Institut National de Radioélectricité et de Cinématographie

Enseignement technique secondaire de qualification

Accès aux études supérieures

Avenue Jupiter, 188

1190 Forest

Borne de commande avec infrastructure sécurisée

Projet personnel de Chakri Ayoub

Pour l'obtention du certificat de qualification

Technicien en informatique

Tuteur : Mr I. Kajjoua

Année scolaire : 2025-2026

Remerciements :

Tout d’abord avant de pouvoir rédiger ce rapport, je voudrais remercier certaines personnes qui m’ont aidé au fil de cette longue année à réaliser ce travail de fin d’études.

Mon professeur principal, qui est Monsieur Kajjoua, et qui m’a guidé du début jusqu’à la fin sur ce que je pouvais améliorer ou ajouter à mon projet, ces conseils m’ont été en tous cas d’une grande utilité et je le remercie énormément.

Mes profs d’informatique, qui m’ont appris beaucoup de choses durant ces 3 dernières années, leurs connaissances m’ont permis de réaliser au mieux mon projet.

Sans oublier bien sûr mes parents, qui m’ont toujours aidé lorsque je rencontrais des difficultés et qui ont contribué au financement de mes composants de mon projet.

Table des matières :

1. Introduction

A l’heure d’aujourd’hui, plusieurs petites restaurations recherchent des bornes de commandes intelligentes avec des infrastructures sécurisées accessibles sur n’importe quel téléphone à un prix abordable pour mieux promouvoir et gérer leurs gestions de commandes sur place dans les restaurants, et donc pour diminuer le temps d’attente à la caisse pour les clients.

A l'aide de l'informatique, on peut facilement créer et utiliser des composants à disposition pour pouvoir réaliser une borne de commande. C'est tout à fait pour répondre aux besoins de petites restaurations qui débutent que j'ai décidé de réaliser cette borne de commande.

Mon TFE consiste à créer une borne de commande pour les petites restaurations qui débutent et qui n’ont pas forcément un budget élevé pour pouvoir se payer une borne de commande avec une infrastructure sécurisée et qui est accessible directement via le web. Ma borne de commande fonctionne alors avec une application web développée avec du Python (FLASK), deux Raspberry Pi 4, un esp32 et un capteur de mouvement.

Le premier Raspberry Pi 4 sert de serveur pour pouvoir héberger ma borne de commande, ainsi que le deuxième Raspberry Pi 4 sert de backup de données. J’utilise également un esp32 qui envoie les données du capteur de mouvement toutes les 5 secondes à la borne de commande pour savoir si la table du restaurant est libre ou pas.

2. Les Outils

2.1 Le matériel utilisé

2.1.1 Le prix du matériel

Voici la liste du matériel nécessaire à mon projet ainsi que leur prix pour chaque :

Matériel	**Description**	**Quantité**	**Prix**
Raspberry Pi 4 (pack)	Modèle B, 8GB de RAM	2	317.06€
ESP32 (pack)	Modèle WLAN	1	20.00€
Capteur PIR	Modèle SR501	1	8.09€
Câbles de connexion	Mal vers Femelle	4	5.00€
Écran tactile	7 pouces	1	25.16€
Structure 3D	Rouleau PLA	1	15.00€

Le prix du matériel que j'utilise m'est revenu au prix de 390.31€ pour pouvoir réaliser mon projet de travail de fin d'études.

2.1.2 Les Raspberry Pi 4

Qu’est-ce qu’un Raspberry Pi 4 ?

En 2012, il a été créé dans le but d'attirer les jeunes pour qu'ils puissent apprendre de l'informatique, mais à l'heure d'aujourd'hui ce produit s'est répandu jusqu'à même des développeurs professionnels. Il peut s'utiliser d'une manière classique, donc avec un écran, clavier et souris, mais avec des logiciels sur ordinateur, on peut l'utiliser à distance sans clavier et souris.

Le Raspberry Pi 4 contient 1 port Ethernet, 4 ports USB, 1 port jack, 2 sorties micro HDMI, un microprocesseur, une sortie micro SD, et un port DSI pour l’affichage et CSI pour pouvoir utiliser une caméra. Ainsi que des pins GPIO, GND et d’alimentation.

Implantation dans mon projet :

Dans mon projet j’ai utilisé 2 Raspberry Pi 4 :

Mon premier Raspberry Pi 4 permet d'héberger ma borne de commande, il envoie des requêtes à Cloudflared pour pouvoir afficher la borne sur mon nom de domaine grâce au tunnel que j'ai mis en place entre mon Raspberry Pi 4 et le nom de domaine. C'est beaucoup plus intéressant d'héberger ma borne de commande dessus, au lieu de payer pour un serveur VPS en ligne chaque mois. Quant à mon 2ème Raspberry Pi, je l'ai utilisé pour pouvoir faire de la redondance de données. De plus, j'ai configuré un partage de dossier via le protocole SAMBA pour que mon 1er Raspberry Pi 4 puisse envoyer les données de ma borne de commande au dossier Samba sur le 2ème Raspberry Pi 4.

2.1.3 Le ESP32

Qu’est-ce qu’un ESP32 ?

C'est un microcontrôleur très petit qui est doté du Wifi. Il permet d'exécuter des codes simples (comme en Micro-Python) de manière répétée pour pouvoir contrôler de petits composants tels que des capteurs PIR, des servomoteurs, etc. Il ne consomme presque aucune énergie.

Ce petit microcontrôleur est doté d'un processeur double cœur, de 520 Ko de SRAM, de plusieurs PIN GPIO, GND, d'alimentation 5V et 3.3V, sans oublier qu'il a un capteur de température interne.

On peut l'utiliser facilement en le connectant à un PC via un câble qu'on branche sur le port USB du PC et sur le port USB-C de l'ESP32. Une fois qu'il est connecté au PC, on utilise le logiciel Thonny IDE pour pouvoir coder dessus, et ensuite enregistrer le fichier sur sa mémoire.

Implantation dans mon projet :

Dans mon projet, l'ESP32 permet d'envoyer les informations du capteur PIR en temps réel à la borne de commande pour que la borne sache automatiquement quelle table est libre dans le restaurant et pour pouvoir l'attribuer au client quand il sélectionne l'option "Sur place".

2.1.4 Le capteur PIR

Qu’est-ce qu’un capteur PIR ?

Le capteur PIR est un petit capteur qui est capable de pouvoir détecter les mouvements ; il fonctionne en détectant le taux de chaleur corporelle qui se dégage d'un être humain. Contrairement au capteur à ultrason, il n'émet aucun signal, le capteur PIR se base uniquement sur la chaleur émise d'un humain.

Le capteur PIR est composé de 3 éléments très simples qui sont le dôme blanc qui permet de capter toute la chaleur de la pièce, un détecteur central qui détecte la chaleur humaine et, pour finir, une petite carte électronique qui envoie les alertes de mouvement.

Pour pouvoir utiliser un capteur PIR, il doit soit être relié avec des câbles mal femelles à un esp32 ou à un Raspberry Pi, pour qu'on puisse le configurer avec un code micro-python pour qu'ils puissent fonctionner.

Implantation dans mon projet :

Dans mon projet, le capteur PIR permet de détecter les mouvements à une table pour pouvoir attribuer à chaque table du restaurant un statut. Il y a 2 statuts : "libre" et ""occupé". Le statut est réinitialisé chaque 5 secondes et est envoyé à l'ESP32 qui va envoyer l'information du statut à la borne de commande.

2.1.5 Les câbles GPIO

Qu'est-ce qu'un câble GPIO ?

Un câble GPIO qui peut s'appeler également un câble Dupont est un simple fil électrique. Il a 2 embouts différents, un mâle et une femelle, qui permettent de relier 2 composants électroniques facilement sans devoir souder sur une breadboard.

Implantation dans mon projet :

Dans mon projet, le câble GPIO permet de transporter le courant et les données entre mon capteur PIR et mon ESP32.

2.1.6 L'écran tactile

Qu'est-ce qu'un écran tactile ?

L'écran tactile est un périphérique d'affichage qui permet d'interagir avec un ordinateur, une tablette ou bien même un téléphone, tant qu'il est lié à un appareil via un câble HDMI. L'écran tactile est composé d'une dalle LCD et qui est recouvert d'une vitre pour que ça puisse être tactile.

Implantation dans mon projet :

Dans mon projet, l'écran tactile permet au client d'interagir directement avec la borne de commande de manière très fluide, qui est mise en mode kiosk. L'écran tactile est alors relié à mon premier Raspberry Pi 4 avec un câble HDMI.

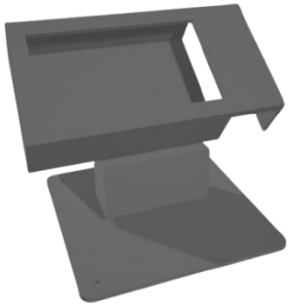
2.1.7 La structure 3D en PLA

Qu’est-ce que le PLA ?

Le PLA (Acide Polylactique) est un filament utilisé généralement pour les domaines, comme l'impression 3D, les emballages alimentaires, les sacs plastiques, etc.

Le PLA est fabriqué à base de ressources renouvelables, notamment avec de l'amidon de maïs, du manioc ou encore de la canne à sucre.

Implantation dans mon projet :



Dans mon projet, j'ai utilisé le PLA pour une raison précise qui est celle de pouvoir imprimer ma structure 3D de ma borne de commande.

J'ai utilisé le PLA car il offre beaucoup d'avantages qui sont les suivants :

- Facile à utiliser (il se place facilement dans les imprimantes 3D standard).
- Peu d'échec (il s'imprime avec pas trop de déformations).
- Le rendu final (le rendu à la fin de l'impression est très lisse et beau à voir).

2.2 Les logiciels utilisés

2.2.1 Le prix des logiciels

Voici la liste des logiciels nécessaires à mon projet ainsi que leur prix pour chaque :

Nom	**Description**	**Prix**
Thonny IDE	Environnement de développement sur Raspberry Pi 4	0.00€
Visual Studio Code	Environnement de développement sur Windows	0.00€
Cloudflare	Crée des tunnels sécurisés	0.00€
Tailscale	Crée des réseaux virtuels	0.00€
TigerVNC	Connexion à distance (VNC)	0.00€
Autodesk Fusion 360	Pour pouvoir modéliser la structure 3D de la borne	0.00€
Stripe	Crée des paiements sécurisés	0.00€

Le prix de tous les logiciels m'est revenu à 0,00€ pour pouvoir réaliser mon projet de travail de fin d'études.

2.2.2 Thonny IDE

Qu’est-ce que Thonny IDE ?

C'est un environnement de développement qui est gratuit et open-source, il a été conçu spécifiquement pour pouvoir débuter en programmation et pour pouvoir coder en Python ou en micro-python. Il est très facile à utiliser grâce à son interface qui est simple à comprendre.

Implantation dans mon projet :

Dans mon projet, ce logiciel m'a permis de pouvoir coder en micro-python pour configurer mon capteur PIR sur mon ESP32.

2.2.3 Visual Studio Code

Qu’est-ce que Visual Studio Code ?

C'est un environnement de développement qui est gratuit, léger et qui a été conçu par Microsoft. Il est disponible dans tous les systèmes d'exploitation (Windows 10-11, MacOS et Linux). Il a été conçu pour la programmation moderne.

Implantation dans mon projet :

Dans mon projet, ce logiciel, je l'utilise pour pouvoir coder ma partie Python en FLASK et pour pouvoir rajouter des templates HTML et CSS.

2.2.4 Cloudflare

Qu’est-ce que Cloudflare ?

