Institut National de Radioélectricité et de Cinématographie

Enseignement technique secondaire de qualification

Accès aux études supérieures

Avenue Jupiter, 188

1190 Forest

Application de Commandes de Sandwiches Scolaires

Karakus Rabia

Pour l’obtention du certificat de qualification

Technicienne en informatique

Année scolaire : 2025-2026

Remerciement.

Je tiens tout d'abord à exprimer ma sincère gratitude à mes professeurs, Monsieur Kajjoua, Monsieur Mdiker, Monsieur Ben Sellam et Monsieur Janah, pour leurs précieux conseils, leur disponibilité ainsi que leurs encouragements tout au long de la réalisation de ce travail de fin d'études.

Je remercie également les membres du jury pour l'attention qu'ils porteront à ce travail lors de sa présentation.

Enfin, je remercie ma famille et mes proches pour leur soutien constant et leurs encouragements tout au long de cette année. Leur présence a été essentielle pour mener ce projet à bien.

Je n'oublie pas mes amis, qui m'ont accompagné et soutenu pendant l'élaboration de ce travail, et dont l'aide m'a été précieuse.

Pour finir, je remercie toutes les personnes qui, de près ou de loin, ont contribué à l'aboutissement de ce projet.

Table des matières

1) Introduction :

Aujourd'hui, la technologie est partout dans la vie des étudiants. On se sert souvent de son téléphone pour gagner du temps et simplifier des choses de tous les jours.

Dans les écoles, pendant les pauses, il y a souvent la queue pour acheter à manger, surtout des sandwiches. Ça fait perdre du temps, et des fois, certains élèves n'ont même pas le temps de manger calmement.

C'est pour ça que j'ai eu l'idée de créer SandwichSchool, une application où les élèves peuvent commander leurs sandwiches à l'avance depuis leur téléphone, mais seulement dans leur école.

Le but de ce travail est donc de créer cette application et de proposer une solution simple pour mieux organiser les pauses.

Pour faire ce projet, j'ai suivi plusieurs étapes : d'abord réfléchir à ce qu'il fallait, puis imaginer l'application, et enfin essayer de la développer.

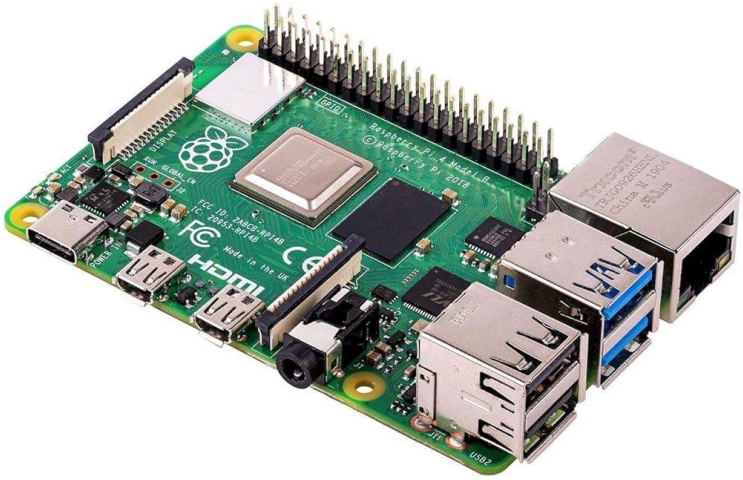
Dans ce rapport, je vais d'abord présenter le projet et ce qu'il fallait, ensuite expliquer comment j'ai pensé l'application, et enfin montrer ce que j'ai fait.

2) Les outils

2.1. Matériels utilisés

Matériel	Quantité	Prix
Rasbperry pi 4	1	113,99€

2.1.1 Raspberry pi 4



Le Raspberry Pi 4 est un mini-ordinateur. C'est la fondation Raspberry Pi, une association anglaise, qui l'a créé. Il est sorti en juin 2019 et il est plus puissant que les versions d'avant. Même s'il est tout petit (85 mm x 56 mm), il a presque tout ce qu'un ordinateur normal a. Il a un processeur à quatre cœurs qui va à 1,8 GHz, et on peut choisir la mémoire RAM : 2 Go, 4 Go ou 8 Go. Il marche surtout avec Linux, grâce à un système appelé Raspberry Pi OS.

