien, l'homme fait face à de nombreux événements qu'il vit comme des contraintes. D'origines très diverses, elles peuvent être **matérielles**, physiques, temporelles, sociales.

Avec l'avènement du numérique, l'homme du XXIème siècle a fait du **smartphone** le vecteur principal pour échapper à ces **contraintes**. En effet, ce condensé de dématérialisation et de centralisation des tâches nous permet aujourd'hui en quelques clics de contacter ses proches, d'organiser des évènements ou encore d'effectuer des achats en ligne.

Néanmoins, de nombreux objets ou services échappent encore à cette tendance. Parmi ceux-ci, nous pouvons compter la **clef de serrure**, qui tient un rôle crucial, celui de l'accès au domicile.

N'ayant pas de représentant numérique, le **prêt de clef**, et a fortiori **le vol ou la perte**, peut entraîner de sérieux ennuis.

Objectifs

Mettre en place une **ouverture numérique** grâce au smartphone relié à une porte connectée. Ce mécanisme a plusieurs avantages.

D'abord, il limite les problèmes liés à la perte ou au vol, puisque même dans un tel cas, l'ouverture resterait possible avec un autre appareil via le **Cloud**.

Ensuite, il réduit les contraintes liées au prêt de clefs. En effet, le propriétaire peut autoriser d'autres utilisateurs, en fonction de conditions qu'il aura **lui-même défini**, à déverrouiller la porte. Ainsi, il peut par exemple autoriser le compte de son fils à rentrer **à tout moment**, inviter **ponctuellement** un ami, ou encore **refuser l'accès** de l'aide ménagère en dehors de ses heures de travail.

I) Cahier des charges

a) Définition des User Stories

Nous avons commencé par établir toutes les User Stories auxquelles nous voudrions que notre application complètement aboutie répondent. Pour ce faire, il a fallu imaginer quelles actions pourraient être réalisées, par qui et dans quelles conditions.
Les User Stories ont été pensé selon trois cas utilisateurs :

- **user** : utilisation générale de l'application, aucun rapport de lien avec quelconque portes connectées.
- **guest** : lorsque l'utilisateur utilise l'application en fonction de portes qu'il ne possède pas.
- **owner** : l'utilisateur utilise l'application en tant que propriétaire de sa porte.

Comme nous souhaitons laisser la possibilité à un propriétaire de définir les droits qu'il attribue aux utilisateurs qu'il invite sur sa porte, nous avons dans le même temps rédigé ces droits dans des **authorisation** stories.

User Stories

- En tant qu'**utilisateur**, je souhaite pouvoir télécharger l'**application Lockdoor** pour profiter de toutes les **fonctionnalités** (menu, langues, options, paramètres, profils invités et propriétaires, etc).
- En tant qu'**utilisateur**, je souhaite disposer d'un **compte** et d'un **ID Lockdoor** pour avoir une **identité unique** vis-à-vis de l'application et des autres utilisateurs.
- En tant qu'**utilisateur**, je souhaite pouvoir **appareiller** mon smartphone à une porte connectée que je viens d'acheter pour en devenir le **propriétaire** à la **1ère utilisation**.
- En tant qu'**utilisateur**, je souhaite pouvoir demander en **ami** d'autres **utilisateurs** via leur ID Lockdoor pour faciliter les interactions entre nous.
- En tant qu'**utilisateur**, je souhaite pouvoir **me faire ajouter** sur des portes connectées pour obtenir le rôle d'**invité**.

Guest stories

- En tant qu'**invité**, je souhaite pouvoir accéder à la **liste** de portes connectées auxquelles j'ai **déjà** été **invité**.
- En tant qu'**invité**, je souhaite pouvoir observer les **autorisations** que je possède sur **une porte** en particulier.
- En tant qu'**invité**, je souhaite pouvoir **exercer** sur une porte **les droits** qui m'ont été **accordés**.

Owner stories

- En tant que **propriétaire**, je souhaite pouvoir **inviter** des **utilisateurs** sur ma porte pour leur octroyer le profil d'**invité**.
- En tant que **propriétaire**, je souhaite pouvoir **définir** des **autorisations personnalisables** à **chacun** de mes invités.
- En tant que **propriétaire**, je souhaite pouvoir créer des **rôles** qui, sous **un nom**, regroupent une **sélection d'autorisations**.
- En tant que **propriétaire**, je souhaite pouvoir **adresser un rôle** que j'ai **défini** à un **nouvel invité** pour éviter les redondances et **faciliter l'ajout** d'**invités**.
- En tant que **propriétaire**, je souhaite pouvoir, **à tout instant**, modifier les rôles et les **autorisations** de mes **invités**.
- En tant que **propriétaire**, je souhaite pouvoir, **à tout instant**, ajouter ou supprimer des **utilisateurs** de ma liste d'**invités**.
- En tant que **propriétaire**, je souhaite pouvoir, **à tout instant**, bloquer des **utilisateurs** ou des **invités** de ma porte (Blacklisting) pour leur en interdire **toute interaction**.

Authorisation stories

- Posséder **tous les droits**
- **Connaître l'état** de la porte
- **Modifier l'état** de la porte
    - *à distance (Internet)*
    - *à proximité (NFC)*
- Inviter **ponctuellement** des **utilisateurs**
    - *à distance (Internet)*
    - *à proximité (NFC)*
- Ajouter des **utilisateurs** comme **invités**
    - **ne peut** octroyer des droits **autres** que les siens
    - peut autoriser le nouvel **invité**
- Accès à l'**historique d'ouverture**

Nous avons à présent un aperçu général et complet de l'utilisation de l'application selon les cas d'utilisation. Mais pour le bon déroulement du projet, il faut découper le développement de l'application par étape, bloc par bloc, par ordre de priorité.

b) Hiérarchisation des User Stories

PRIORITÉS 1 (compte + ID, existence des rôles propriétaire, invité) :

- En tant qu'**utilisateur**, je souhaite pouvoir me créer un **compte Lockdoor** pour avoir une **identité unique** vis-à-vis de l'application et des autres utilisateurs.
- En tant qu'**utilisateur**, je souhaite pouvoir **appareiller** mon smartphone à une porte connectée que je viens d'acheter pour en devenir le **propriétaire** à la **1ère utilisation**.
- En tant que **propriétaire**, je souhaite pouvoir **gérer** l'état des portes que je possède.
- En tant que **propriétaire**, je souhaite pouvoir **inviter** des **utilisateurs** sur ma porte pour leur octroyer le profil d'**invité**.
- En tant que **propriétaire**, je souhaite pouvoir **définir** des **autorisations personnalisables** à **chacun** de mes invités.
- En tant qu'**utilisateur**, je souhaite pouvoir **me faire ajouter** sur des portes connectées pour obtenir le rôle d'**invité**.
- En tant qu'**invité**, je souhaite pouvoir **exercer** sur une porte **les droits** que son propriétaire m'a accordés (autonomie, gérer les invités, etc).

PRIORITÉS 2 (historique, gérer les invités, les autorisations, les interdictions) :

- En tant que **propriétaire**, je souhaite pouvoir **accéder** aux **historiques** de **changements d'états** des portes que je possède.
- En tant que **propriétaire**, je souhaite pouvoir, **à tout instant**, modifier les rôles et les **autorisations** de mes **invités**.
- En tant que **propriétaire**, je souhaite pouvoir, **à tout instant**, ajouter ou supprimer des **utilisateurs** de ma liste d'**invités**.
- En tant que **propriétaire**, je souhaite pouvoir, **à tout instant**, bloquer des **utilisateurs** ou des **invités** de ma porte (Blacklisting) pour leur en interdire **toute interaction**.

PRIORITÉS 3 (faciliter les priorités 2 pour la création de rôles + améliorer l'expérience de l'invité) :

- En tant qu'**invité**, je souhaite pouvoir accéder à la **liste** de portes connectées auxquelles j'ai **déjà** été **invité**.
- En tant qu'**invité**, je souhaite pouvoir observer les **autorisations** que je possède sur **une porte** en particulier.
- En tant que **propriétaire**, je souhaite pouvoir créer des **rôles** qui, sous **un nom**, regroupent une **sélection d'autorisations**.
- En tant que **propriétaire**, je souhaite pouvoir **adresser un rôle** que j'ai **défini** à un **nouvel invité** pour éviter les redondances et **faciliter l'ajout** d'**invités**.

PRIORITÉS 4 (relations d'amitié entre utilisateurs) :

- En tant qu'**utilisateur**, je souhaite pouvoir demander en **ami** d'autres **utilisateurs** via leur ID Lockdoor pour faciliter les interactions entre nous.

PRIORITÉS 5 (modifications de paramètres) :

- En tant qu'**utilisateur**, je souhaite pouvoir télécharger l'**application Lockdoor** pour profiter de toutes les **fonctionnalités** (menu, langues, options, paramètres, profils invités et propriétaires, etc).