Avant tout, c'est une entreprise américaine. Cloudflare est l'un des réseaux les plus vastes. L'option Cloudflare tunnels permet de créer des tunnels sécurisés entre un serveur et un nom de domaine pour que le serveur puisse envoyer la page web qu'il souhaite afficher sur le nom de domaine.

Pour faire cela, Cloudflare expose le serveur local de manière sécurisée sur internet. Ensuite, il crée un lien chiffré (qui est le tunnel) direct entre le serveur et le nom de domaine public. Cloudflare utilise un accélérateur web pour que tout soit bien synchronisé entre le serveur local et la page web qui est affichée sur le nom de domaine. Cloudflare utilise des bots pour surveiller les attaques DDoS.

Implantation dans mon projet :

Dans mon projet Cloudflare, permet de pouvoir faire afficher ma borne de commande qui est une application web sur mon nom de domaine " chakri.be ". Il a fallu que je configure un tunnel entre mon nom de domaine et mon premier Raspberry Pi qui sert de serveur avec Cloudflare. Le premier Raspberry Pi 4 envoie une requête à Cloudflare pour faire afficher la borne sur le nom de domaine en toute sécurité.

2.2.5 Tailscale

Qu’est-ce que Tailscale ?

Tailscale est un logiciel qui permet de créer un réseau privé virtuel (C'est un VPN de type Mesh) qu'on appelle "Tailnet" dans lequel on peut relier tous nos appareils entre eux de manière sécurisée, c'est comme s'ils étaient tous branchés à la même box wifi. Grâce au réseau virtuel Tailnet, on peut accéder à n'importe quel appareil n'importe où dans le monde.

Implantation dans mon projet :

Dans mon projet j'utilise Tailscale car dessus j'ai mis tous mes appareils que j'utilise : mes deux Raspberry Pi 4, ma tour ainsi que mon PC portable. Tailscale leur donne une adresse IP avec laquelle je peux y accéder à l'appareil via SSH ou VNC. Je l'utilise parce que quand j'ai commencé à travailler à distance je pouvais faire ça que localement, donc je devais prendre à chaque fois tout avec moi, alors qu'avec mon tailnet je sais laisser les deux Raspberry Pi 4 chez moi.

2.2.6 Stripe

Qu’est-ce que Stripe ?

Stripe est une entreprise qui propose à ses clients de pouvoir intégrer des environnements de paiement sécurisé dans leur application.

Implantation dans mon projet :

J’utilise Stripe pour pouvoir intégrer un système de paiement gratuit sur ma borne de commande qui propose de payer via carte bancaire et bien plus encore.

2.2.7 TigerVNC

Qu’est-ce que TigerVNC ?

TigerVNC est un logiciel de bureau à distance qui utilise le protocole VNC, il permet donc d'afficher graphiquement un ordinateur distant sur une machine cliente. TigerVNC est gratuit, performant et c'est un logiciel qui sait s'adapter à des connexions à faible débit.

Implantation dans mon projet :

Dans mon projet, j'utilise TigerVNC pour pouvoir me connecter à distance en VNC à mes deux Raspberry Pi 4 pour éviter de toujours prendre un clavier, une souris pour travailler dessus.

2.2.8 Autodesk Fusion 360

Qu’est-ce que Autodesk Fusion 360 ?

Autodesk Fusion 360 est un logiciel qui permet de pouvoir concevoir, simuler et fabriquer des objets 3D. Il propose toutes les fonctionnalités de base pour faire de la conception 3D.

Implantation dans mon projet :

Dans mon projet, j'ai utilisé Fusion 360 pour modéliser la structure de ma borne de commande. J'ai aussi fait des tests de simulation, ce que d'autres programmes de conception 3D ne proposent pas.

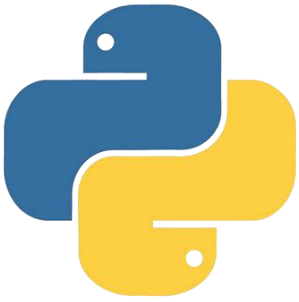
2.3 Les langages de programmation utilisés

Voici la liste des langages de programmation nécessaires dans mon projet :

Nom	Description
Python	Flask avec templates HTML et CSS
Micro-Python	Pour configurer le Capteur PIR

2.3.1 Python

Qu'est-ce que Python ?



Python est un langage de programmation gratuit et open-source, il a été créé par Guido van Rossum en 1991. A l'heure d'aujourd'hui, il est utilisé pour le développement web, l'analyse de données, l'intelligence artificielle ou encore l'automatisation.

Implantation dans mon projet :

Dans mon projet j'utilise Python avec principalement la bibliothèque FLASK, ce qui implique que j'utilise des templates HTML et CSS, tout ça pour pouvoir coder ma borne de commande sous forme d'une application web fiable et fluide.

2.3.2 Micro-Python

Qu’est-ce que Micro-Python ?

Micro-python est un langage plus léger basé sur Python 3 et qui a été conçu spécifiquement pour fonctionner sur des microcontrôleurs qui ont très peu de mémoire. Micro-Python tient seulement sur quelques kilo-octets et il permet de programmer du matériel électronique comme des ESP32 ou encore des Raspberry Pi Pico.

Implantation dans mon projet :

Dans mon projet, j'utilise du Micro-python pour pouvoir configurer mon ESP32 pour qu'il puisse contrôler mon capteur PIR. C'est ce code qui permet de lire les données envoyées par le capteur PIR en temps réel. Le code Micro-Python est alors enregistré dans la mémoire de l'ESP32.

2.4 Les bases de données utilisées

Voici la liste des bases de données nécessaires dans mon projet :

Nom	Description
Firebase	Base de données en ligne
SQLite	Base de données locale
DB Browser For SQLite	Logiciel qui permet d'exploiter un fichier SQL

2.4.1 Firebase

Qu'est-ce que Firebase ?

Firebase est une plateforme de Google qui fournit une suite d'outils aux développeurs tels que des bases de données en temps réel, des systèmes d'authentification ou encore de l'hébergement.

Implantation dans mon projet :

Dans mon projet, j'utilise Firebase pour leur authentification très sécurisée et également pour stocker chez eux les comptes clients qui ont été créés sur la borne de commande. Je ne stocke pas chez moi les comptes clients parce que je respecte la vie privée de mes clients, alors c'est pour ça que je préfère confier les comptes à des bases de données comme Firebase qui sont très sécurisées.

2.4.2 SQLite

Qu’est-ce que SQLite ?

SQLite est un système de gestion de base de données qui est très léger. Ce qui le rend avantageux, c'est qu'il n'a pas besoin d'un serveur à côté pour fonctionner. On dit qu'il est "Serverless", donc ce qui signifie "Sans serveur". Il enregistre toutes les données dans un fichier ".db" qui est stocké localement sur la machine.

Implantation dans mon projet :

Dans mon projet, je l'utilise pour stocker des informations ; Pour être plus précis, mon fichier SQLite s'appelle " database.db " et il contient quatre tables. Cette base de données me permet de stocker les points des clients dans la table "clients", stocker les commandes dans la table "commandes", stocker les produits du restaurant dans la table "produits" et pour stocker les statuts des tables libres ou occupées (qui sont envoyées par l'ESP32) dans la table "tables_resto".

2.4.3 DB Browser for SQLite

Qu’est-ce que DB Browser for SQLite ?

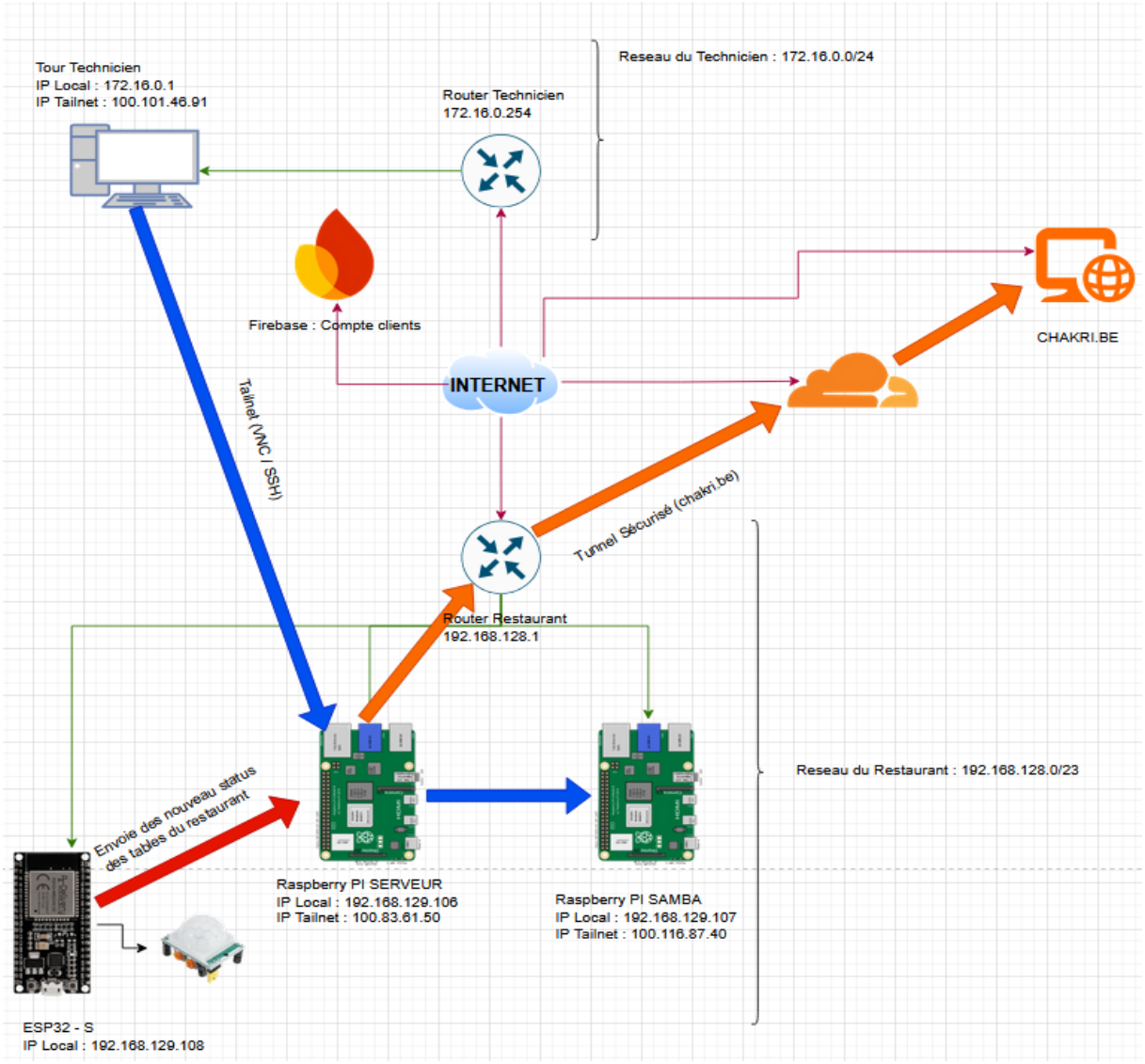
DB Browser for SQLite est un outil gratuit et open-source qui permet de visualiser, modifier ou créer des bases de données SQLite via un outil graphique.

Implantation dans mon projet :

Dans mon projet, je l'ai utilisé plusieurs fois pour vérifier si les tables ont bien été créées et également pour vérifier si les données de chaque table s'enregistrent correctement.