Pour ce projet, le Raspberry Pi 4 sert de serveur local pour faire tourner l'application SandwichSchool. C'est lui qui lance Flask et aussi la base de données SQLite. Comme ça, l'application peut être utilisée sur les appareils connectés au réseau de l'école.

2.2 Logiciels utilisés

Logiciels	Description	Prix
Visual Studio Code	Environnement de développement sur Windows	0.00€
SQLite	Système de gestion de base de données	0.00€
Raspberry Pi OS	Système d'exploitation installé sur le Raspberry Pi	0.00€
Cloudfare	Créer des tunnel sécurisé	0.00€
Tailscale	Créer des réseaux virtuels	0.00€
TigerVNC	Connexion à distance	0.00€
Stripe	Créer des paiements sécurisé	0.00€
Thonny IDE	Environnement de Développement sur Raspberry pi 4	0.00€
DB Browser For SQLite	Logiciel qui permet d'exploiter un fichier SQ	0.00€

Comme on peut le voir, ce projet ne m'a rien coûté.

2.2.1 Visual Studio Code

C'est quoi ?

C'est un environnement de développement gratuit, léger et conçu par Microsoft. Il est disponible sur plusieurs systèmes d'exploitation comme Windows (10 et 11), macOS et Linux. Il est utilisé pour la programmation moderne.

Pour mon projet, je l'ai utilisé pour pouvoir coder mon application, plus précisément la partie Python/ Flask et de rajouter les templates HTML et CSS

2.2.2 SQLite

C’est quoi ?

SQLite est un système pour gérer les bases de données, léger et facile à utiliser. Il permet de ranger, organiser et retrouver facilement des infos, par exemple les comptes des utilisateurs ou les commandes.

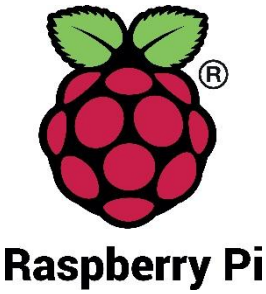
Contrairement à d'autres systèmes comme MySQL, SQLite n'a pas besoin de serveur. Toutes les données sont mises dans un seul fichier, ce qui est super pratique pour les petits projets ou les applications locales. C'est aussi rapide, facile à installer et il n'y a presque rien à configurer.

Du coup, on l'utilise souvent dans les projets d'école, les applis mobiles ou sur des appareils comme le Raspberry Pi.

Dans mon projet SandwichSchool, j'utilise SQLite pour enregistrer les infos des utilisateurs et les commandes de sandwichs. Ça permet de garder toutes les données bien rangées et d'y accéder vite depuis l'application.

2.2.3 Rasbperry PI OS

C’est quoi ?



Raspberry Pi OS, c'est le système qu'on utilise sur le Raspberry Pi. C'est un système basé sur Linux, fait exprès pour marcher sur ce petit ordinateur. Il gère le matériel (comme le processeur, la mémoire, etc.) et permet de lancer des programmes, par exemple une application web ou une base de données.

Il existe avec une interface graphique (comme un ordinateur normal) ou en version plus simple, juste avec des lignes de commande.

Pour mon projet SandwichSchool, Raspberry Pi OS sert à faire tourner le serveur local. Il permet de lancer l'application Flask et la base de données, comme ça les utilisateurs peuvent se connecter au site depuis le réseau de l'école.

2.2.4 CloudFare

C’est quoi ?

Cloudflare est un service sur internet qui aide à sécuriser un site web et à le rendre plus rapide. Il fait le lien entre les utilisateurs et le serveur.

Grâce à Cloudflare, le site est mieux protégé contre des attaques. Ça permet aussi que les pages se chargent plus vite grâce à un système de cache.

Pour mon projet, Cloudflare sert à montrer mon application web sur mon adresse « sandwichschool.be ». J'ai mis en place un tunnel entre Cloudflare et mon Raspberry Pi 4, qui est le serveur. Le Raspberry Pi se connecte à Cloudflare, qui fait l'intermédiaire pour que l'application soit accessible en toute sécurité via l'adresse du site.