II) État de l'art

Les user stories étant définies, notre objectif était de trouver la meilleure manière de développer notre projet. Pour cela nous avons dû passer par des phases de recherche sur des projets **similaires** disponibles sur internet en open source. Ces recherches, en considérant le matériel que nous avions **disponible**, nous ont permis de nous fixer sur les technologies que nous utiliserons pour ce projet à savoir :

- **une carte raspberry pi** : pour avoir accès au Wi-Fi
- **une carte RFID** : pour détecter le scan d'un badge
- **une carte arduino uno** : pour transmettre les trames reçues par la carte RFID à la carte raspberry pi.

Ces technologies nous serviront à simuler la **porte connectée**.

S'ajoutent à cela les technologies internes, c'est-à-dire les technologies liées au **serveur** et à l'**interface web**. En effet, nous avons choisi d'implémenter un serveur afin qu'il traite les données reçues par le raspberry pi mais aussi pour qu'il traite les données reçues par l'interface web et qu'il **synchronise** ces 2 entités. Ces technologies sont les suivantes :

- **language Javascript** : pour le développement du serveur
- **html, css et javascript** : pour le développement de l'interface web
- **base de données SQL** : pour la gestion des utilisateurs de la porte

III) Développement du projet

a) Architecture générale

“En informatique l'architecture est la structure générale inhérente à un système informatique, l'organisation des différents éléments du système (logiciels et/ou matériels et/ou humains et/ou informations) et des relations entre les éléments. Cette structure fait suite à un ensemble de décisions stratégiques prises durant la conception.”[1]
L’architecture du projet LockDoor est composée de **trois entités** que nous allons détailler dans la suite de ce compte rendu :

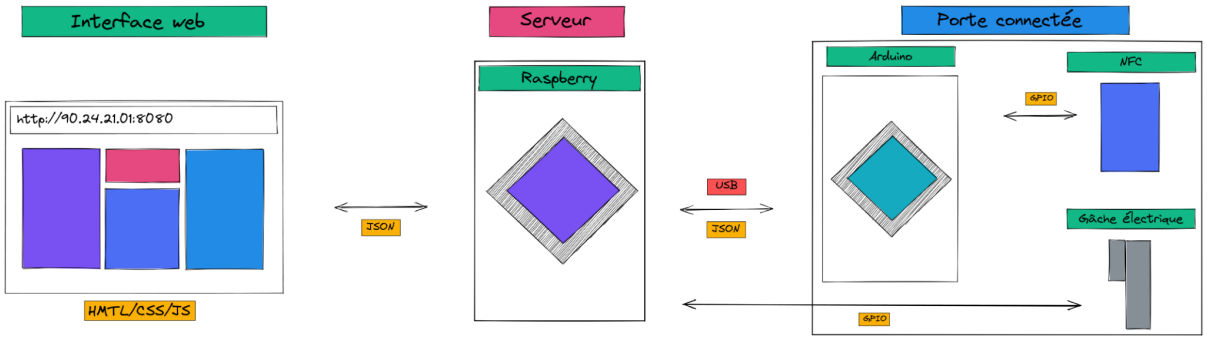


Figure 1 : Architecture globale de LockDoor

Nous avons tout d’abord l’entité Porte connectée, qui représente la maquette de LockDoor, contenant l’ensemble du hardware avec les modules connectés au nano-ordinateur notamment pour **piloter** la porte en local. L’entité serveur, qui permet le stockage de l'interface web, la base de données et veillera à la **synchronisation** entre les éléments **distants** (Interface web) et les éléments **locaux** (porte connectée).
L’entité Interface web à pour rôle, à partir d’un site Internet, de piloter la porte connectée à distance, d’inviter des utilisateurs et de leur donner des droits d’accès.

[1] "Architecture informatique - Définition et Explications." https://www.techno-science.net/glossaire-definition/Architecture-informatique.html. Date de consultation : 2 mai. 2022.

b) Porte connectée

Cette entitée est composée de **2 modules**, à savoir un **lecteur RFID** qui va permettre à l'utilisateur ayant accès à la porte connectée de pouvoir la piloter en badgeant, et la **gâche électrique** qui simule l'ouverture et la fermeture de la porte.

Le lecteur[2] est connecté à une Arduino Uno qui va traiter les données, puis les envoyer à la Raspberry Pi 4 avec une liaison USB[3]. Pour finir, la gâche électrique[4] est connectée à la Raspberry, qui lui transmettra des informations concernant son ouverture et sa fermeture. De plus, le nano-ordinateur a pour rôle de **transmettre** les informations locales sur le site Internet, mais aussi **récupérer** les informations distantes générées à distance avec l'interface web.

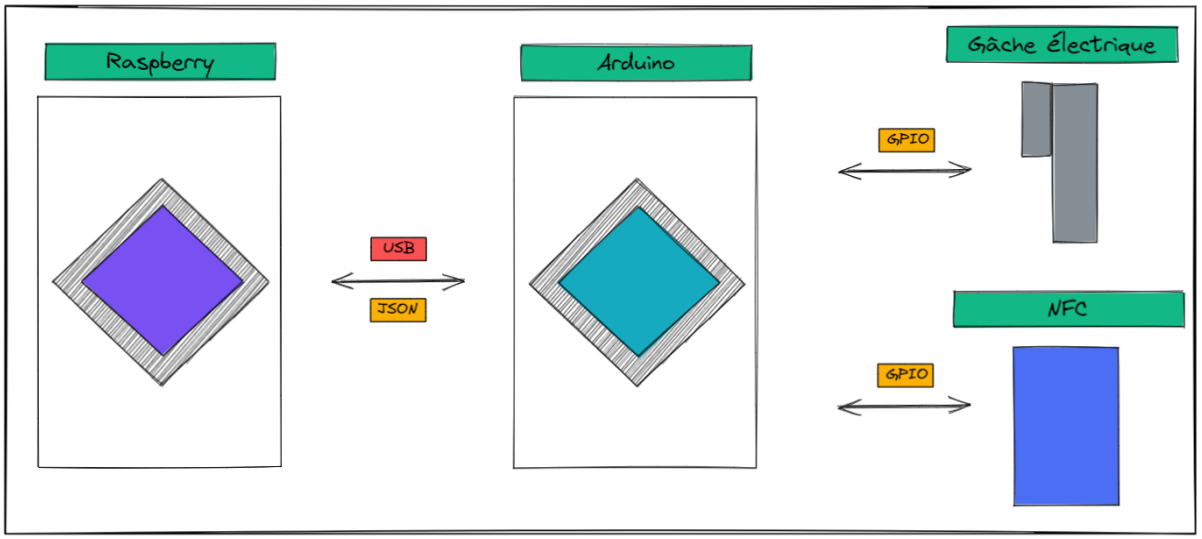


Figure 2 : Zoom sur l'architecture de la porte connectée

Pour illustrer les propos, nous pouvons retrouver sur la figure 2 l'architecture de cette entité.

[2] "MFRC522 RFID Reader With Arduino Tutorial - miliohm.com." 17 juil.. 2020, https://miliohm.com/mfrc522-rfid-reader-with-arduino-tutorial-the-simplest-way-to-read-rfid-tag/ . Date de consultation : 16 mai. 2022.

[3] "Communication série entre Raspberry Pi et Arduino - AranaCorp." 28 mars. 2020, https://www.aranacorp.com/fr/communication-serie-entre-raspberry-pi-et-arduino/. Date de consultation : 16 mai. 2022.

[4] "KY-019 5V Relay Module - ArduinoModulesInfo." 25 mars. 2017, https://arduinomodules.info/ky-019-5v-relay-module/. Date de consultation : 16 mai. 2022.