3. Les schémas

3.1 Topologie réseau



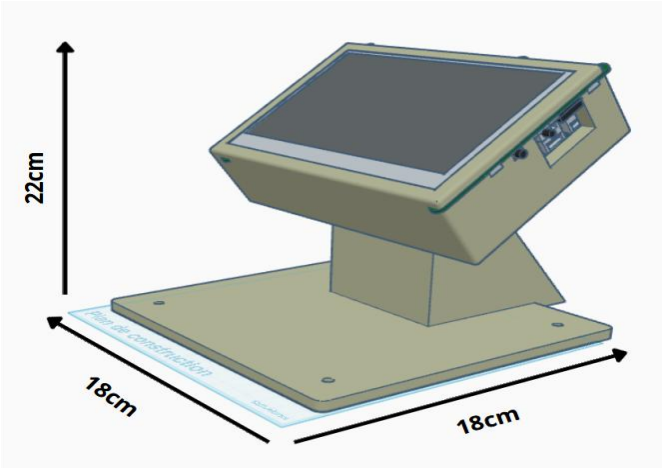
Explication de ma topologie réseau :

Comme vous pouvez le constater, le matériel de ma borne de commande est tout simplement branché en local sur le routeur du restaurant. Les clients peuvent y accéder depuis le nom de domaine "chakri.be" car j'ai exposé mon serveur sur internet (Premier Raspberry Pi 4) via un tunnel sécurisé "Cloudflare". Du côté de l'administration, j'ai mis en place un tunnel VPN pour que mes appareils soient sur un même réseau virtuel et que le technicien puisse y accéder à distance à partir de chez lui.

3.2 Schéma Mécanique

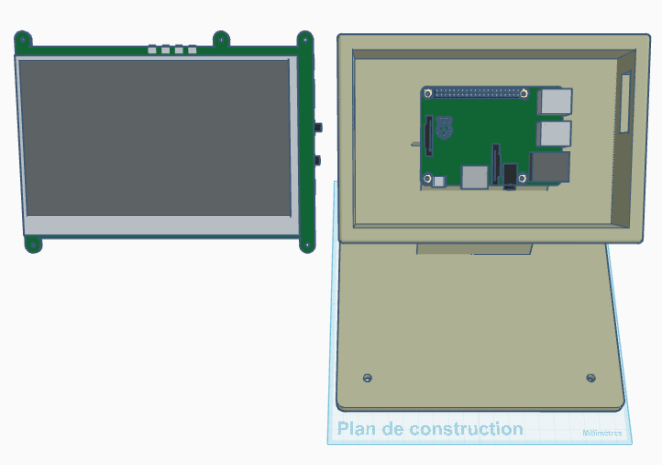
J’ai décidé de représenter le schéma mécanique sous plusieurs formes d’abord la partie borne de commande et ensuite la partie capteur PIR + ESP32.

3.2.1 Partie Borne de commande



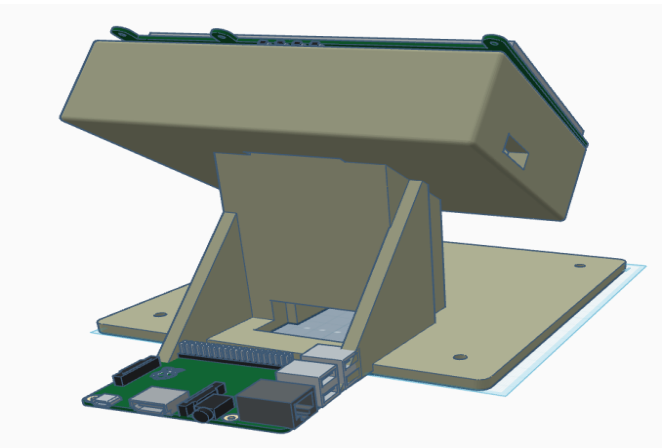
1) Vue en perspective :

Ici, comme vous pouvez le constater, la borne de commande est assez compacte et pratique à mettre en place pour sa petite taille. Elle fait exactement 18 cm de longueur et de largeur, ainsi que 22 cm de hauteur.



2) Vue de l’intérieure :

A l'intérieur de la borne, on a notre premier Raspberry Pi 4 qui sert d'hébergeur de la borne de commande, l'écran tactile de 7 pouces est donc relié au Raspberry Pi 4 en HDMI, l'écran permet au client d'interagir avec la borne.



3) Vue de l’arrière :

A l'arrière, comme vous venez de le remarquer, il y a le deuxième Raspberry Pi 4 qui sert de serveur samba. Il permet de partager un dossier samba "VDB_Backup" dans lequel le premier Raspberry Pi 4 envoie les données de la borne.

3.2.2 Partie ESP32 + Capteur PIR

1) Vue du haut :

Il s'agit du boîtier dans lequel l'ESP32 et le capteur de mouvement PIR seront, ce boîtier fait exactement 12 cm de longueur et 6.5 cm de largeur ainsi que 3.5 cm de profondeur. Le boîtier possède des encoches pour pouvoir s'attacher facilement sur le mur, il est assez petit pour que le client ne le voie pas et qu'il ne soit pas dérangé constamment. Ce boîtier a un petit trou, comme vous pouvez le voir, pour que le dôme du capteur PIR puisse s'ajuster facilement et l'ESP32 se place au milieu de la plaque arrière du boîtier.

4. Le travail réaliser

4.1 Explication du fonctionnement de la borne - Frontend

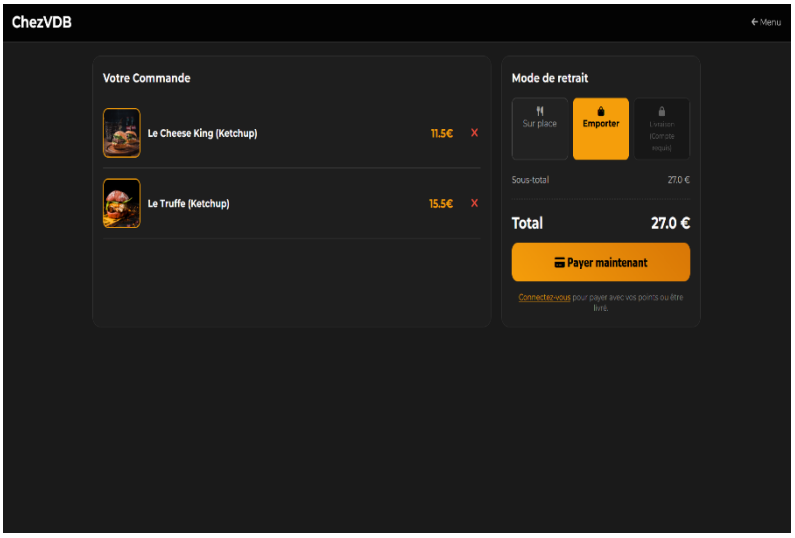
4.1.1 Le côté invité

1) L'écran d'accueil :

La borne invite forcément le client à interagir avec en appuyant sur " commander maintenant" ou bien en se connectant à un compte.

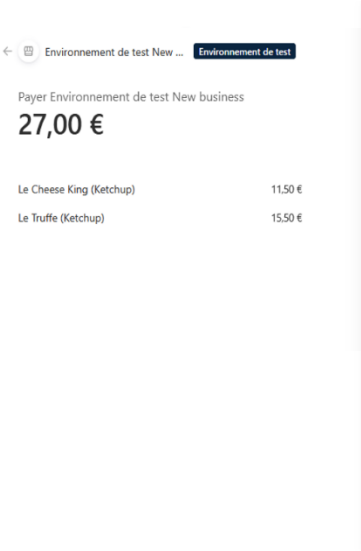
2) L'écran de menu :

Le client a le choix de choisir ce qu'il veut par rapport au vaste menu proposé par le restaurant.



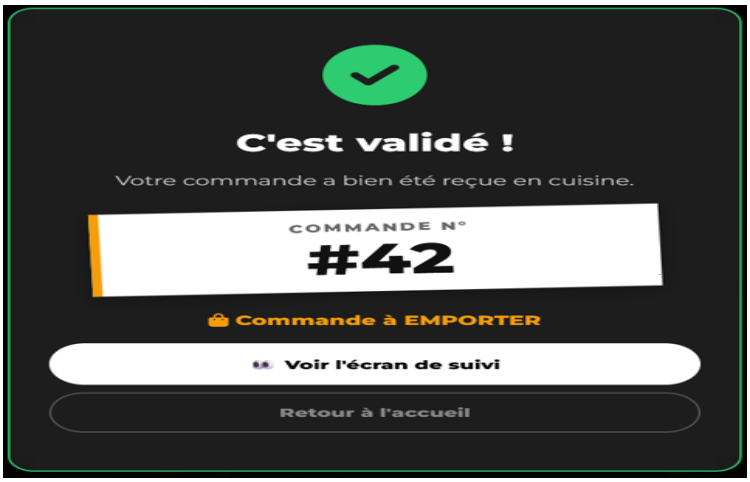
3) L'écran du panier :

Le client arrive alors sur son panier, dans lequel il peut choisir entre " Sur place", "Emporter" ou bien " Livrer ", mais cette option nécessite un compte pour pouvoir l'utiliser. Il a également le choix de supprimer un des menus ou plusieurs, une fois que le client a choisi ce qu'il voulait, il peut passer au paiement. L'option choisie ici est " Emporter".



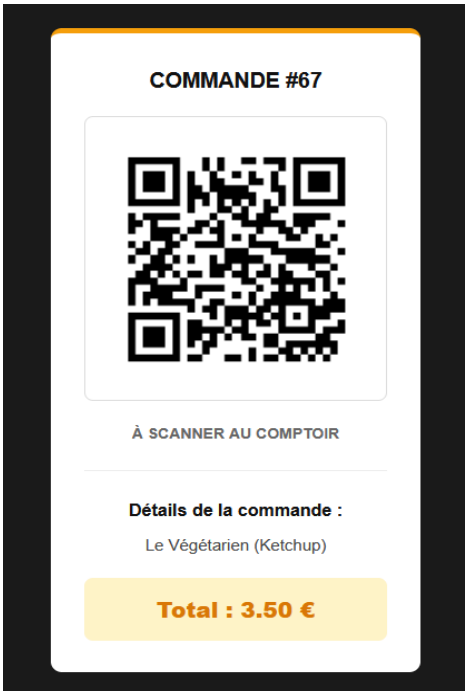
4) L'écran de paiement :

La page de paiement s'affiche alors, dans laquelle le client se doit de compléter des informations telles que son vrai mail (pour qu'il puisse recevoir son ticket par mail). Ensuite, il doit compléter les informations de sa carte et payer.



5) L'écran de confirmation :

Pour finir, le client est redirigé vers cette page-ci qui lui indique son numéro de ticket, et il reçoit un ticket avec un QR code à son adresse mail.



6) les informations du ticket :

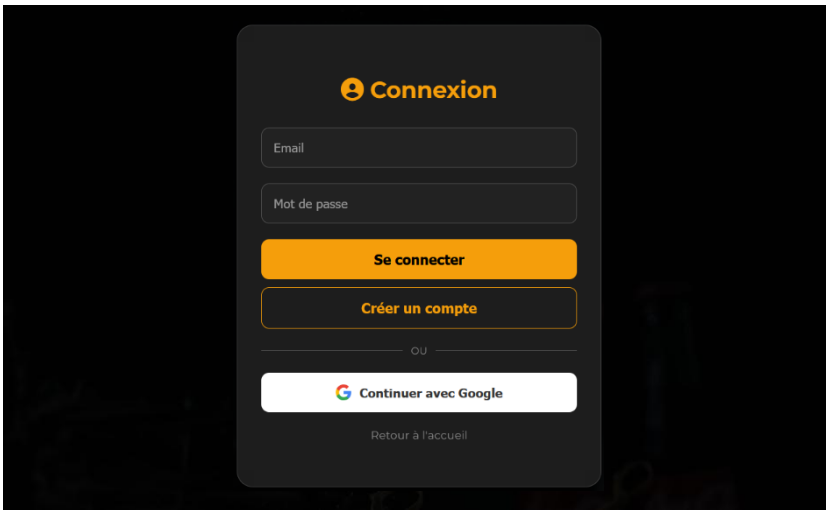
Le client reçoit alors le ticket avec un QR code qu'il doit faire scanner au comptoir pour que la commande puisse être validée. Ici, bien sûr, ce n'est pas le ticket qui correspond à la commande 42 car à ce moment-là je n'avais pas encore mis ce système en place.