2.2.5 Tailscale

C’est quoi ?

Tailscale est un outil pour connecter plusieurs appareils entre eux de façon sécurisée, via internet.

C'est comme un réseau privé où tous les appareils peuvent parler entre eux comme s'ils étaient sur le même réseau local, même s'ils sont à des endroits différents.

Pour mon projet, Tailscale aide à accéder au serveur sans devoir faire des réglages réseau compliqués, comme ouvrir des ports. Ça permet aussi de garder une connexion sécurisée entre les appareils.

J'utilise ça parce qu'au début, je ne pouvais travailler qu'en local, et je devais toujours emporter tout mon matériel. Maintenant, grâce à mon réseau Tailscale (tailnet), je peux laisser mon Raspberry Pi 4 chez moi et continuer à travailler dessus de n'importe où, simplement et en toute sécurité.

2.2.5 TigerVNC

C’est quoi ?

TigerVNC est un logiciel pour prendre le contrôle d'un ordinateur à distance, via le protocole VNC.

Ça permet de voir l'écran d'un ordinateur loin de nous sur un autre ordinateur. TigerVNC est gratuit, marche bien et peut s'adapter même avec une connexion lente.

Dans mon projet, j'utilise TigerVNC pour me connecter à distance à mon Raspberry Pi 4. Comme ça, je peux travailler dessus sans avoir besoin d'un écran, d'un clavier ou d'une souris directement, ce qui est plus pratique et flexible.

2.2.6 Stripe

C’est quoi ?

Stripe est une entreprise qui offre des solutions pour mettre des systèmes de paiement sécurisés dans les applications.

Dans mon projet, j'utilise Stripe pour ajouter un système de paiement à mon application. Les utilisateurs peuvent payer facilement par carte bancaire ou par d'autres moyens, et c'est sécurisé.

Mais attention, le système est en mode test, donc les vraies transactions ne sont pas actives

2.2.7 Thonny IDE

C’est quoi ?

C'est un programme gratuit et open source pour le développement. Il est fait pour aider à apprendre la programmation, surtout avec Python et MicroPython.

Son interface est simple et facile à comprendre, donc c'est très facile à utiliser, surtout pour ceux qui débutent.

Pour mon projet, ce logiciel m'a aidé à installer les éléments nécessaires sur le Raspberry Pi 4, que j'ai utilisé pour développer mon application.

2.2.8 DB Browser For SQLite

C'est quoi ?

DB Browser for SQLite est un logiciel gratuit et open source pour gérer facilement une base de données SQLite, grâce à un écran où on clique.

Ça évite de tout faire en écrivant des lignes de commande, donc c'est plus simple à utiliser. Avec ce logiciel, on peut créer des tableaux, ajouter ou modifier des infos, et voir directement ce qu'il y a dans la base.

Dans mon projet, j'ai utilisé DB Browser for SQLite pour gérer la base de données de mon Raspberry Pi 4. Ça m'a permis de vérifier mes données et de tester facilement sans devoir tout faire avec du code.