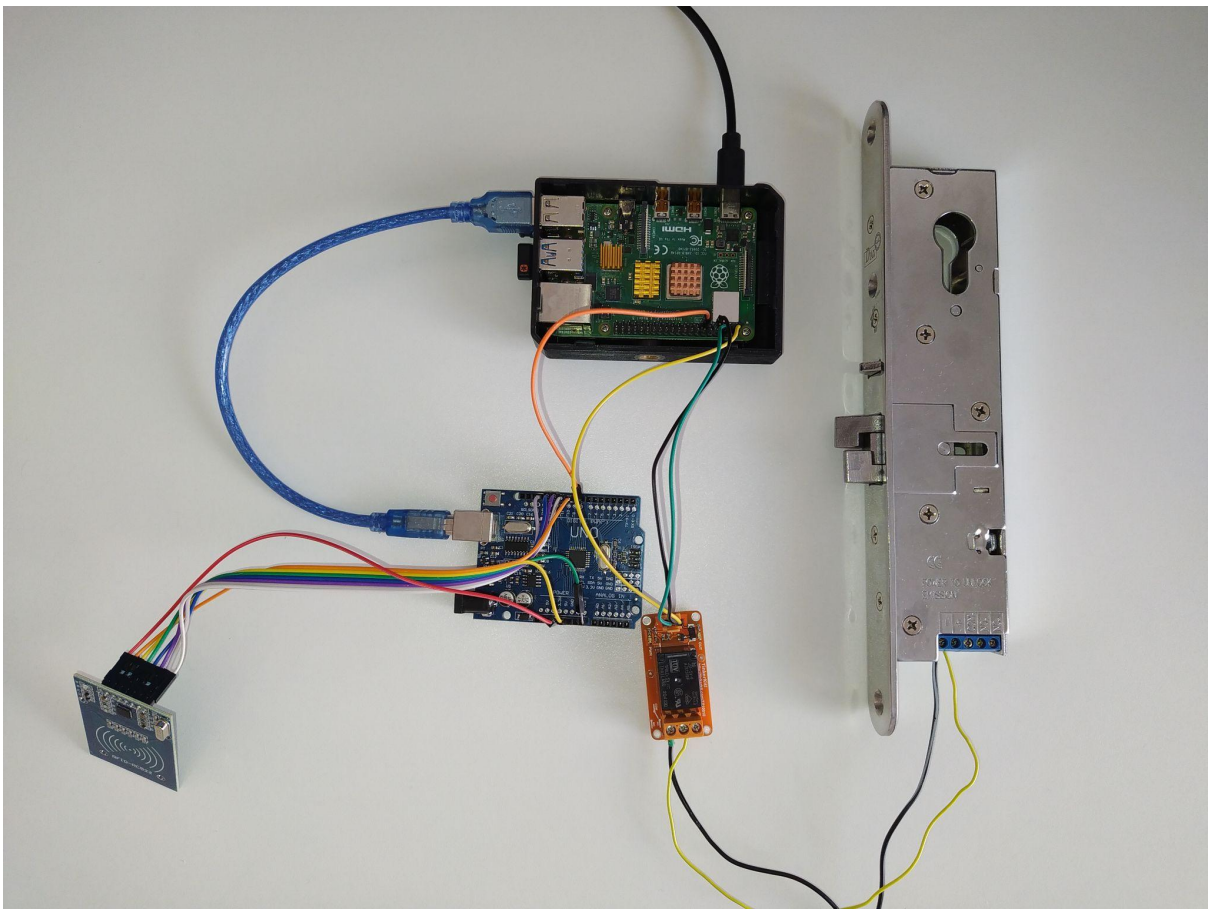


Figure 3 : Maquette de la porte connectée

La figure ci-dessus représente le montage complet de l'entité Porte connectée. Elle est composée d'une **Raspberri Pi 4**, d'une **Arduino Uno**, d'un **module RFID** et d'une **gâche électrique**. Avant d'expliquer le choix et le rôle de chaque composant, nous allons voir le schéma électrique de la maquette sur la figure 4 :

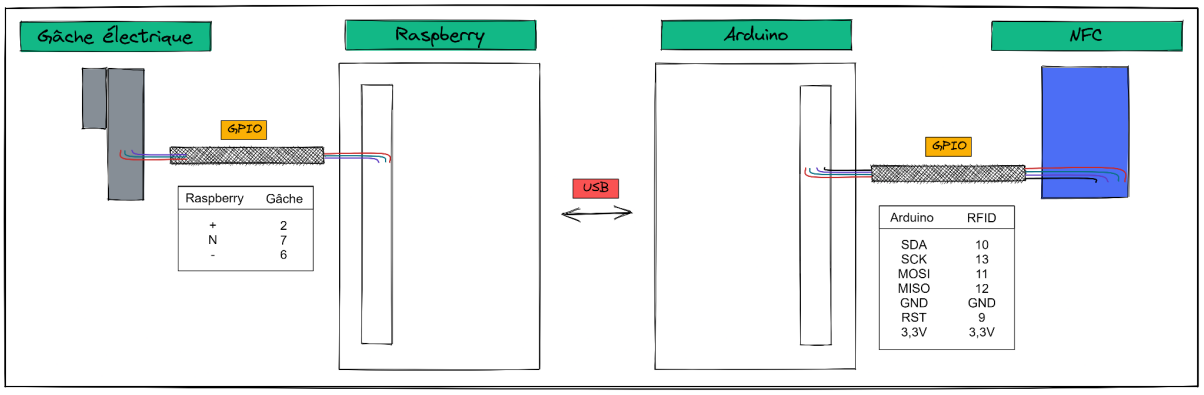
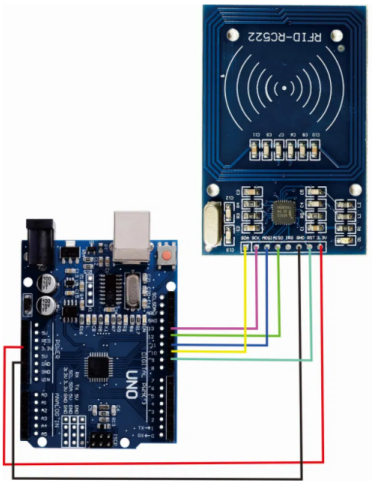


Figure 4 : Schéma électrique global de la porte connectée

Le rôle de l'Arduino Uno est de **récupérer** les informations du module **RFID**, ce qui permettra de **simuler un contact physique** entre l'utilisateur et la porte.

Ayant développé les deux modules, Nous avons préféré cette solution plutôt que le module NFC car le constructeur n'assure plus la maintenance des librairies[5] en plus d'être un shield Arduino et par conséquent ne peut pas s'imbriquer sur la Raspberry Pi si l'on souhaite compresser le montage en supprimant l'Arduino. Nous pouvons voir ci-dessous le schéma électrique plus détaillée de l'Arduino :



Arduino	RFID-RC522
SDA	10
SCK	13
MOSI	11
MISO	12
GND	GND
RST	9
3.3V	3.3V

Figure 5 : Schéma électrique de l'Arduino Uno

Le rôle de la Raspberry Pi 4 est de **récupérer** les informations envoyées par la carte **Arduino Uno**, mais aussi permet de **piloter** le comportement de la gâche électrique par l'intermédiaire d'un relais 12V - 5V puisque la gâche électrique nécessite plus de 5 V pour fonctionner, tension maximale délivrée par la Raspberry. De plus, ce nano-ordinateur **héberge le serveur**, ainsi que la **page Web** et la **base de données** de notre projet. Nous pouvons retrouver ci-dessus le schéma électrique liant la Raspberry, le relais et la gâche :

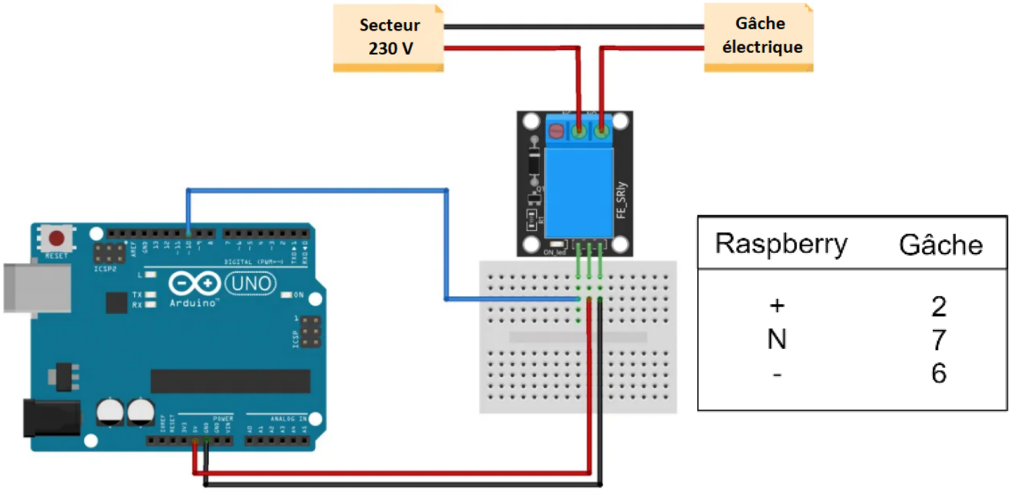


Figure 6 : Schéma électrique de la Raspberry Pi 4

[5] "M24SR64-Y - Dynamic NFC/RFID tag IC with 64-Kbit EEPROM ...." https://www.st.com/en/nfc/m24sr64-y.html. Date de consultation : 22 mai. 2022.

c) Interface Web

Notre application repose sur un client web, dont le **front** est géré en **HTML / CSS** et le **back** en **Javascript**.
Avant de développer le code lui-même, nous avons créé une **maquette** de site via l'outil en ligne **Figma** qui permet de simuler l'aspect du site et le routage de **ses** pages. Cette étape a permis de **structurer** le contenu de la page web et de créer son **ergonomie**, ce qui nous a permis par la suite de gagner du temps sur la phase de développement, aussi bien en front-end qu'en back-end.