Dans le cas où le client aurait choisi de manger sur place, il y aurait eu exactement les mêmes étapes, mais à l'écran de confirmation, il y aurait eu un message en plus pour l'avertir qu'il a directement une place de libre dans le restaurant.

4.1.2 Le côté compte client

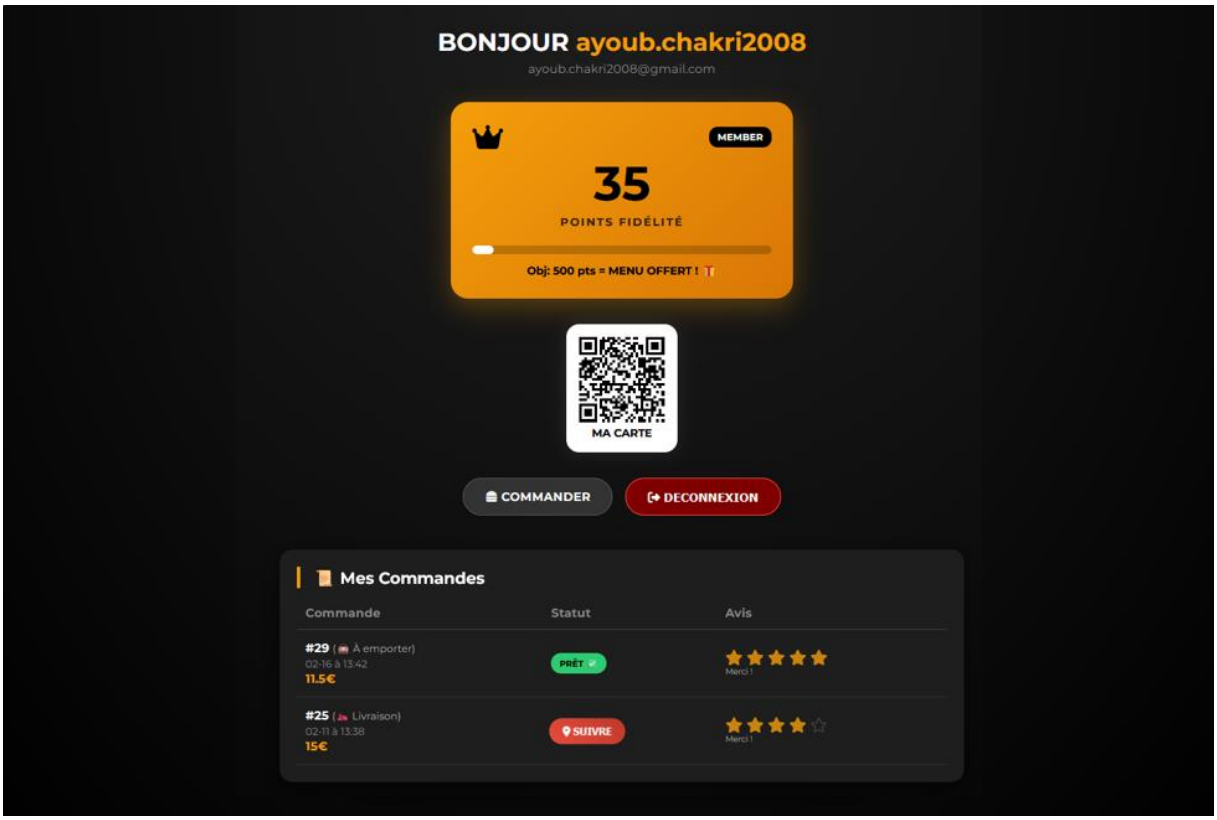
1) L'écran d'accueil :

Pour qu'un client puisse se connecter, il doit appuyer sur le bouton "connexion" de l'écran d'accueil.



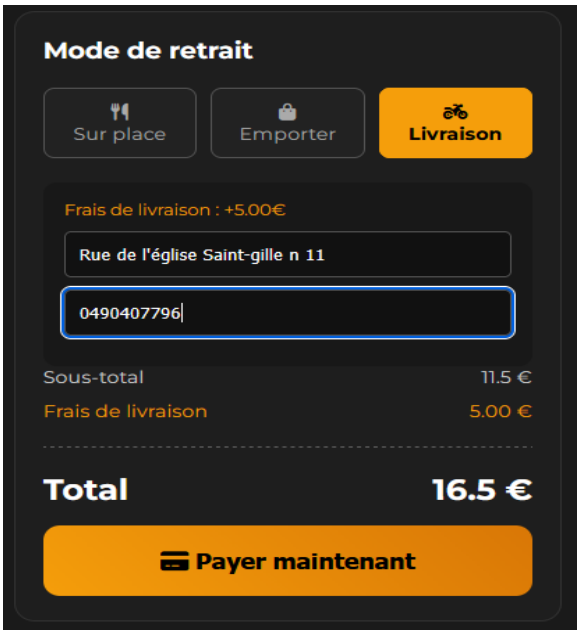
2) L'écran de connexion :

Le client a alors le choix de se connecter en tapant ses identifiants à la main sur l'écran ou bien en se connectant directement avec Google.

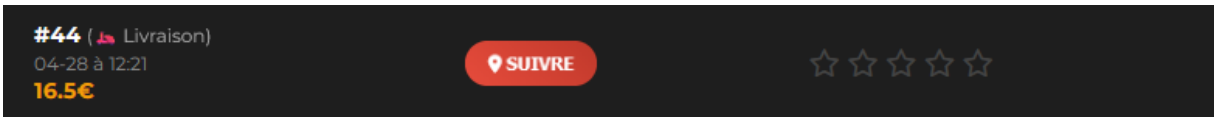


Une fois que le client a créé son compte ou bien qu'il s'est tout simplement connecté, alors il a accès à sa carte membre, il peut encaisser des points de fidélité en commandant des menus. Il peut aussi voir son historique de commande ainsi que le statut de la commande (si la commande est encore à la cuisine, prête ou bien encore suivre le statut de livraison, c'est-à-dire voir où on en est le livreur sur son chemin). Le client peut également donner un avis par rapport à chaque commande en cliquant sur un nombre d'étoiles qui varie entre 1 et 5.

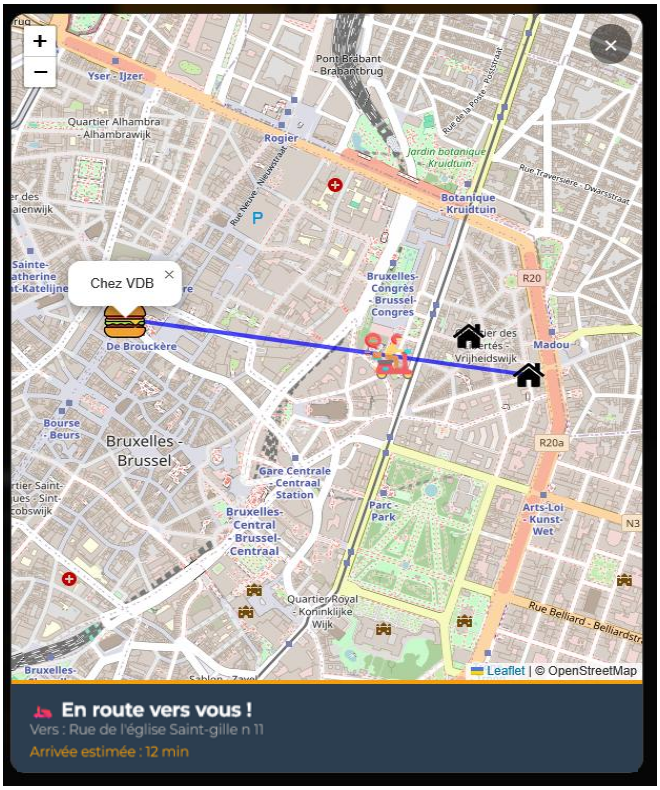
Maintenant qu'on est connecté à un compte client, je peux facilement vous montrer l'option "Livrer". Donc ce sont exactement les mêmes étapes : le client clique sur commander, il choisit son menu et dans le panier, il sélectionne l'option "Livrer".



Le client choisit l'option "Livraison" et ensuite il est obligé de renseigner 2 informations, dont son adresse ainsi que son numéro de téléphone, pour que le livreur le tienne informé si jamais il y a un problème lors de la livraison.

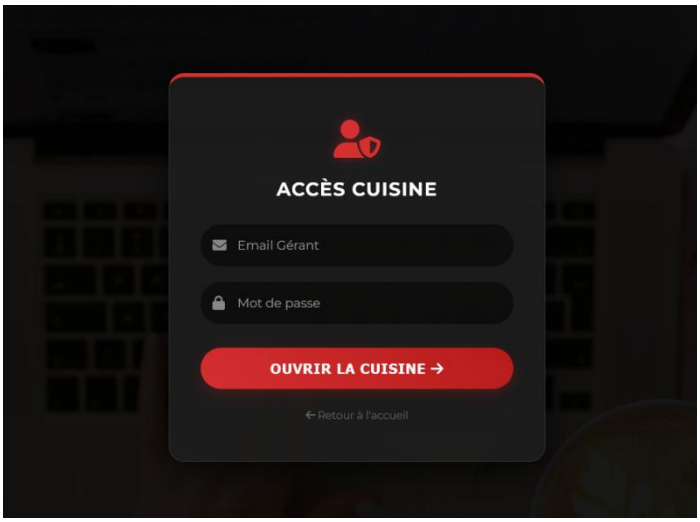


Comme vous l'avez constaté, la commande est apparue dans l'historique de commande du client et il peut désormais suivre la trajectoire du livreur en cliquant sur "Suivre".



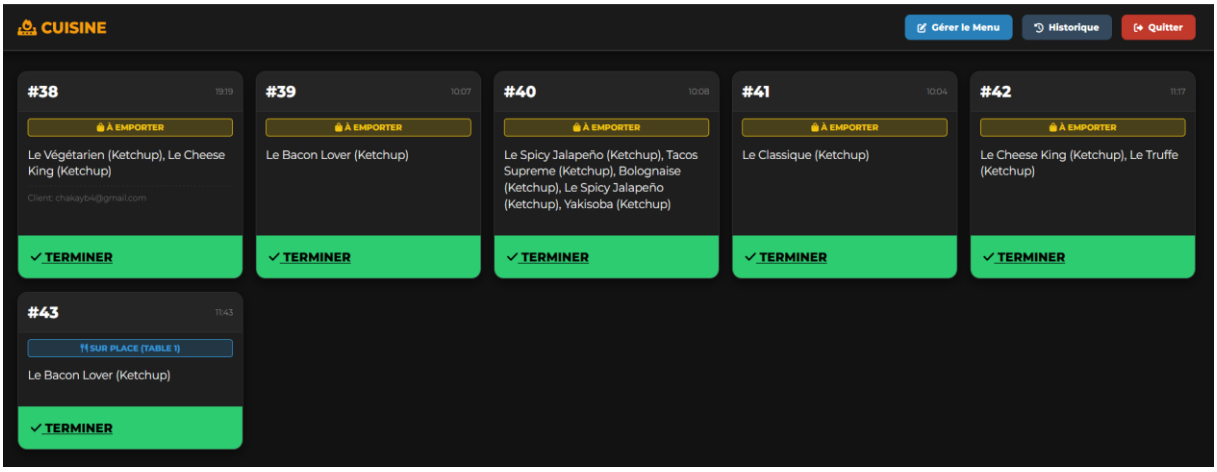
Un pop-up s'affiche alors avec une vraie carte de la capitale de Bruxelles et avec une trajectoire précise allant du restaurant jusqu'à l'adresse renseignée par le client. Bien sûr, cette option reste une simulation car je n'utilise pas de vrai tracker pour que ça soit réel et après tout, ce n'est pas comme si j'avais des UBER à mon service.