2.3 Langages utilisé

Nom	Description
Python	Flask avec templates HTML ET CSS

2.3.1 Python

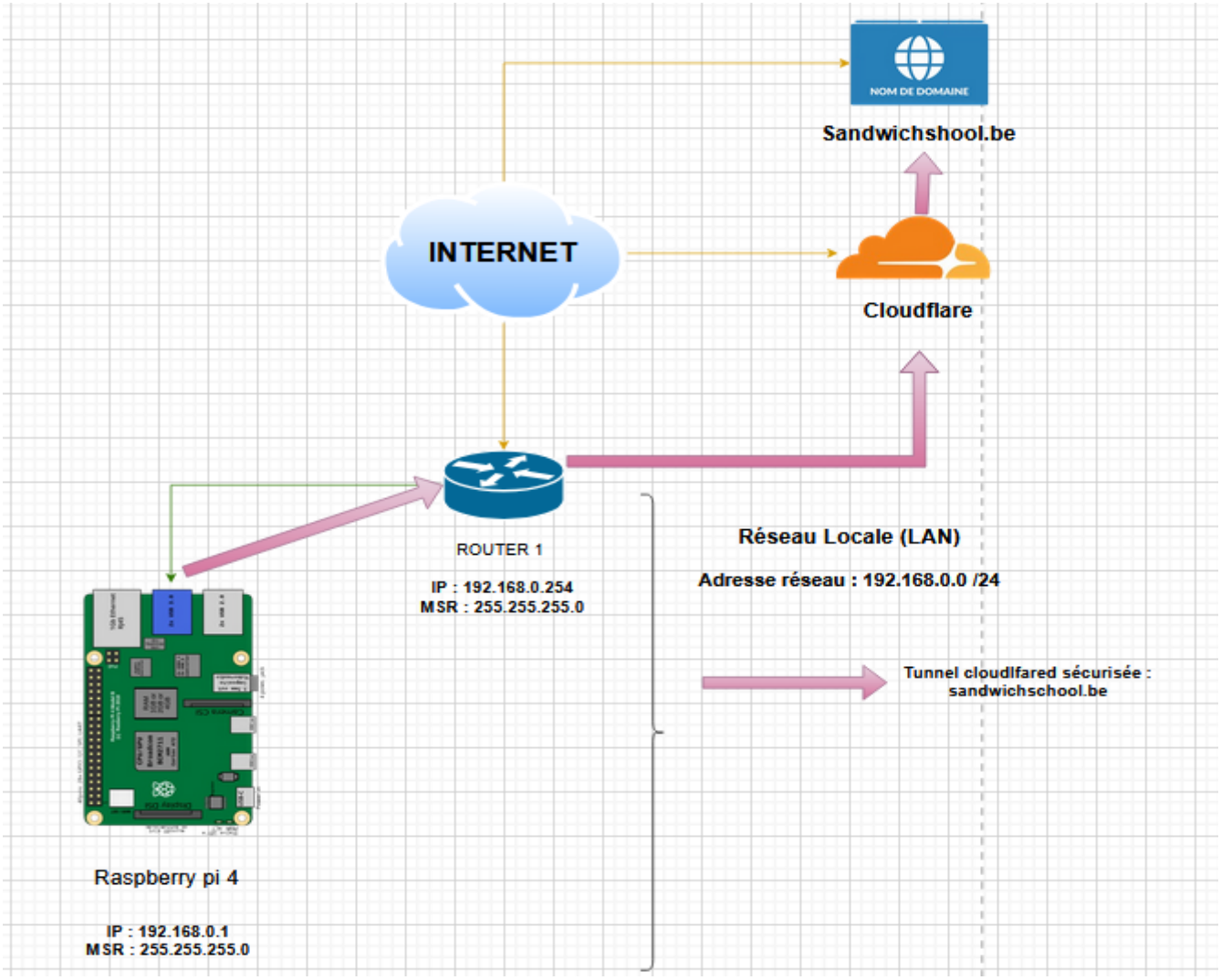
C'est quoi ?

Python est un langage de programmation open source et gratuit, créé par Guido van Rossum en 1991. Aujourd'hui, il est largement utilisé dans de nombreux domaines tels que le développement web, l'analyse de données, l'intelligence artificielle ou encore l'automatisation.

Dans mon projet, j'utilise Python principalement avec le framework Flask. Ça me permet de développer une application web en utilisant des templates HTML et CSS. L'objectif est de créer une application web fiable, fluide et accessible.

3) Les schémas

3.1 La topologie



Explication de ma topologie :

Cette organisation montre comment mon application marche avec le réseau.