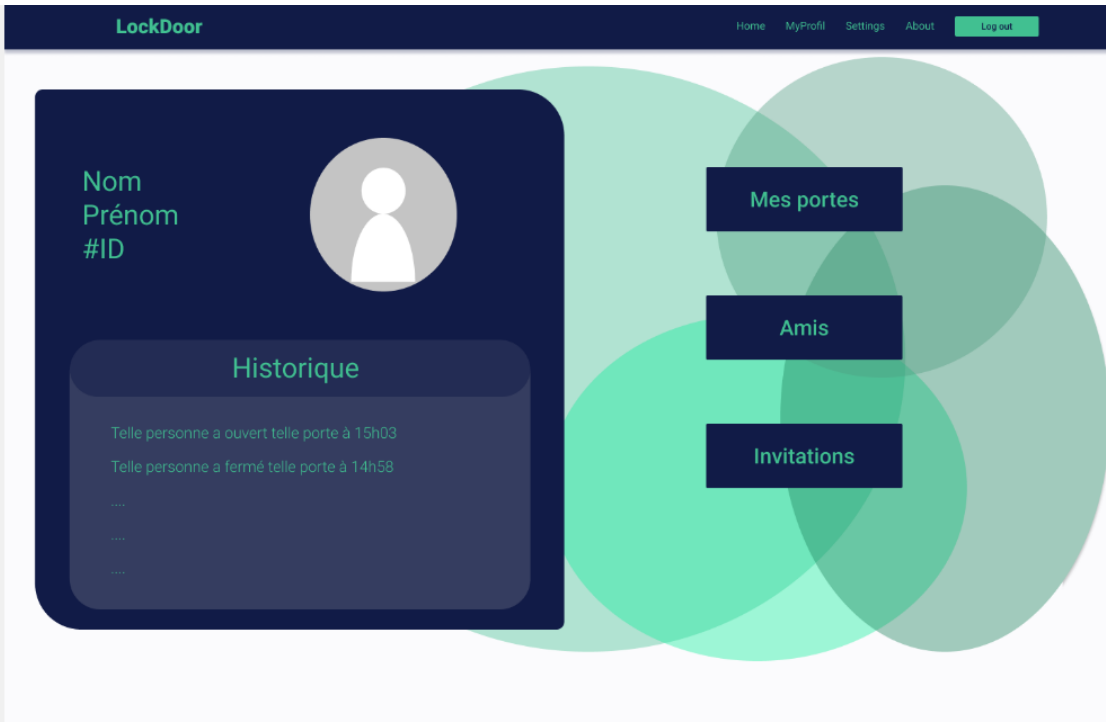


Figure 7 : Maquette page "myProfil" du site web LockDoor

Cette maquette ne sera pas exactement respectée mais son objectif est avant tout de pouvoir donner la structure du site, ses **fonctionnalités**, son **architecture**.

On peut donc se connecter au site en tant qu'utilisateur. Il faut disposer d'un compte LockDoor (identifiant et mot de passe), puis une fois connecté, l'utilisateur peut accéder à un panel de pages dont voici la structure générale :

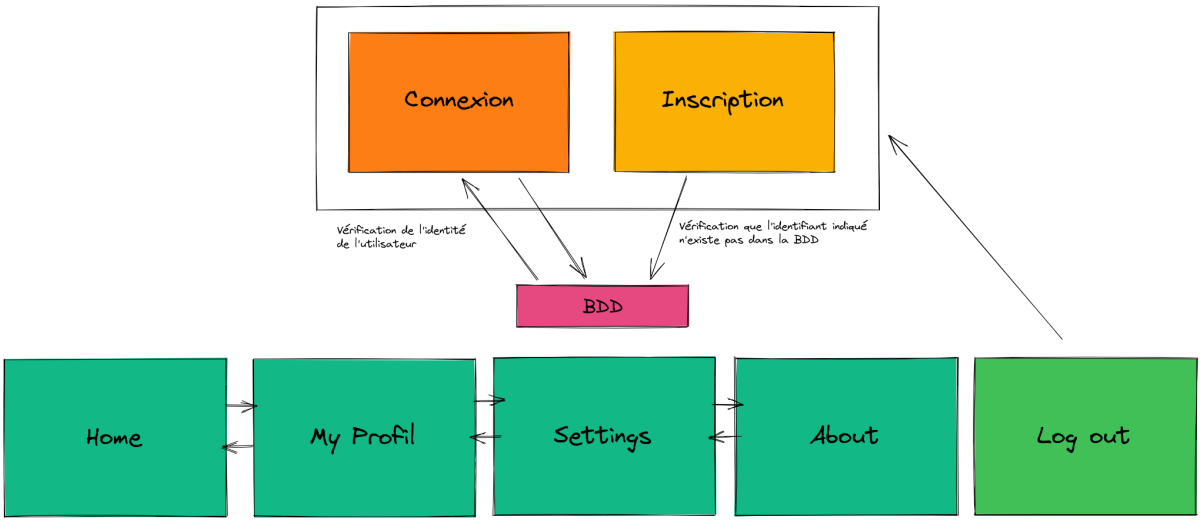


Figure 8 : Structure du site web

Nous allons voir en détail les fonctionnalités que le site nous offre à travers une explication de chaque page.

Tout d'abord, nous arrivons sur une page d'accueil (*cf. Figure 9*) à la racine où l'utilisateur doit choisir entre se connecter s'il a déjà un compte LockDoor ou s'inscrire s'il n'en possède pas.

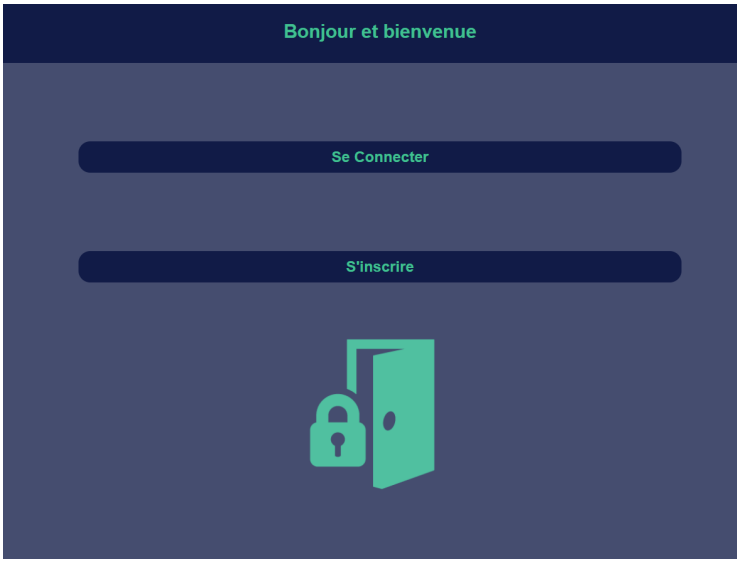


Figure 9 : Page d'accueil

<u>Cas d'une inscription :</u>
Si l'utilisateur décide de s'inscrire il va devoir rentrer en identifiant ainsi qu'un mot de passe puis quand il validera la saisie par l'appuie d'un bouton "submit", les informations qu'il a fournies seront **vérifiées** par l'intermédiaire du **serveur** dans notre **base de données**. Si l'identifiant saisi n'existe pas, l'utilisateur sera rediriger vers la page Home (*cf. Figure 12*). Sinon l'utilisateur restera bloqué sur la page d'inscription.

Figure 10 : Page d'inscription

Cas d'une connexion :
Si l'utilisateur décide se connecter il doit rentrer son identifiant ainsi que son mot de passe. Si les deux informations **correspondent** à celles enregistrées dans la **base de données**, il sera rediriger vers la page Home (*cf. Figure 12*). Sinon il restera bloqué sur la page de connexion.

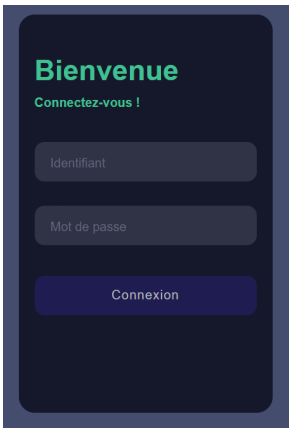


Figure 11 : Page de connexion

Voyons maintenant le détail des pages accessibles sur le site web.

Une fois connecté, l'utilisateur est sur la page principale du site (*cf. Figure 12*). Il aura à disposition une **barre de navigation** lui permettant de passer d'une page à une autre en toute simplicité.

- Page Home : (*cf. Figure 12*)

La page Home est la page principale du site, elle permet de voir **l'état de la porte** (verrouillée ou déverrouillée) visuellement par un voyant (gris : la porte est verrouillée, vert : la porte est déverrouillée). Un bouton à l'intérieur de ce voyant permet de **changer** l'état de la porte. Cette page permet également d'avoir un visuel sur **l'historique** d'ouverture de la porte (utilisateur, action, heure).