4.1.3 Le côté cuisine



1) L'écran de connexion - cuisine :

L'accès à la cuisine est sécurisé via directement une authentification dont seul le chef de cuisine connaît, j'ai mis en place cela pour des mesures de sécurité.



2) L'écran Dashboard :

Toutes les commandes en attente figurent dans cet écran précisément, avec le numéro et le détail pour chaque commande. Pour que la commande soit prête, il faut que le chef appuie sur "terminer". Dans ce Dashboard, on a le bouton qui donne accès aux faits de rajouter, modifier ou supprimer des produits et on a également le bouton "Historique" qui permet de visualiser toutes les anciennes commandes.

La partie frontend, j'ai utilisé de l'IA pour pouvoir générer des templates personnalisés pour ma borne de commande.

4.2 Explication du fonctionnement de la borne - Backend

La partie Frontend permet au client d'interagir avec la borne de commande, mais derrière cela on a ce qu'on appelle le Backend qui va lui servir à faire fonctionner la borne de commande, pour cela j'ai utilisé le langage Python avec FLASK. Voici quelques notions importantes dans mon code qui permettent de faire fonctionner la borne.

4.2.1 L'intégration de ma base de données SQLite

```
def get_db_connection():
    conn = sqlite3.connect('database.db')
    conn.row_factory = sqlite3.Row
    return conn
```

J'utilise cette fonction qui permet au serveur Flask d'obtenir une connexion avec la DB lorsqu'il a besoin de lire ou d'écrire des données dans la base de données locale « database.db ».

Exemple :

```
@app.route('/menu')
def page_menu():
    conn = get_db_connection()
    produits = conn.execute('SELECT * FROM produits').fetchall()
    conn.close()
    return render_template('menu.html', produits=produits)
```

J'ai pris cette route comme exemple car, comme on peut le voir, la fonction "page_menu" fait appel à "get_db_connection()" qu'elle va stocker dans la variable "conn" (cette variable va juste stocker la connexion). Si on ne faisait pas appel à cette fonction, ça serait impossible d'afficher l'ensemble des données de la table produits dans la page " menu.html ".

4.2.2 L’intégration du paiement en ligne avec Stripe

```python
@app.route('/create-checkout-session', methods=['POST'])
def create_checkout_session():
    panier = session.get('panier', [])
    if not panier: return redirect(url_for('page_menu'))
    mode = session.get('mode_commande', 'emporter')
    items = [{'price_data': {'currency': 'eur', 'product_data': {'name': p['nom']}, 'unit_amount': int(p['prix'] * 100)}, 'quantity': 1} for p in panier]
    if mode == 'livraison': items.append({'price_data': {'currency': 'eur', 'product_data': {'name': 'Livraison '}, 'unit_amount': 500}, 'quantity': 1})
    try:
        s = stripe.checkout.Session.create(line_items=items, mode='payment', success_url=YOUR_DOMAIN + '/success', cancel_url=YOUR_DOMAIN + '/panier', met
        session['stripe_session_id'] = s.id
    except Exception as e: return str(e)
    return redirect(s.url, code=303)
```

Cette route " create-checkout-session" permet tout simplement de gérer le paiement via Stripe uniquement au moment où le client valide son panier, bien sûr le paiement ne se fait pas par magie, il y a des étapes qui se font derrière et qui sont les suivantes.

- Tout d'abord, cette route va récupérer les informations de son panier telles que les produits et l'option que le client a choisie.

- Vu que Stripe a besoin des prix en centimes pour fonctionner, elle va formater le prix de chaque produit pour faire passer le prix final d'euro aux centimes. Pour faire cela, la route utilise "ont(p['prix'] * 100) ".

- Si l'option sélectionnée est la "Livraison", alors cette route va tout simplement rajouter 5 euros au prix initial, donc comme c'est en centimes, elle rajoute 500 centimes.

- Une fois que les trois étapes sont faites, c'est à ce moment-là que la route va rediriger vers le paiement sécurisé, où le client va retrouver les produits qu'il a commandés.

- La dernière étape sert à vérifier si le client a vraiment payé ou pas, si maintenant tout se passe bien alors il sera redirigé vers la page de confirmation " succes.html " et si le client annule alors il est redirigé sur la page du panier " panier.html ".

4.2.3 L’intégration de l’envoi de mail

```python
def envoyer_ticket_mail(destinataire, numero_ticket, montant, details):
    EMAIL_VDB = "ayoub.chakri2008@gmail.com"
    PASS_VDB = "qvvmhwzydpnpfgfn"

    msg = EmailMessage()
    msg['Subject'] = f'🎟️ Votre QR Code - Commande #{numero_ticket}'
    msg['From'] = EMAIL_VDB
    msg['To'] = destinataire

    msg.set_content(f"Voici le QR code de votre commande #{numero_ticket}. À présenter au comptoir.")

    lien_vendeur = f"{YOUR_DOMAIN}/ticket/{numero_ticket}"
    lien_encode = urllib.parse.quote(lien_vendeur)
    qr_url = f"https://quickchart.io/qr?text={lien_encode}"
```

```python
html_content = f"""
<!DOCTYPE html>
<html>
<body style="margin:0; padding:40px 20px; background-color:#1a1a1a; text-align:center; font-family:'Arial', sans-serif;">
    <div style="background-color:#ffffff; padding:30px; border-radius:10px; display:inline-block; box-shadow: 0 10px 25px rgba(0,0,0,0.5); border-top: 5px solid #f59e0b; max-width: 400px;">
        <h2 style="color:#111; margin-top:0; text-transform:uppercase;">Commande #{numero_ticket}</h2>

        <img src="{qr_url}" alt="QR Code" width="250" height="250" style="display:block; margin: 20px auto; border: 1px solid #ddd; padding: 10px; border-radius: 8px;">
        <p style="color:#666; font-size:14px; font-weight:bold; text-transform:uppercase; margin-bottom:0;">À scanner au comptoir</p>

        <hr style="border: none; border-top: 1px solid #eee; margin: 25px 0;">

        <h3 style="color:#111; margin: 0 0 10px 0; font-size:16px;">Détails de la commande :</h3>
        <p style="color:#444; font-size:15px; margin: 0 0 20px 0; line-height:1.5;">{details}</p>

        <div style="background-color:#fef3c7; padding: 15px; border-radius: 8px;">
            <p style="color:#d97706; font-size:20px; font-weight:900; margin: 0;">Total : {montant:.2f} €</p>
        </div>
    </div>
</body>
</html>
"""
```

Pour pouvoir envoyer des mails contenant le ticket digital (avec QR code) à mes clients, j'ai dû intégrer la bibliothèque SMTPlib et j'utilise une fonction qui envoie les mails en suivant plusieurs étapes simples.

Tout d'abord, elle va préparer l'email et le mot de passe qu'elle va utiliser pour envoyer le mail (bien sûr ce sont toujours les mêmes identifiants et ils ne changent pas). Après avoir fait ça, la fonction va récupérer le mail du destinataire (donc du client) et de l'expéditeur (celle que j'ai définie au début de la fonction). Ensuite, elle va rédiger le contenu du mail en se basant sur les informations de la commande du client. Vu que j'ai décidé que la fonction envoie le mail via GMAIL, la fonction va alors utiliser SMTP_SSL sur le port 465 (c'est la connexion sécurisée via Gmail) pour se connecter avec les identifiants que j'ai définis et donc envoyer le mail.

Si jamais ça n'envoie pas le mail et qu'il y a une erreur (ce qui serait très bizarre), alors ça affichera uniquement "erreur envoi mail" dans le terminal du serveur flask.

4.2.4 Communication avec l’ESP32

```python
@app.route('/api/update_tables', methods=['POST'])
def update_tables_iot():
    data = request.json
    if data:
        conn = get_db_connection()
        try:
            for table_id, status in data.items():
                conn.execute("UPDATE tables_resto SET status = ? WHERE id = ?", (status, table_id))
            conn.commit()
            return jsonify({"status": "success"})
        except Exception as e:
            return jsonify({"status": "error", "message": str(e)}), 500
        finally:
            conn.close()
    return jsonify({"status": "error"}), 400
```

Cette route est primordiale car c'est une adresse à laquelle l'ESP32 envoie des requêtes JSON. C'est requête concerne tout simplement les données des statuts des tables du restaurant si elles sont "libres" ou "occupées". L'ESP32 envoie ses données à cette route qui sert d'adresse, car cette route va enregistrer les différents statuts dans la table " tables_restaurants " de la base de données " database.db ". Si l'ESP32 n'enverra pas ses données, alors lorsque le client sera à l'étape du panier, il cliquera sur l'option " Sur place", mais aucune table réelle ne lui sera attribuée.

Dans le screen, comme vous pouvez le voir, les données des statuts des tables reçues par l'ESP32 sont enregistrées dans la base de données et ensuite la route enverra une requête aussi pour dire que tout va bien "success". Si l'esp32 n'envoie pas de requête avec les données et que le client a sélectionné l'option "Sur place", alors la route retourne juste un JSON avec un code 400 pour dire qu'elle n'a rien reçu.

4.2.5 Système de fidélité avec points

```
pts_gagnes = 0
if user_email_db != 'Invité':
    pts_gagnes = int(montant * 10)
    if conn.execute("UPDATE clients SET points = points + ? WHERE email = ?", (pts_gagnes, user_email_db)).rowcount == 0:
        conn.execute("INSERT INTO clients (email, points) VALUES (?, ?)", (user_email_db, pts_gagnes))
```

Ce code concerne une petite partie importante de la route "success", c'est ce qui permet que chaque compte client bénéficie d'un système de fidélité avec points. Le système de fidélité s'applique donc qu'au client ayant un compte, pour chaque euro dépensé le client gagne 10 points sur sa carte de fidélité. Prenons un exemple assez simple et qui est le suivant :

Exemple : Ma commande m'a coûté exactement 27€, alors j'ai gagné 10% du montant total mais en point, c'est-à-dire 270 points.

Une fois que le client encaisse ses points, si le compte client existait déjà dans la table "clients" de la base de données " database.db ", alors ce sera juste une mise à jour de ses points " UPDATE". Et si le compte client est nouveau, alors le compte va juste être inséré dans la table de données avec ses points.

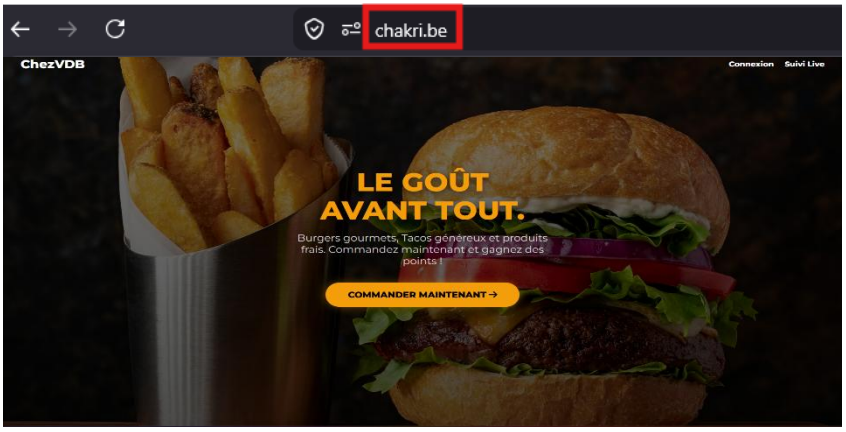
```
@app.route('/payer-points', methods=['POST'])
def payer_points():
```

Le client ayant un compte avec des points peut alors bénéficier de la route /payer-point qu'il y a dans mon code Python. Cette route permet au client de vérifier le nombre de points de son compte et de valider une commande s'il a assez de points.