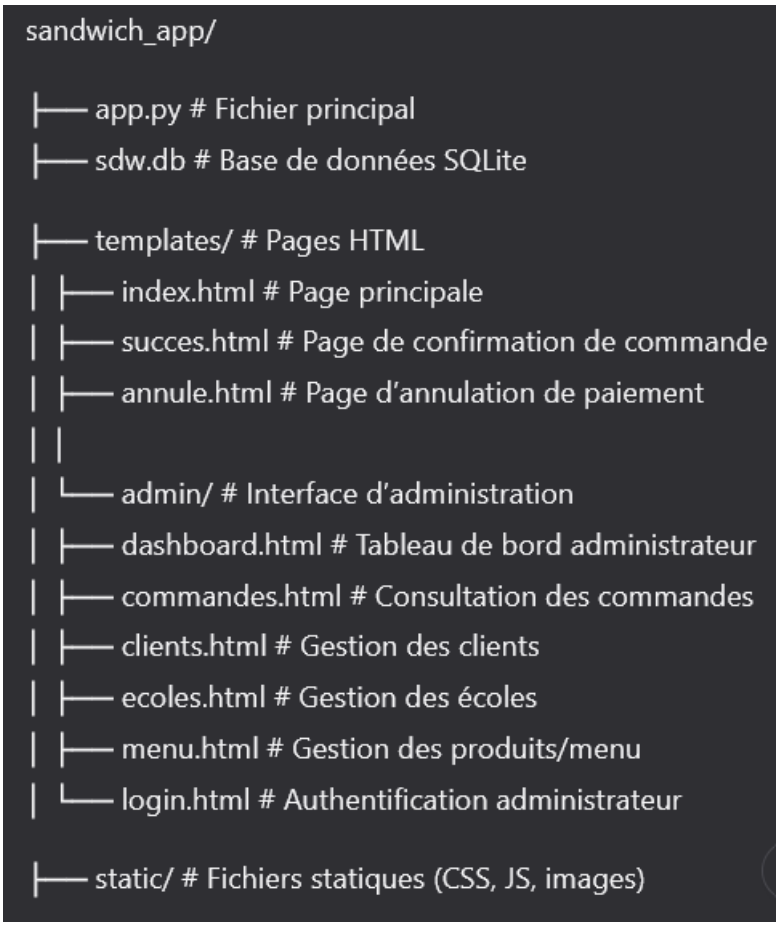
Le Raspberry Pi est branché au réseau local (LAN). Il a une adresse IP (192.168.0.1) et c'est lui qui héberge l'application.

Le routeur permet de relier le réseau local à Internet. Lui aussi a une adresse IP (192.168.0.254).

Quand un utilisateur veut aller sur le site (sandwichschool.be), sa demande passe par Internet, puis par Cloudflare. Cloudflare sécurise la connexion et renvoie la demande vers le réseau local grâce à un tunnel sécurisé.

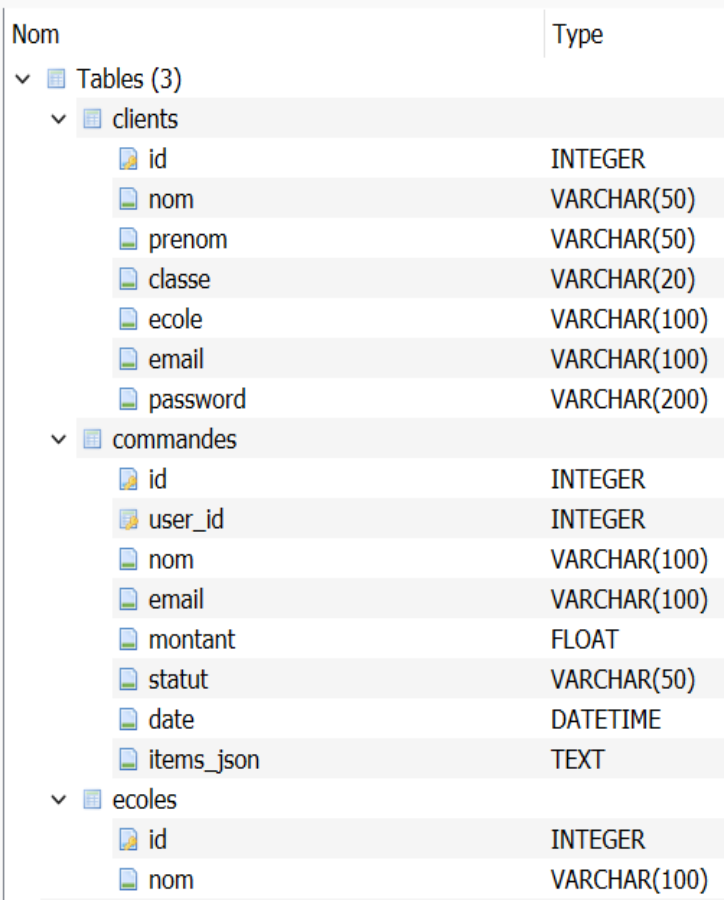
Après, la demande arrive au Raspberry Pi, qui envoie la réponse au client.