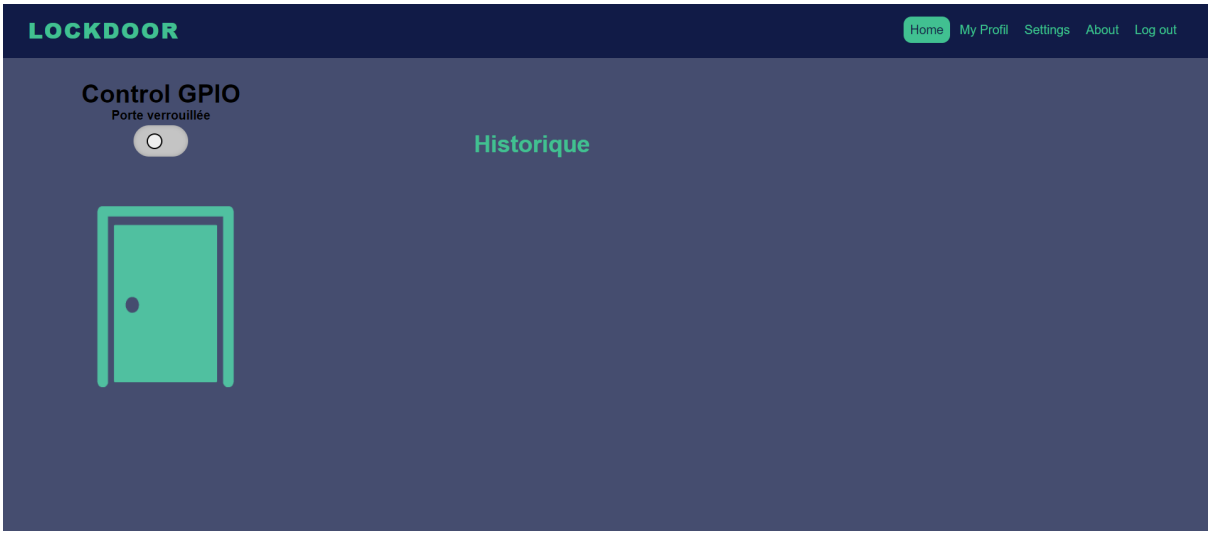


Figure 12 : Page Home

- Page MyProfil : (*cf. Figure 13*)

La page MyProfil permet **d'identifier** la personne connectée et de lui afficher l'historique de **ses actions** sur la porte.

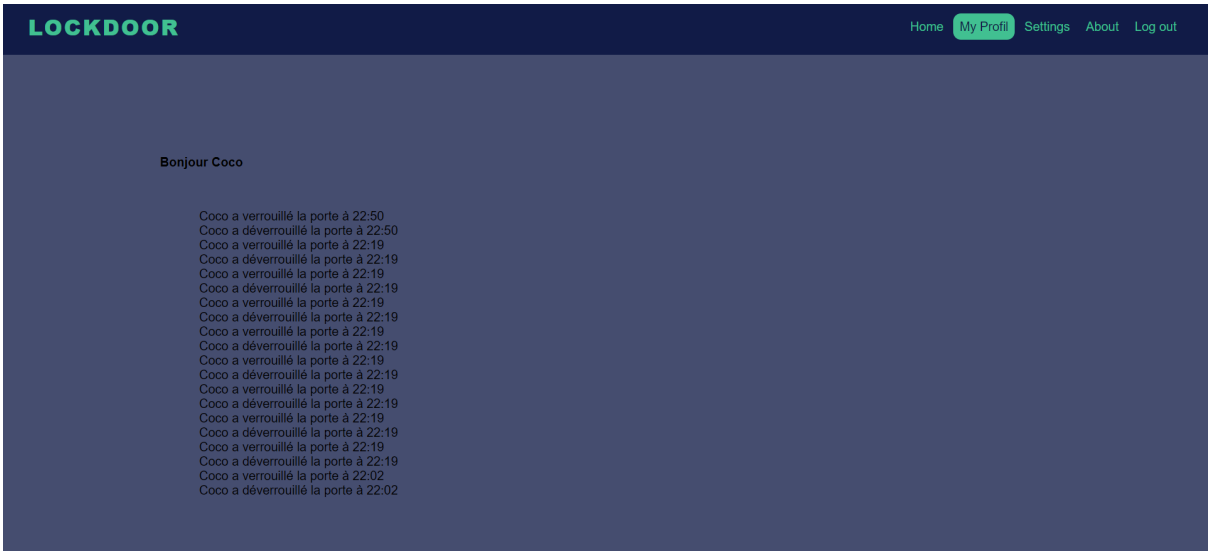


Figure 13 : Page My Profil

- Page Settings : (*cf. Figure 14*)

La page Settings permet au propriétaire de la porte de **modifier** certains paramètres pour **les autres** utilisateurs de la BDD. Le propriétaire va choisir l'utilisateur à qui il veut attribuer des droits d'accès via un menu déroulant. Puis il va avoir le choix entre :

➢ Attribuer tous les droits (l'utilisateur aura le droit de connaître l'état de la porte, de modifier son état localement et à distance.

OU

➢ Attribuer différents droits :
  - Attribuer le droit de connaître l'état de la porte.

ET/OU

  - Attribuer le droit de modifier l'état de la porte localement.

ET/OU

- Attribuer le droit de modifier l'état de la porte à distance.

OU

➢ Supprimer l'utilisateur de la BDD.

En revanche, si on est connecté en tant qu'utilisateur, on ne pourra pas modifier ces paramètres mais **seulement** les consulter.

Figure 14 : Page Settings

- Page About :

La page About est tout simplement une page permettant de détailler les fonctionnalités du site et fonctionnera comme un mode d'emploi du site.

Enfin nous avons un dernier paramètre dans la barre de navigation, un bouton de **déconnexion** (*Log out*) qui renverra vers la **page d'accueil** du site (*cf. Figure 9*). Par conséquent, l'utilisateur sera **déconnecté** de sa session.

Pour cette interface web nous sommes parti d'une maquette que l'on a respecté par rapport à la structure du site, mais où nous avons finalement modifier le contenu des pages. En effet, beaucoup de paramètres et d'illustrations nous ont semblé inutiles pour le développement de l'interface, ce qui nous a conduit à **simplifier** les contenus afin de nous concentrer davantage sur la robustesse et la **fiabilité** du site et non pas sur le design que nous avions imaginé.

La communication entre le serveur et l'interface web se fait par l'intermédiaire de **requêtes HTTP** pour le **routage** et la récupération des pages web (HTML), du style des pages (CSS) et du script associé aux pages (JS), mais aussi de **sockets** pour une **communication synchronisée** avec le serveur grâce à la librairie **socket.io**.

L’envoi de socket s’effectue par la fonction **socket.emit()** tandis que la réception s’effectue par le biais de la méthode **socket.on()**. Comme les sockets client et serveur sont très étroitement liés, nous n’allons détailler que celles actives **côté client**.

Voici la liste des sockets **envoyées** au serveur par le client :

Socket.emit()	**Conditions de l'émission**	**Description de la requête**
‘IWantStateDoor’	home.className == ‘is-active’ → Si on se trouve sur la page Home	Le client souhaite récupérer l’état de la porte (ouverte ou fermée).
‘IWantMyName’	myProfil.className == ‘is-active’ → Si on se trouve sur la page My Profil	Le client souhaite récupérer son nom pour effectuer l’affichage en conséquence
‘getHistory’	Généré après la réponse du serveur à ‘IWantMyName’	Récupérer l’historique des actions de son propre compte
‘IWantAdmin’	settings.className == ‘is-active’ → Si on se trouve sur la page des settings	En fonction du compte actif, faire un affichage adéquat sur la page Settings et autoriser ou non les modifications des droits
‘IWantAccess’	Généré après la réponse du serveur à ‘IWantAdmin’	Savoir si le compte actif peut modifier l’état de la porte
‘setSettings’	Modification des droits utilisateurs depuis la page web	Mise à jour de la BDD
‘gacheT’	Modification de l’état de la porte depuis la page web	Modification physique de l’état et actualisation dans la BDD

Voici la liste des sockets **reçues** par le client :