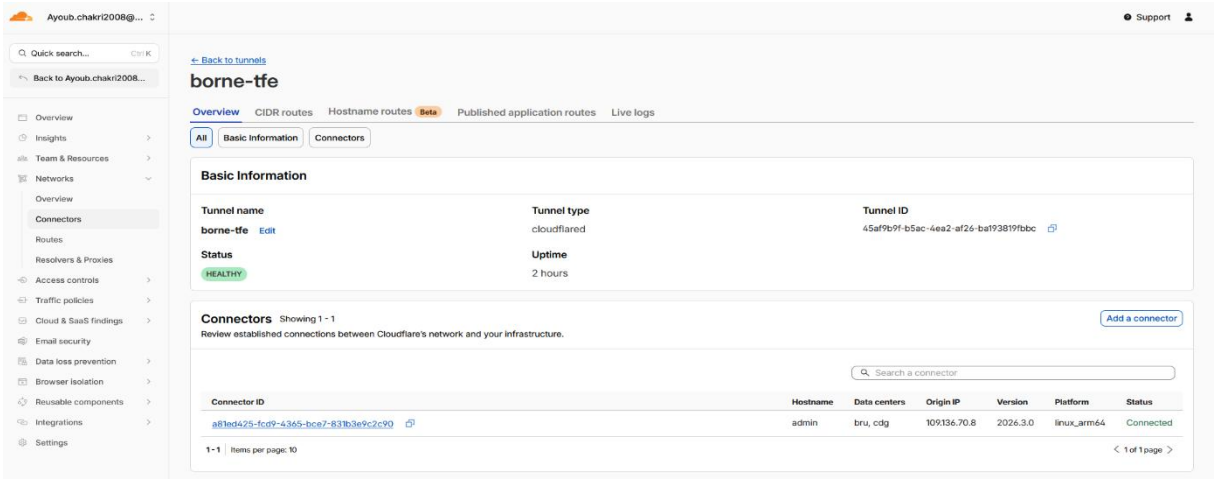
4.3 Les fonctionnalités de la borne

4.3.1 L’accès externe via le nom de domaine « chakri.be »

Ma borne de commande a un avantage par rapport à la concurrence, les autres bornes obligent le client à utiliser l'écran physique du restaurant. Pour rendre ma borne plus intéressante, je l'ai rendue accessible depuis n'importe quel smartphone via mon nom de domaine "chakri.be".



Si la borne du restaurant est occupée par un autre client, alors l'utilisateur peut très simplement scanner le QR code ou mettre chakri.be sur son GSM pour pouvoir commander.



La borne est hébergée sur mon premier Raspberry Pi 4 (Raspberry Pi serveur), pour que je puisse donner l'accès au client via " chakri.be ", le fait d'ouvrir des ports sur mon routeur était trop dangereux donc j'ai préféré mettre en place un tunnel sécurisé avec Cloudflare. Comme vous pouvez le voir, le statut "HEALTY" signifie que mon Raspberry Pi 4 communique bien avec les serveurs de Cloudflare et que le tunnel est actif.

4.3.2 Détection de place libre dans le restaurant

La deuxième fonctionnalité de ma borne de commande consiste à attribuer des tables libres à des clients lorsqu'ils sélectionnent l'option "Sur place" lors de leurs commandes. Pour mettre en place cela, j'ai utilisé un ESP32 et un Capteur de mouvement PIR SR-501. L'ESP32 sert de relais, il envoie donc les données captées par le capteur PIR pour que la borne sache quelle table du restaurant est libre ou pas.

Comment ça fonctionne ?

- Tout d'abord, le capteur PIR détecte les mouvements par la chaleur dégagée du client, s'il n'y a pas de chaleur corporelle, alors la table est libre, sinon la table est occupée.

- L'ESP32, qui lui est connecté au réseau local du restaurant, va intercepter les infos (table libre ou occupée) du capteur PIR et il va formater les données en requête JSON.

- Une fois qu'il a formaté ses données, la requête est alors envoyée à l'API rest que j'ai prédéfinie (" https://chakri.be/api/update_tables "), c'est celle de mon serveur, le serveur met alors à jour la table de données (tables_restaurants) avec la nouvelle requête JSON qu'il a reçue (si c'est un statut libre ou occupé pour la table du restaurant) et c'est comme ça que la borne de commande c'est quelles sont les tables libres et les attributs au client ayant choisi l'option "Sur place" lorsqu'il commande.

Bien sûr, pour qu'il n'y ait pas à chaque fois tant de requêtes envoyées en un temps très petit, j'ai décidé que l'ESP32 attend un certain temps avant d'envoyer une nouvelle requête.

4.3.3 Le serveur SAMBA

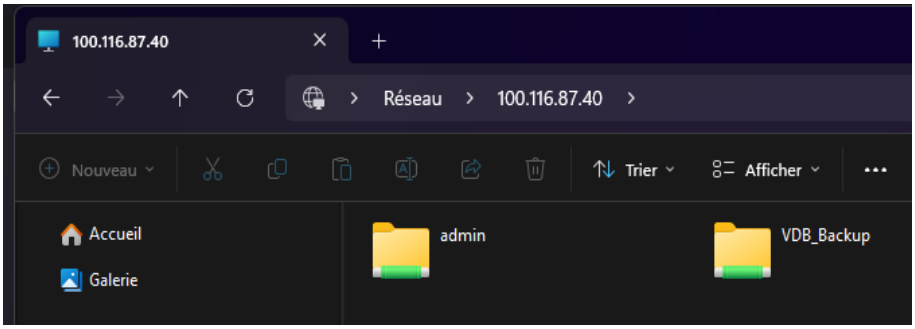
Qu'est-ce qu'un serveur samba ?

Un serveur Samba est une suite logicielle libre permettant le partage de fichiers et d'imprimantes entre des systèmes Linux et Windows sur un même réseau local.

Dans l'infrastructure de ma borne de commande, j'utilise un serveur Samba qui est pour moi une fonctionnalité que n'importe quelle infrastructure devrait avoir, car dans mon utilisation du serveur Samba, il me permet d'avoir une redondance de données. Imaginer si seulement la carte SD du premier Raspberry Pi qui héberge ma borne de commande sature, alors mes données seront perdues. Le serveur Samba, je l'ai configuré sur mon deuxième Raspberry Pi et je partage un dossier Samba sur le réseau virtuel de mon tailnet pour que personne d'autre n'ait accès à mes appareils de mon tailnet donc de mon tunnel VPN. Le premier Raspberry Pi 4 pourra alors envoyer les données de la borne de commande dans ce dossier Samba qui est sur le deuxième Raspberry Pi 4.

```
[VDB_Backup]
   path = /home/admin/vdb_backup
   writeable = yes
   browseable = yes
   public = no
   create mask = 0777
   directory mask = 0777
   force user = admin
```

Voici la configuration du partage du dossier dans lequel les données de la borne sont envoyées chaque jour à 00H00. Le seul user qui peut y accéder est le technicien avec son user "admin", il a tous les droits sur les données.



Le partage du dossier Samba est accessible qu'aux appareils inscrits dans mon tunnel VPN ; aucun autre appareil n'appartenant pas à mon tunnel n'a accès à ce dossier Samba.

5. Conclusion de mon projet

En résumé, ce travail de fin d'études m'a permis d'apprendre, de faire et de déployer une borne de commande avec une infrastructure sécurisée et entièrement fonctionnelle. Avec le développement web (Python avec Flask et des templates HTML/CSS) et du matériel à portée de main comme le Raspberry Pi 4, j'ai pu créer une infrastructure complète. La totalité de mes objectifs ont été atteints pour ce projet de fin d'études.

Franchement, ce qui m'a vraiment plu dans ce projet, c'est d'apprendre à chaque fois que je faisais quoi que ce soit, comme la mise en place de l'ESP32 avec son capteur de mouvement PIR ou bien le fait d'avoir mis en place le tunnel sécurisé, le tunnel VPN. Ce projet m'a vraiment fait sortir de ma zone de confort et m'a permis de toucher à plein de domaines différents que ce soit la gestion de base de données, le réseau, la programmation et même l'impression 3D.

Même si la borne de commande est opérationnelle à l'heure d'aujourd'hui, il y aura toujours moyen de l'améliorer, comme par exemple, je voudrais bien plus tard rajouter un vrai terminal de paiement physique en Bancontact directement sur la borne, comme ça le client a plusieurs choix de paiement. Par rapport à l'infrastructure, je voudrais plus tard acheter un serveur NAS pour déployer plusieurs bornes, pour bien sûr les vendre à plusieurs petites restaurations à un prix qui brisera la concurrence, mais encore une fois c'est pour plus tard.

6. Sitographie

6.1 Documents

ADAFRUIT, « PIR Motion Sensor », sur https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor, /, consulté le 12 février 2026.

CLOUDFLARE, « Cloudflare Tunnel documentation », sur https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/, /, consulté le 25 mars 2026.

FLASK, « Pallets Projects - Flask Documentation », sur https://flask.palletsprojects.com/, /, consulté le 10 novembre 2025.

MICROPYTHON, « Quick reference for the ESP32 », sur https://docs.micropython.org/en/latest/esp32/quickref.html, /, consulté le 4 février 2026.

PYTHON, « Python Language Reference », sur https://docs.python.org/3/, /, consulté le 5 novembre 2025.

REAL PYTHON, « Python SQLite : Working With Databases », sur https://realpython.com/python-sqlite-sqlalchemy/, /, consulté le 8 décembre 2025.

STRIPE, « Stripe API Reference for Payments », sur https://stripe.com/docs/api, /, consulté le 28 janvier 2026.

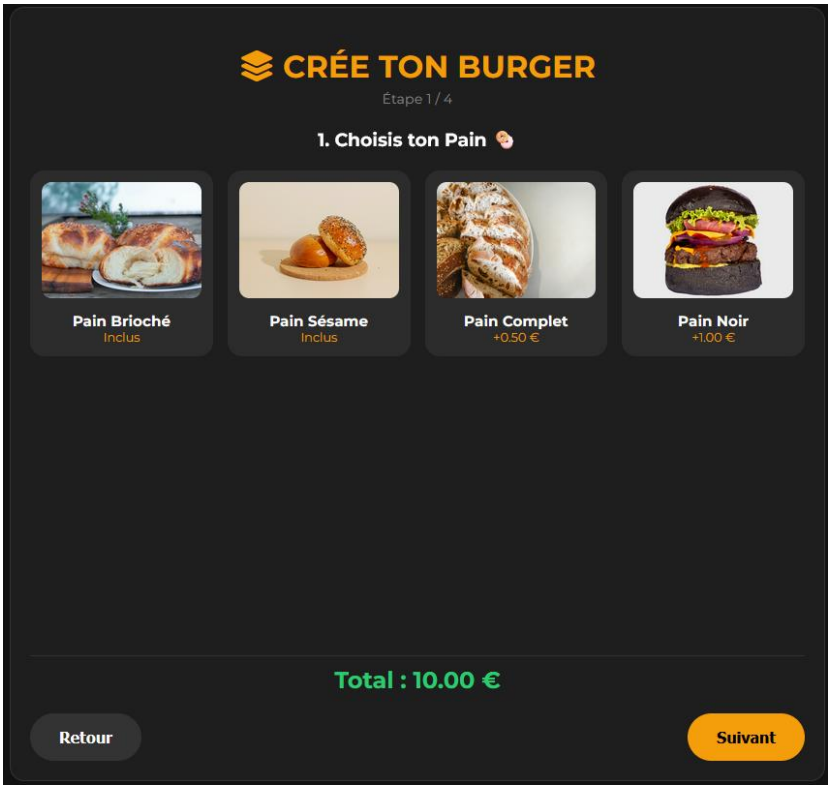
TAILSCALE, « Tailscale Documentation », sur https://tailscale.com/kb/, /, consulté le 30 mars 2026.