3.2 Organisation des fichiers



3.3 Base de données

La base de données est composée de trois tables principales :



➔ La table "**clients**" contient les infos des utilisateurs : leur nom, prénom, classe, école, email et mot de passe. Il sert à identifier chaque utilisateur de façon unique.

➔ La table "**commandes**" enregistre ce que les utilisateurs ont commandé. On y trouve l'identifiant du client, le montant, l'état de la commande, la date et aussi les produits commandés sous forme de données JSON.

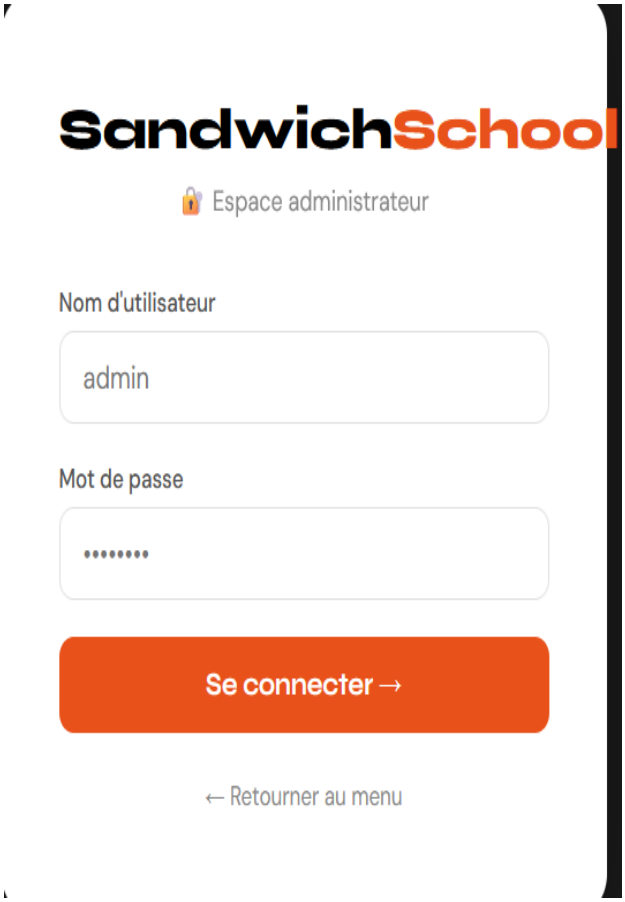
➔ La table "**ecoles**" sert à garder la liste des écoles disponibles dans l'application.

Ces différents tableaux sont liés entre eux, surtout grâce à l'identifiant du client (user_id). Ça permet de relier chaque commande à un utilisateur précis.

Ces différentes tables sont liées entre elles, notamment grâce à l'identifiant du client (user_id), ce qui permet d'associer chaque commande à un utilisateur précis.

4) Travail réaliser

4.1 Coté admin



1) La page de connexion :

Cette page permet à l'administrateur d'aller sur la partie admin de l'application.

Pour se connecter, il doit écrire son nom d'utilisateur (admin) et son mot de passe. Une fois les infos validées, l'administrateur peut accéder aux différentes fonctions de gestion : gérer les données, les utilisateurs (ajouter, modifier, supprimer).

Cette étape sécurise l'accès et évite que n'importe qui puisse changer les infos.