Socket.on(){ }	**Conditions de la réception**	**Traitement**
‘nomClient’	Avoir émis ‘IWantMyName’, soit être sur la page My Profil.	Actualisation du front et envoi de ‘getHistory’.
‘history’	Avoir émis ‘getHistory’ soit être sur la page My Profil.	Actualisation du front pour afficher l’historique des actions du compte actif.
‘stateDoor’	Avoir émis ‘IWantStateDoor’ soit être sur la page Home.	Mise à jour de la page Home.
‘historyHome’	Avoir émis ‘IWantStateDoor’ et avoir l’autorisation d’accéder à l’historique.	Affichage de l’historique ou non sur la page Home.
‘access’	Avoir émis ‘IWantStateDoor’ et	Savoir si le compte actif peut

	avoir l'autorisation de gérer l'ouverture de la porte.	modifier l'état de la porte et l'affichage sur la page Home.
'noAccessGache'	Avoir émis 'IWantStateDoor' et ne pas avoir l'autorisation de gérer l'ouverture de la porte.	Savoir si le compte actif peut modifier l'état de la porte et l'affichage sur la page Home.
'admin'	Avoir émis 'IWantAdmin' soit être sur la page des Settings.	Savoir si le compte actif peut modifier les droits utilisateurs.
'pseudoNotFound'	Essayer de modifier les droits d'un utilisateur.	Savoir que l'utilisateur saisi n'existe pas pour l'afficher sur la page Settings.
'pseudoFound'	Essayer de modifier les droits d'un utilisateur.	Savoir que l'utilisateur saisi existe pour l'afficher sur la page Settings.
'deleteUser'	Suppression d'un utilisateur.	Savoir si l'utilisateur a bien été supprimé pour l'afficher sur la page Settings.
'gache'	L'état de la porte a été modifié à distance.	Affichage de l'état de la porte sur la page Home.
'gacheLocal'	L'état de la porte a été modifié en local (RFID).	Affichage de l'état de la porte sur la page Home.

d) Serveur

La Raspberry Pi joue **à la fois** le rôle du client et du serveur web[6] afin de stocker l'interface web et la base de données. Pour cela, nous avons configuré un serveur web **Apache**, qui permet entre autres de servir des fichiers HTML via le protocole web HTTP et HTTPS. On comptabilise 44% de serveurs web Apache dans le monde, et c'est le plus populaire disponible pour le Raspberry, notre choix s'est donc naturellement penché sur ce logiciel.

Nous avons mis en ligne[7] le serveur web de la Raspberry Pi afin que les utilisateurs puissent y avoir accès **depuis Internet**. Pour cela, nous avons tout d'abord rediriger les requêtes reçus par la box où l'on a ouvert les ports vers la Raspberry :

Type de requête	Port externe de la requête	Port interne de la redirection	Protocole employé	Équipement cible
HTTP	80	80	TCP	raspberry*
HTTPS	443	443	TCP	raspberry*

Figure 15 : Configuration de la redirection des requêtes

[6] "Mise en place d'un serveur web Apache sur Raspberry Pi." 28 sept.. 2020, https://www.raspberrypi-france.fr/mise-en-place-dun-serveur-web-apache-sur-raspberry-pi/. Date de consultation : 16 mai. 2022.

[7] "Rendre votre Raspberry Pi accessible depuis internet avec DynDNS ...." 23 déc.. 2017, https://raspberry-pi.fr/mettre-en-ligne-serveur-web-raspbian-dydns-port-forwarding/. Date de consultation : 23 mai. 2022.

Comme nous pouvons le voir sur la figure 15, nous avons **redirigé** les requêtes web avec le port 80 pour le protocole HTTP et le port 443 pour le protocole HTTPS vers l'équipement cible, ici raspberry. La figure 16 représente un exemple de comportement de la box :

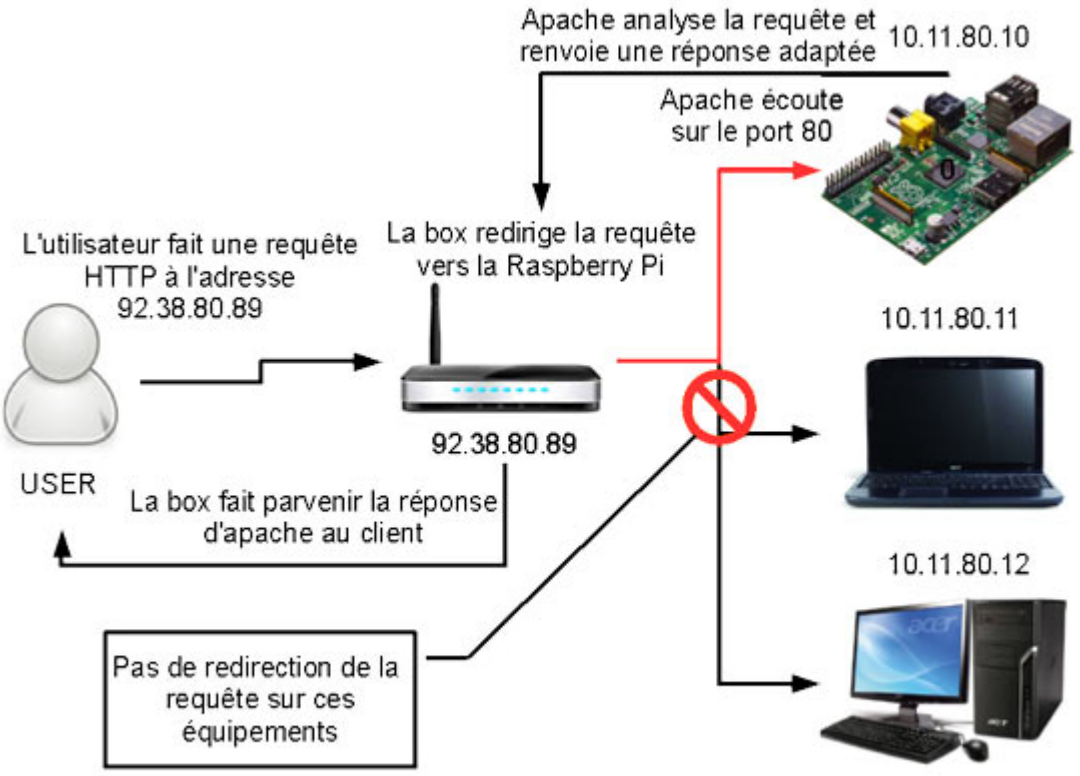


Figure 16 : Schéma du comportement de la box avec les requêtes

De plus, nous avons implémenté un **WebSocket**[8] qui permet une communication bidirectionnelle en temps réel sur le web.
En effet, en cliquant sur le bouton qui permet de changer l'état de la porte sur l'interface web, cela va activer ou désactiver un GPIO (broche) sur la Raspberry Pi et changer l'état de la porte connectée. De plus, les utilisateurs présentant leur badge devant le lecteur feront changer l'état de la porte, qui sera aussi mis à jour sur le site web. Pour résumer, le WebSocket permet une communication bi-directionnelle entre les entités Interface Web et Porte connectée le tout en temps réel.

Description détaillée de son contenu

Le serveur a été codé en langage **JavaScript** avec l'utilisation de la librairie ***"express"***. Le routage des pages du site web se fait par l'intermédiaire de requêtes **HTTP GET et POST** du client. Nous utilisons seulement les requêtes POST dans le cas où le client a besoin de **transmettre** des données au serveur (cas d'un connexion ou d'une inscription). Toutes les autres requêtes seront des GET, à savoir le routage des pages html, le script de chaque page, codé en javascript et le style des pages grâce à des codes en css.

[8] "Node.js Raspberry Pi Webserver with WebSocket - W3Schools." https://www.w3schools.com/nodejs/nodejs_raspberrypi_webserver_websocket.asp. Date de consultation : 16 mai. 2022.

Le serveur communique avec le client par le biais de sockets grâce à la librairie socket.io pour javascript. A chaque fois qu'un client sera connecté, le serveur ouvrira un socket pour ce client et **attendra des évènements** produits par le client afin de traiter ses requêtes. Le serveur met fin à la communication avec le socket actuel et en crée un nouveau **à chaque fois** que le client rafraîchit une page ou qu'il change de page. En effet, à chaque fois que le client demande au serveur une page html de la barre de navigation du site, un script javascript est exécuté. Ce script permet entre autres d'ouvrir un socket à chaque fois qu'il est appelé. c'est pourquoi, lors d'une même session, le serveur est amené à communiquer avec de **multiples sockets**.

Le serveur permet également la gestion de la base de données à travers des fonctions asynchrones qui vont nous servir à aller chercher les données dont on a besoin selon les cas de traitements.