W3SCHOOLS, « HTML & CSS Tutorial », sur https://www.w3schools.com/html/, /, consulté le 22 octobre 2025.

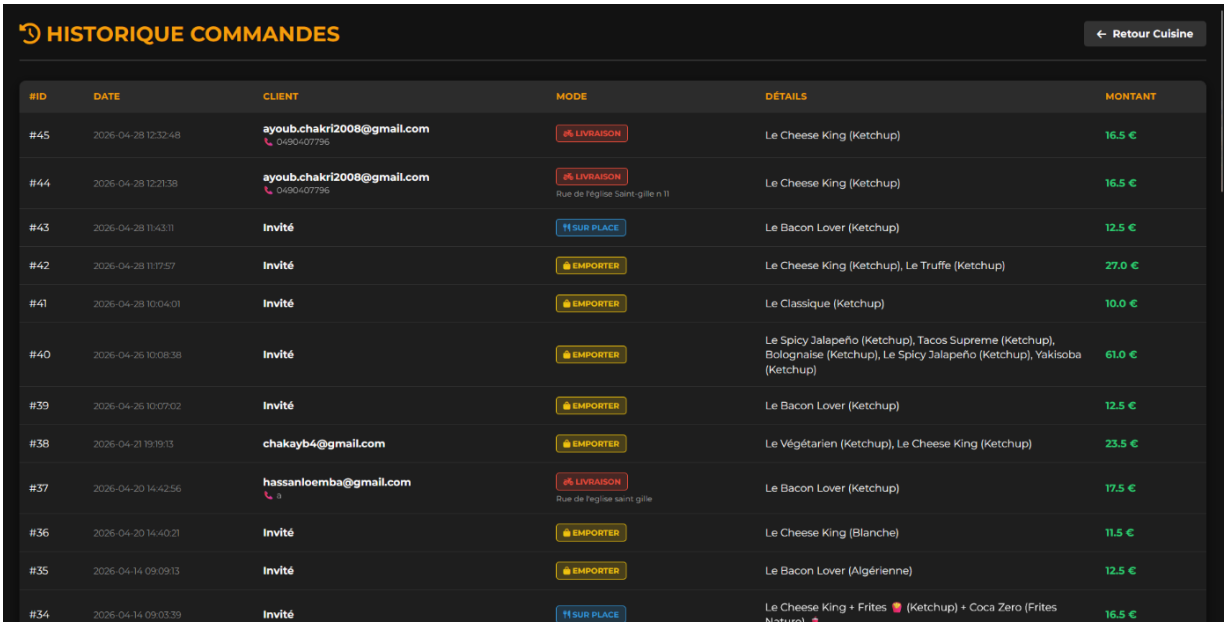
BELGINUX, « Créer un partage samba », sur https://belginux.com/creer-un-partage-samba/, /, consulté le 20 avril 2026.

7. Annexes

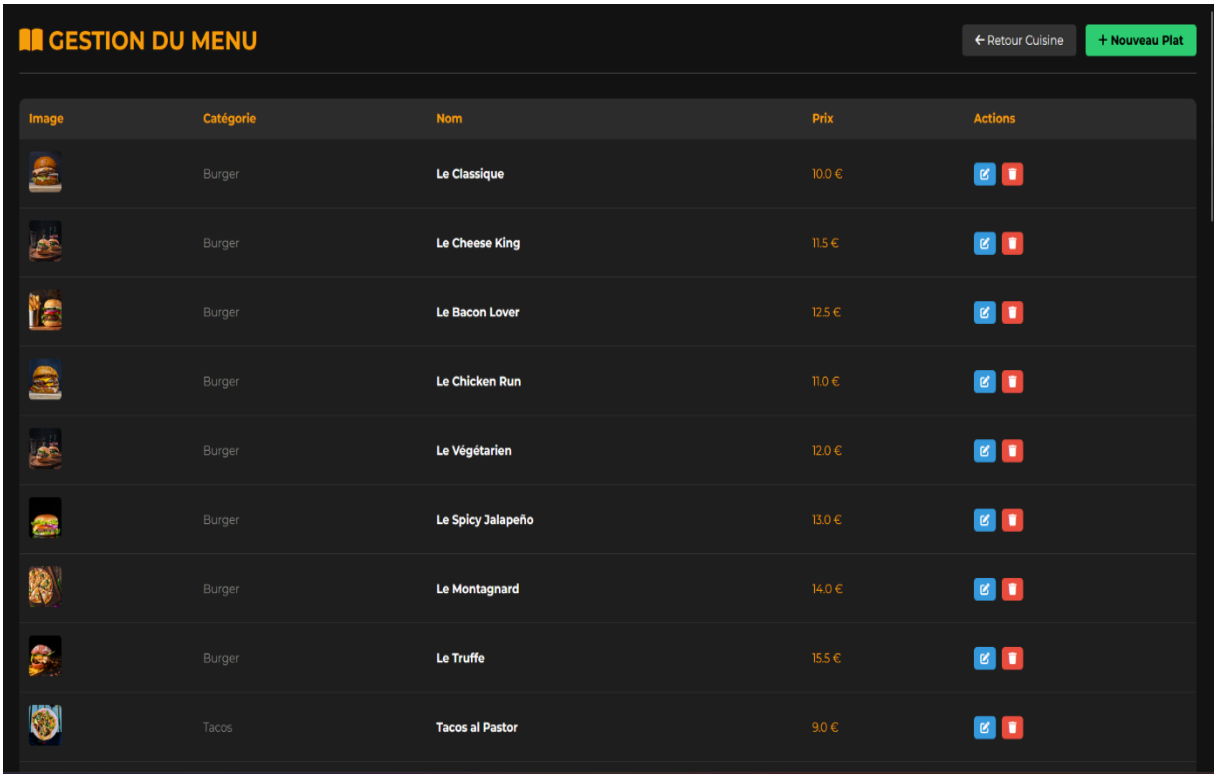
Coté invité/compte client - Crée son propre menu :

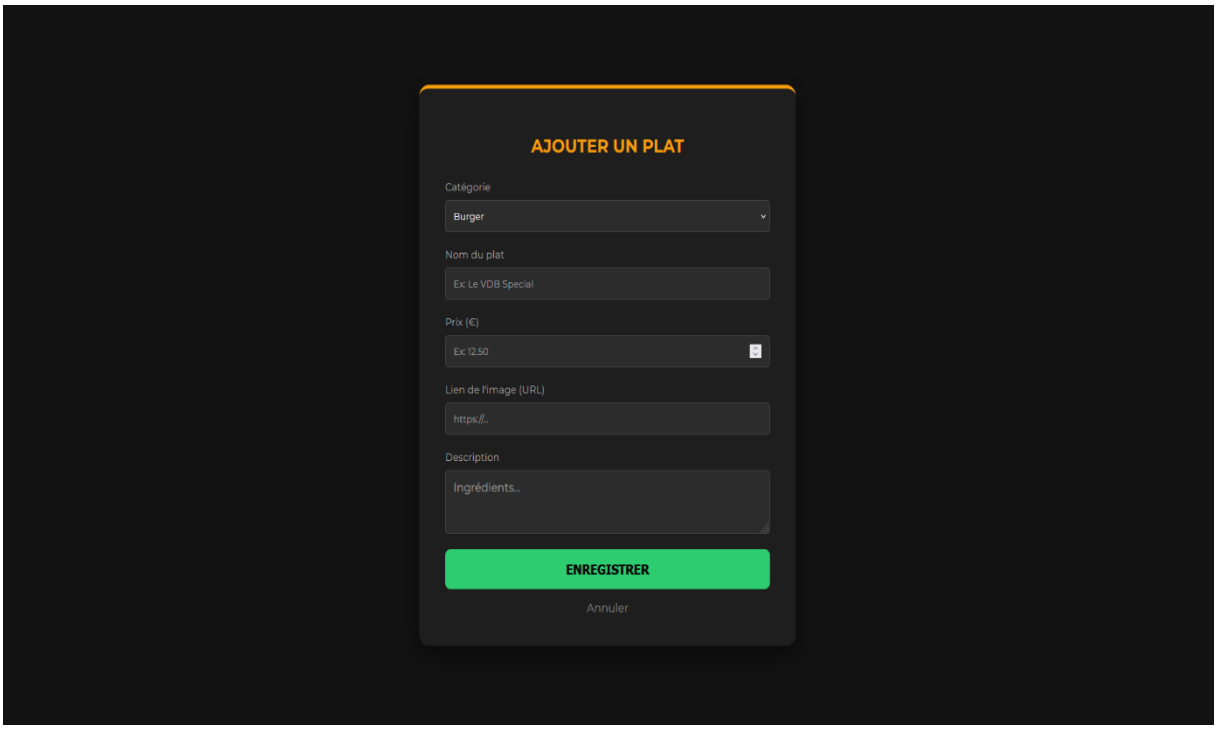


Côté cuisine – Historique des commandes :



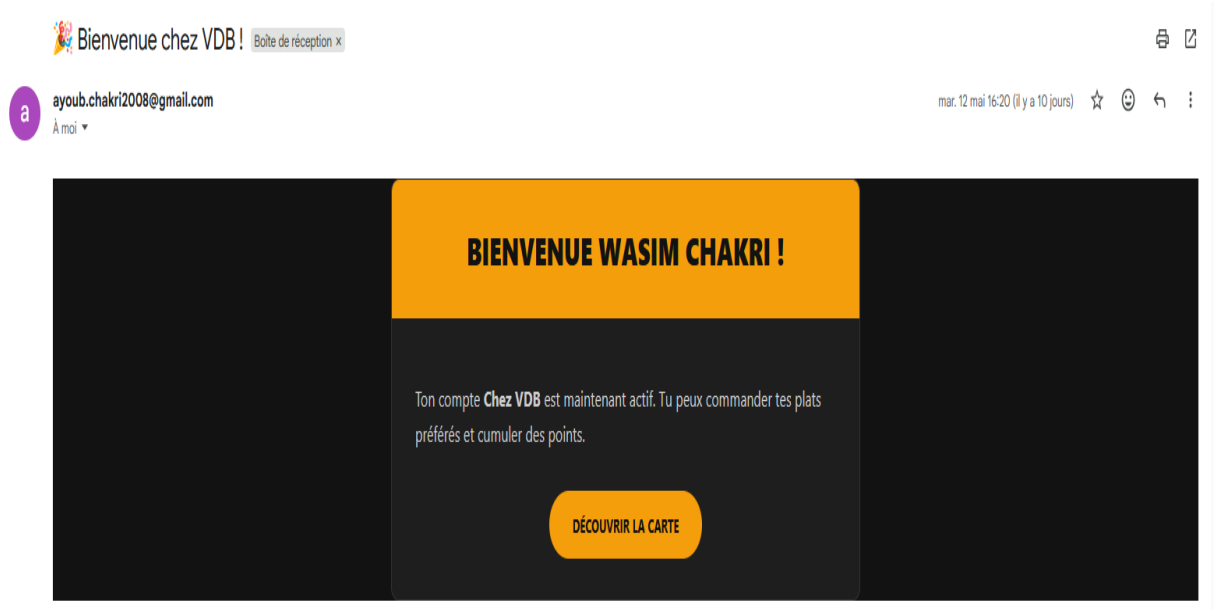
Côté cuisine – ajouté, modifier ou supprimer un nouveau plat :