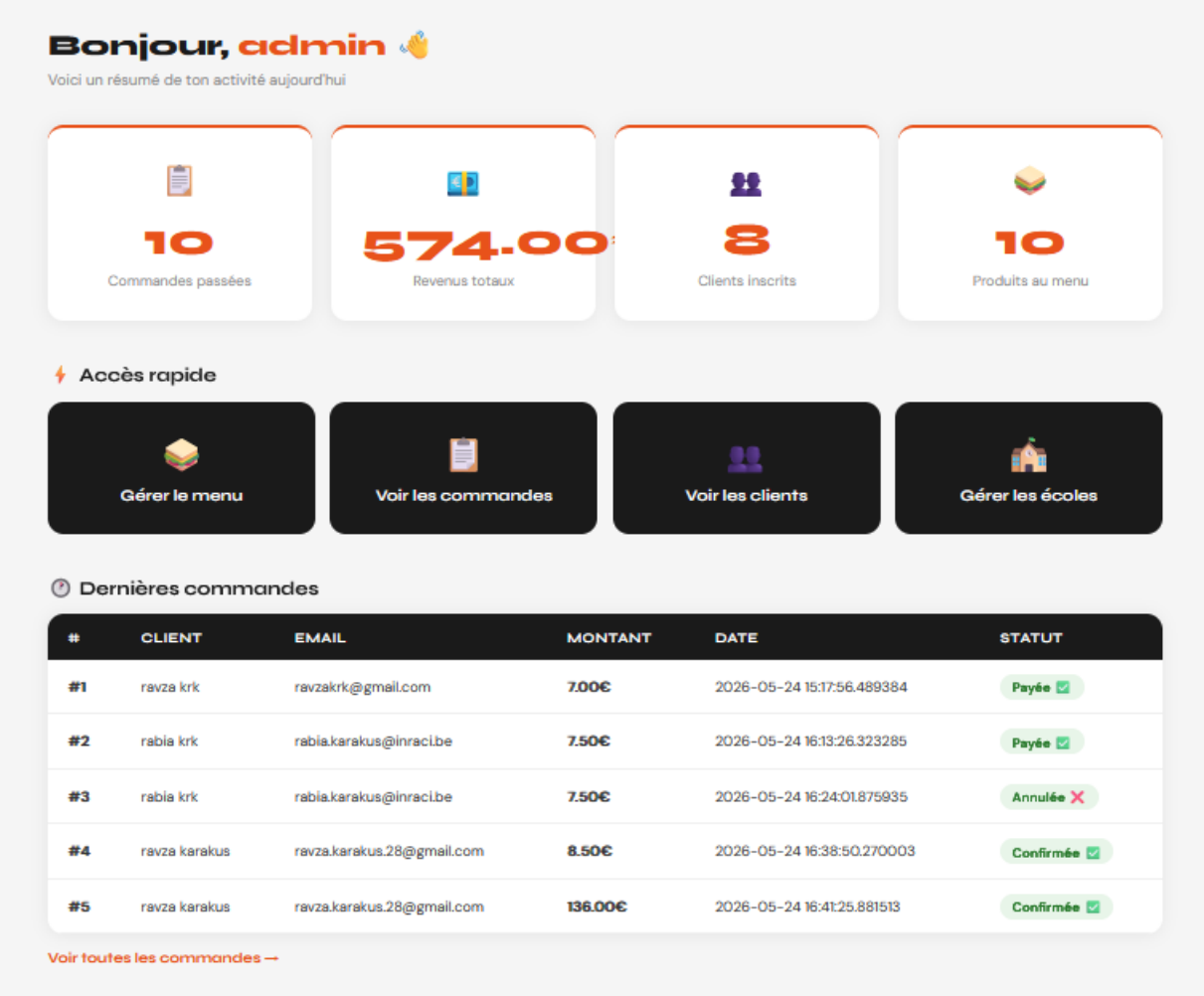
2. Tableau de bord admin :

Après s'être connecté, l'admin arrive sur le tableau de bord principal.

Cette page montre un résumé de ce qui se passe sur l'application : le nombre de commandes faites, l'argent gagné, le nombre de clients inscrits et les produits au menu.

Le tableau de bord permet aussi d'accéder vite aux différentes fonctions : gérer le menu, voir les commandes, gérer les clients et les écoles.

Cette interface aide l'administrateur à tout voir d'un coup et à gérer facilement l'application.