Diagramme UML initial

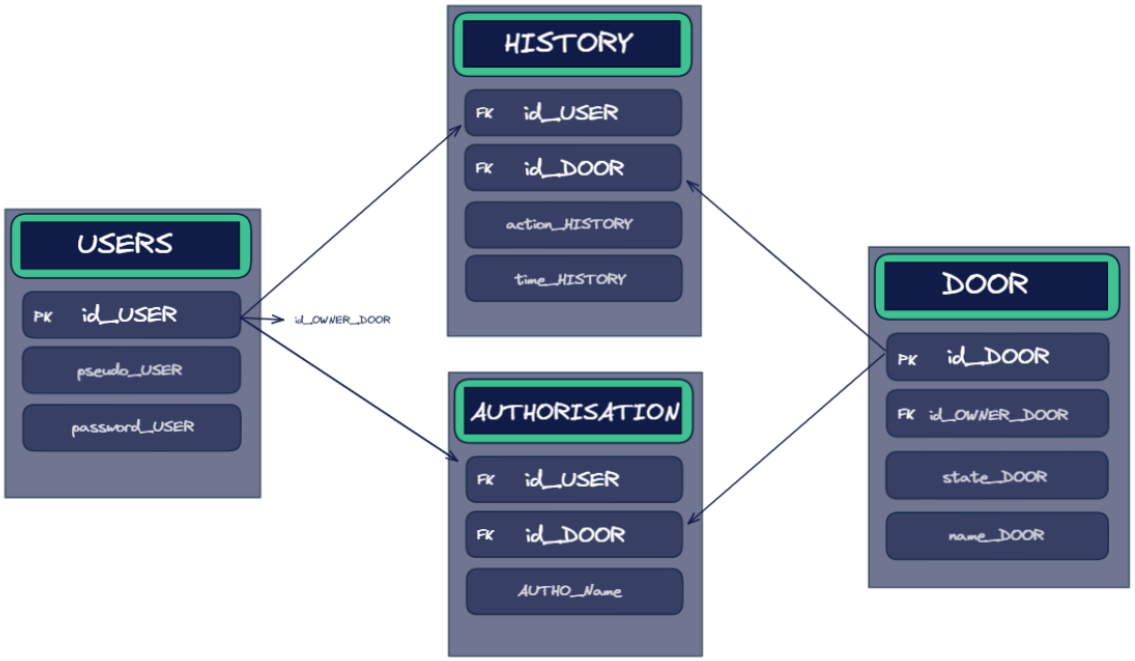


Figure 17 : Schéma fonctionnel de la base de données générale

Ce diagramme répond au fonctionnement de l'application dans le cas d'un développement plus avancé, avec des fonctionnalités supplémentaires notamment liées à la **multiplicité des portes**.
Un **'USER'** est défini par son identifiant, son pseudo et son mot de passe.
Une **'DOOR'** est définie par son identifiant, l'identifiant de son propriétaire, son état, et son nom.
La table **'HISTORY'** lie les deux tables précédentes, pour y rentrer les actions que l'on souhaite sauvegarder des utilisateurs sur une porte.

La table **‘AUTHORISATION’** lie également les tables ‘user’ et ‘door’, et on rentrera les droits de chaque user dans de multiples autorisations ‘AUTHO_Name’ renseigné par une chaîne de caractères.
Cela permet de répondre aux contraintes **complexes** notamment **temporelles** définies dans le **cahier des charges** comme l’autorisation de l’ouverture selon des plages horaires, des jours hebdomadaires ou calendaires, etc…

Ces AUTHO pourraient prendre une forme telles que :

- *coco, add, admin, garage :* ajoute l’utilisateur coco en tant qu’admin (tous les droits) sur la porte du garage.
- *mathis, local, remote, coffrefort :* donne à mathis l’autorisation de contrôler la porte coffre-fort aussi bien à distance qu’en local.
- *theo, local, villa, 01/06/2022, 15/06/2022, 8h, 10h :* donne à theo l’autorisation d’ouvrir la villa en local du 01 juin 2022 au 15 juin 2022 entre 8h et 10h pour qu’il effectue l’entretien de la villa.

NB : ce diagramme ne permet pas de répondre aux priorités 4 et 5 des User Stories, pour lesquelles on aurait par exemple pu rajouter des tables “friendship” et “account_settings”.

Diagramme UML implémenté

En réalité, la base de données précédente est trop complexe pour une première implémentation. Pour **accélérer** les phases de **développement**, **d’intégration**, **de test**, nous avons réduit la BDD pour notre utilisation particulière dans laquelle nous n’avons qu’une gâche connectée soit une porte connectée. De plus, nous nous sommes rendus compte que nous n’aurions pas le temps d’implémenter **l’ensemble** des User Stories.

Pour ces deux raisons, nous avons réalisé une nouvelle base de données, plus simple et légère, qui offre forcément moins de possibilités, mais qui sera plus facile à traiter.

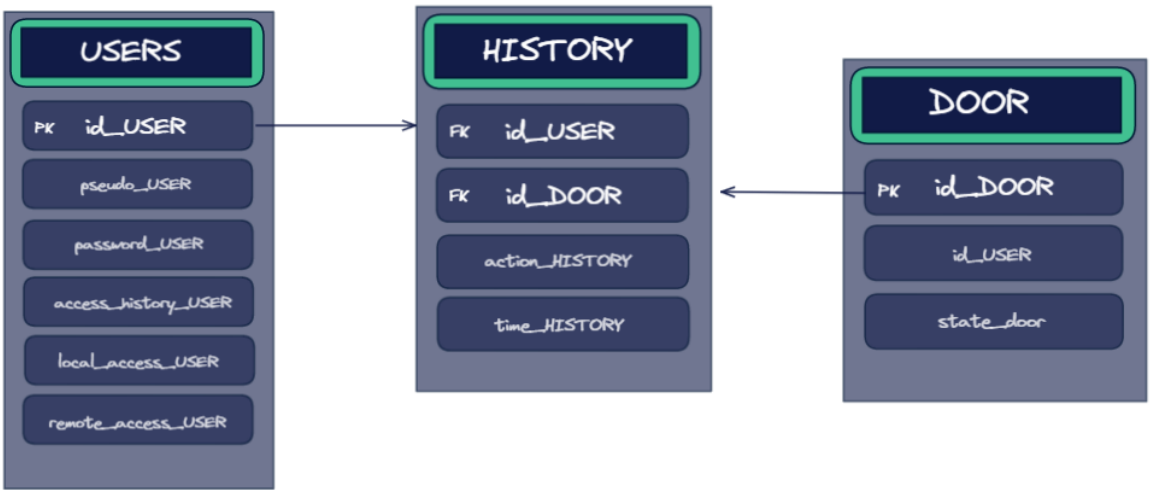


Figure 17 : Schéma fonctionnel de la base de données implémentée

L'unicité de la porte apporte de nombreuses **simplifications**. De plus, nous avons vu les autorisations des utilisateurs à la baisse. Nous n'avons pas laissé les possibilités les plus complexes, ainsi elles peuvent toutes être exprimées par oui ou par non, soient des **attributs booléens** (access_history_USER, local_access_USER, remote_access_USER). La table 'authorisations' a donc été remplacée par des booléens propres à chaque utilisateur, qui ne pourront être valables que sur l'unique porte.

NB : En fait, la BDD aurait pu être encore réduite, puisque la porte n'a plus besoin d'identifiant, ni de nom. Mais il paraissait important de garder une certaine **logique de structure** en vue d'un **éventuel développement** du projet.

Enfin, voici un schéma explicitant le fonctionnement global du projet : la partie client, la partie serveur, et leurs manières de communiquer les informations :

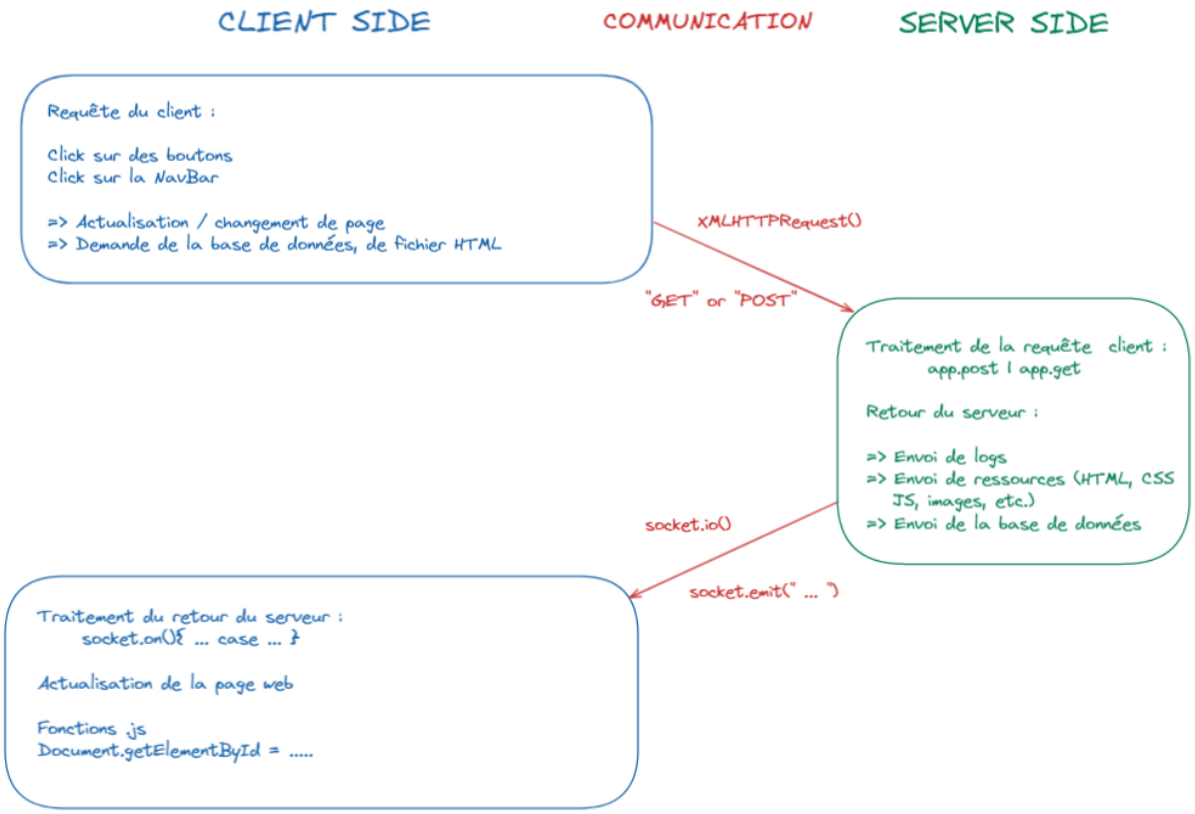


Figure 18 : Schéma illustrant la communication entre le serveur et le client

IV) Prise de recul sur les objectifs initiaux (User Stories)