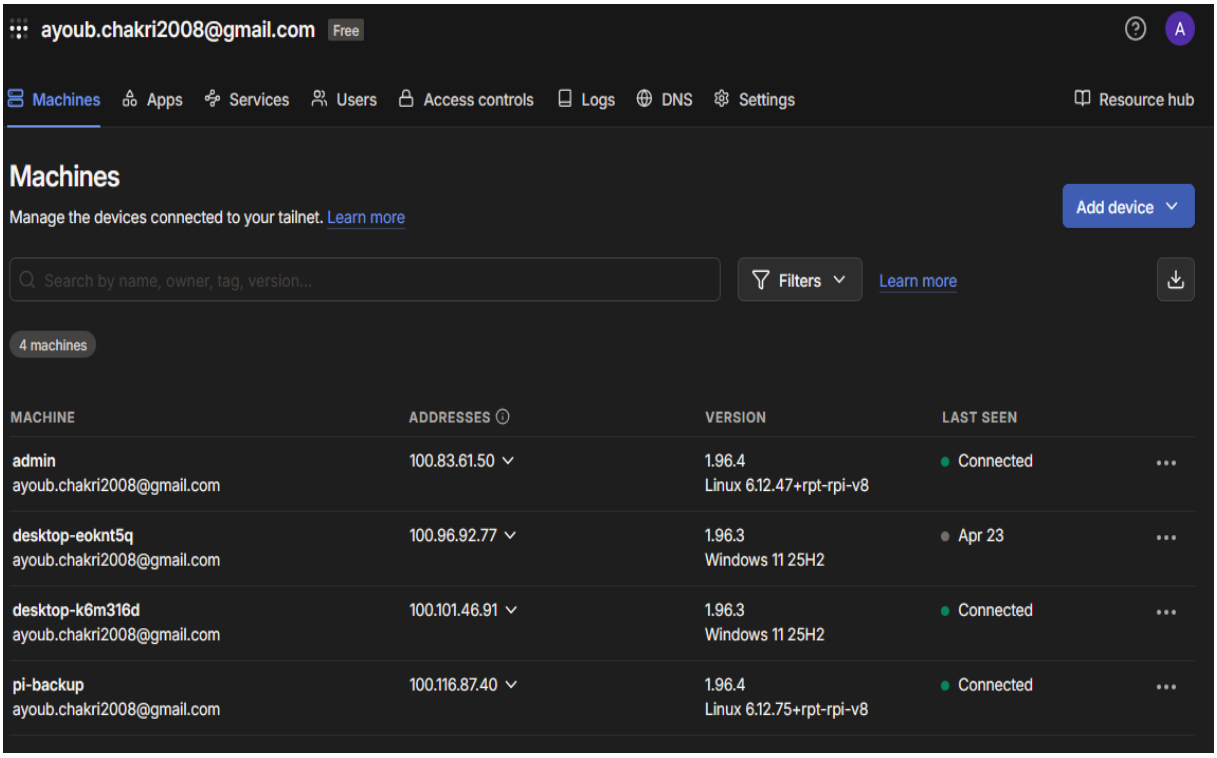
Valider la commande au comptoir après avoir scanné le QR code que le client a reçu dans le ticket par mail.

Un message de bienvenue envoyé au client lorsqu'il s'est inscrit en tant que membre.



Écran de suivi des commandes pour le client :

Les 2 Raspberry Pi 4 ainsi que mes 2 ordinateurs dans mon tunnel VPN :



Le script de sauvegarde des données vers le dossier Samba (montage du dossier) :

```
 GNU nano 8.4                                                    sauvegarde.sh

sudo umount -l /mnt/vdb_remote

sudo mount -t cifs //100.116.87.40/VDB_Backup /mnt/vdb_remote -o username=admin,password='Inraci#2025',uid=1000,gid=1000

if mountpoint -q /mnt/vdb_remote; then
    cp -ruL /home/admin/Desktop/TFE /mnt/vdb_remote/
    echo "La sauvegarde a été reuissi $(date)" >> /home/admin/log.txt

    sudo umount /mnt/vdb_remote
else

    echo "ERREUR : Le pi-backup est eteint et ne repond pas $(date)" >> /home/admin/log.txt
fi
```

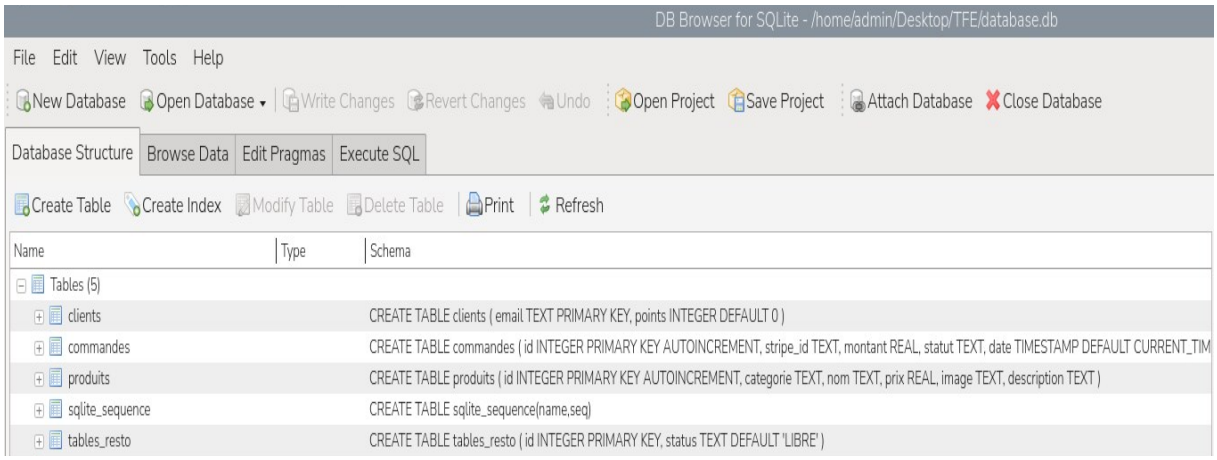
Config Firewall - le 1er Raspberry Pi 4 (bloque les entrées et accepte les sorties) :

```
root@admin:/home/admin# sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere
5900/tcp                   ALLOW IN    Anywhere
22/tcp (v6)                ALLOW IN    Anywhere (v6)
5900/tcp (v6)              ALLOW IN    Anywhere (v6)
```

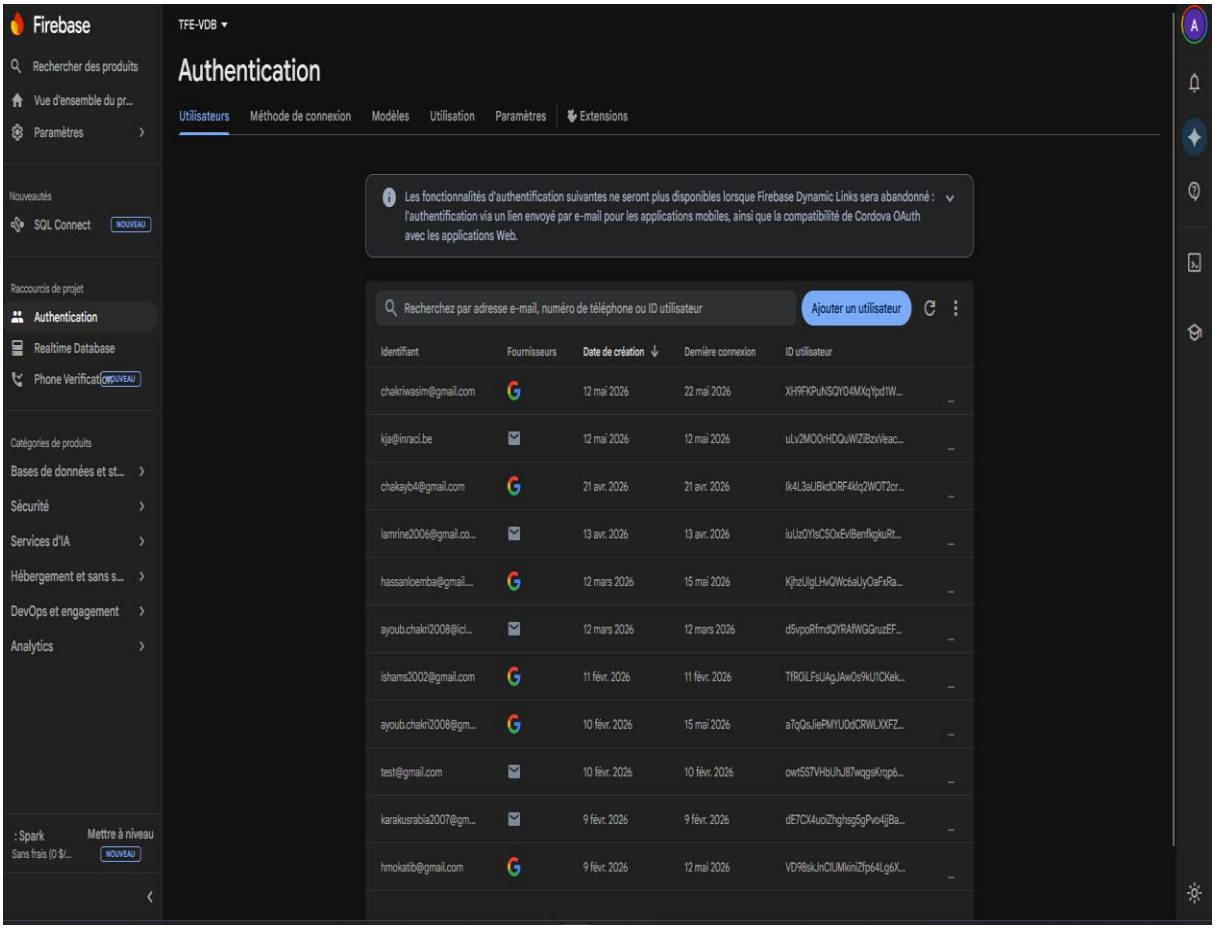
Config Firewall - le 2ème Raspberry Pi 4 (IDEM que le 1er mais il autorise le réseau Tailscale à rediriger les données vers le dossier samba) :

```
root@admin:/home/admin# sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere
5900/tcp                   ALLOW IN    Anywhere
22/tcp (v6)                ALLOW IN    Anywhere (v6)
5900/tcp (v6)              ALLOW IN    Anywhere (v6)
```

Les tables de données de la base de données de ma borne de commande :



La base de données du côté Firebase qui stocke les comptes clients :



Voici les liens où vous pourriez trouver l'ensemble des codes utilisés pour mon projet :

Mon code python FLASK :

https://www.mediafire.com/file/4lwkkj6n2euzwax/app.py/file

Ma base de données SQLITE :

https://www.mediafire.com/file/l5zeyy9vuikve9q/database.db/file

Mon dossier templates :

https://www.mediafire.com/folder/nkiqy5ztkadjk/templates

Mon code micro-python (ESP32 + Capteur PIR) :

https://www.mediafire.com/file/hyejdkmzgbs3jr1/main.py/file