Après avoir atteint la limite de temps accordée à ce projet, il paraît intéressant d'observer à quel point les objectifs du cahier des charges ont été remplis. Revoyons donc les User Stories par ordre de priorité.

PRIORITÉS 1 (compte et ID, existence des rôles propriétaire, invité) :

(100%, 7/7)

- **DISPONIBLE :** En tant qu'**utilisateur**, je souhaite pouvoir me créer un **compte Lockdoor** pour avoir une **identité unique** vis-à-vis de l'application et des autres utilisateurs.
- **DISPONIBLE :** En tant qu'**utilisateur**, je souhaite pouvoir **appareiller** mon smartphone à une porte connectée que je viens d'acheter pour en devenir le **propriétaire** à la **1ère utilisation**.
- **DISPONIBLE :** En tant que **propriétaire**, je souhaite pouvoir **gérer** l'état des portes que je possède.
- **DISPONIBLE :** En tant que **propriétaire**, je souhaite pouvoir **inviter** des **utilisateurs** sur ma porte pour leur octroyer le profil d'**invité**.
- **DISPONIBLE :** En tant que **propriétaire**, je souhaite pouvoir **définir** des **autorisations personnalisables** à **chacun** de mes invités.
- **DISPONIBLE :** En tant qu'**utilisateur**, je souhaite pouvoir **me faire ajouter** sur des portes connectées pour obtenir le rôle d'**invité**.
- **DISPONIBLE :** En tant qu'**invité**, je souhaite pouvoir **exercer** sur une porte **les droits** que son propriétaire m'a accordés (autonomie, gérer les invités, etc).

PRIORITÉS 2 (historique, gérer les invités, les autorisations, les interdictions) :

(75%, 3/4)

- **DISPONIBLE :** En tant que **propriétaire**, je souhaite pouvoir **accéder** aux **historiques** de **changements d'états** des portes que je possède.
- **DISPONIBLE :** En tant que **propriétaire**, je souhaite pouvoir, **à tout instant**, modifier les rôles et les **autorisations** de mes **invités**.
- **DISPONIBLE :** En tant que **propriétaire**, je souhaite pouvoir, **à tout instant**, ajouter ou supprimer des **utilisateurs** de ma liste d'**invités**.
- **INDISPONIBLE :** En tant que **propriétaire**, je souhaite pouvoir, **à tout instant**, bloquer des **utilisateurs** ou des **invités** de ma porte (Blacklisting) pour leur en interdire **toute interaction**.

PRIORITÉS 3 (faciliter les priorités 2 pour la création de rôles + améliorer l'expérience de l'invité) :

(0%, 0/4)

- **INDISPONIBLE :** En tant qu'**invité**, je souhaite pouvoir accéder à la **liste** de portes connectées auxquelles j'ai **déjà** été **invité**.

- **INDISPONIBLE :** En tant qu'**invité**, je souhaite pouvoir observer les **autorisations** que je possède sur **une porte** en particulier.
- **INDISPONIBLE :** En tant que **propriétaire**, je souhaite pouvoir créer des **rôles** qui, sous **un nom**, regroupent une **sélection d'autorisations**.
- **INDISPONIBLE :** En tant que **propriétaire**, je souhaite pouvoir **adresser un rôle** que j'ai **défini** à un **nouvel invité** pour éviter les redondances et **faciliter l'ajout** d'**invités**.

PRIORITÉS 4 (relations d'amitié entre utilisateurs) :

(0%, 0/1)

- **INDISPONIBLE :** En tant qu'**utilisateur**, je souhaite pouvoir demander en **ami** d'autres **utilisateurs** via leur ID Lockdoor pour faciliter les interactions entre nous.

PRIORITÉS 5 (modifications de paramètres) :

(0%, 0/1)

- **INDISPONIBLE :** En tant qu'**utilisateur**, je souhaite pouvoir télécharger l'**application Lockdoor** pour profiter de toutes les **fonctionnalités** (menu, langues, options, paramètres, profils invités et propriétaires, etc).

Nous observons qu'une bonne partie des user stories a été un **succès**. De plus, les réussites suivent parfaitement **l'ordre de priorité** que nous nous étions fixés.

A partir de l'ordre de **priorité 3**, les user stories demandent toutes la **multiplicité** des portes. Un travail conséquent serait alors nécessaire. Comme expliqué précédemment, il faudrait, en plus d'implémenter de nouvelles fonctionnalités plus complexes, **restructurer** la base de données et **redéfinir** les éléments disposés sur les pages HTML en conséquence, ce qui impliquerait aussi de nombreuses **modifications** sur le travail déjà effectué.

Conclusion

Nous pouvons donc en conclure que ce projet, de part sa durée et son contexte, nous a permis de renforcer nos compétences **humaines** en appliquant la méthode **AGILE** afin de faire des points réguliers sur l'avancement de chaque membre du projet, mais aussi en utilisant des **outils de gestion** comme GitHub, GANTT ou encore Notion.

De plus, nous avons acquis des compétences **techniques** pour développer chaque entité du projet avec la réalisation de maquette matériel, mais aussi le développement du code pour le serveur, l'interface web et les modules composant la porte connectée.

Les fonctionnalités du projet étant intégralement implémentées, l'ajout du module sécurité implémentés dans chaque entités afin de chiffrer la base de données ainsi que la communication à travers le WebSocket est un élément important qui pourra être ajouté dans le futur pour concrétiser LockDoor. Etant donné que le prototype à été développé pour le fonctionnement qu'avec une seule porte connectée, la déportation du serveur Web sur un serveur de l'ISTIC est une solution à envisager pour assurer un bon fonctionnement avec l'implémentation de plusieurs portes connectées de propriétaires différents, dans ce cas la Raspberry Pi servira de portail assurant la réception, le changement et l'envoi de l'état de la gâche électrique au serveur de l'ISTIC qui héberge l'interface web.

Annexes

"Communication série entre Raspberry Pi et Arduino • AranaCorp." *AranaCorp*, 28 March 2020, https://www.aranacorp.com/fr/communication-serie-entre-raspberry-pi-et-arduino/. Accessed 16 May 2022.

"Définition d'architecture informatique." *techno-science.net*, https://www.techno-science.net/glossaire-definition/Architecture-informatique.html. Accessed 2 May 2022.

"KY-019 5V Relay Module." *ArduinoModules*, 25 March 2017, https://arduinomodules.info/ky-019-5v-relay-module/. Accessed 16 May 2022.

"M24SR64-Y - Dynamic NFC/RFID tag IC with 64-Kbit EEPROM, NFC Forum Type 4 Tag and I2C interface." *STMicroelectronics*, https://www.st.com/en/nfc/m24sr64-y.html. Accessed 22 May 2022.

"MFRC522 RFID with arduino tutorial." *miliohm.com*, 17 July 2020, https://miliohm.com/mfrc522-rfid-reader-with-arduino-tutorial-the-simplest-way-to-read-rfid-tag/. Accessed 16 May 2022.

"Mise en place d'un serveur web Apache sur Raspberry Pi." *Raspberry Pi France*, 28 September 2020, https://www.raspberrypi-france.fr/mise-en-place-dun-serveur-web-apache-sur-raspberry-pi/. Accessed 16 May 2022.

"Node.js Raspberry Pi Webserver with WebSocket." *W3Schools*, https://www.w3schools.com/nodejs/nodejs_raspberrypi_webserver_websocket.asp. Accessed 16 May 2022.

"Rendre votre Raspberry Pi accessible depuis internet avec DynDNS et le port Forwarding." *Raspberry Pi FR*, 23 December 2017,

https://raspberry-pi.fr/mettre-en-ligne-serveur-web-raspbian-dydns-port-forwarding/. Accessed 23 May 2022.