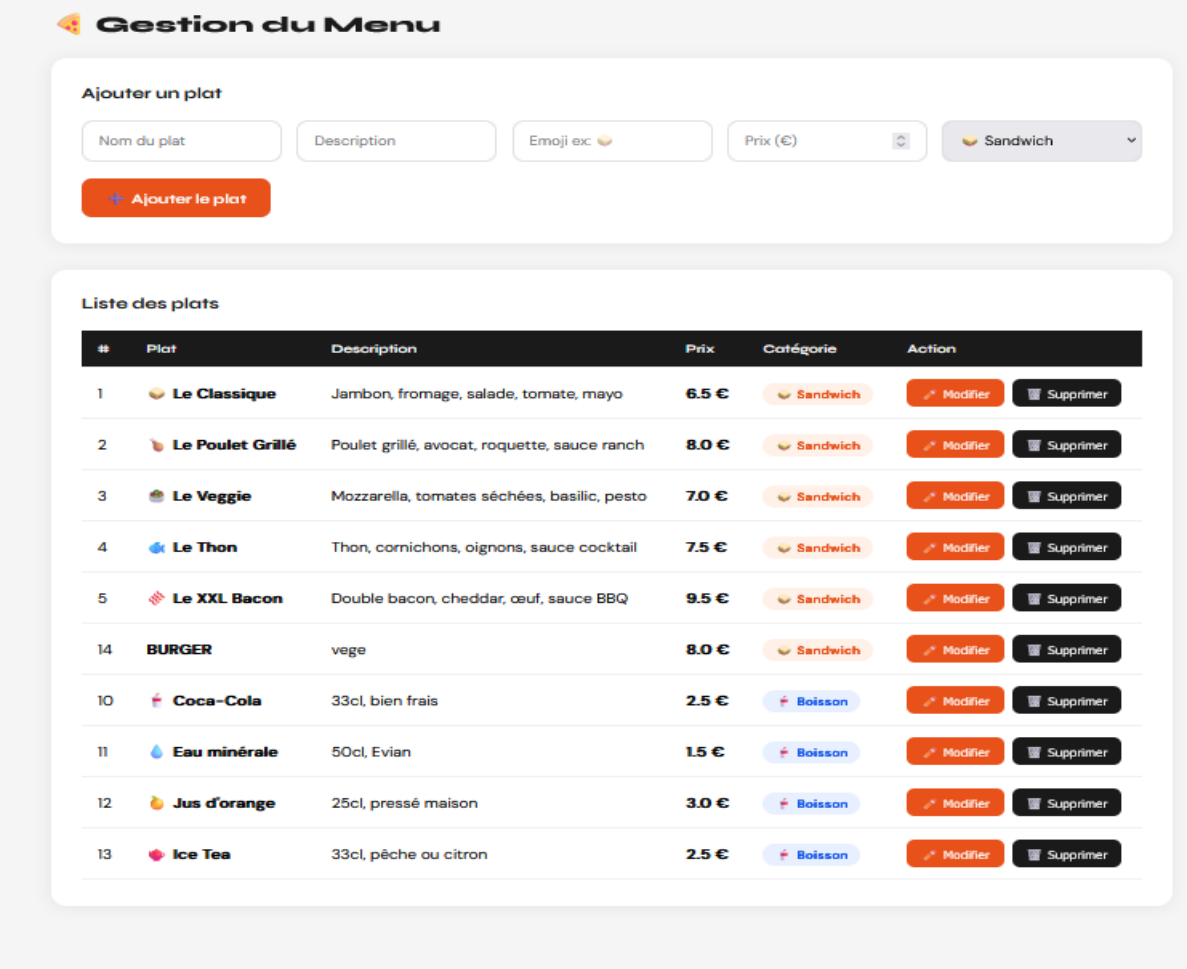
3. Gestion du menu :

Sur cette page, l'admin peut gérer les différents produits du menu.

Il peut ajouter un nouveau plat en mettant son nom, sa description, son prix, une icône (emoji) et sa catégorie.

La liste des plats déjà là est aussi affichée, avec leurs infos principales (nom, description, prix et catégorie). L'admin peut modifier ou supprimer un plat avec les boutons prévus pour ça.

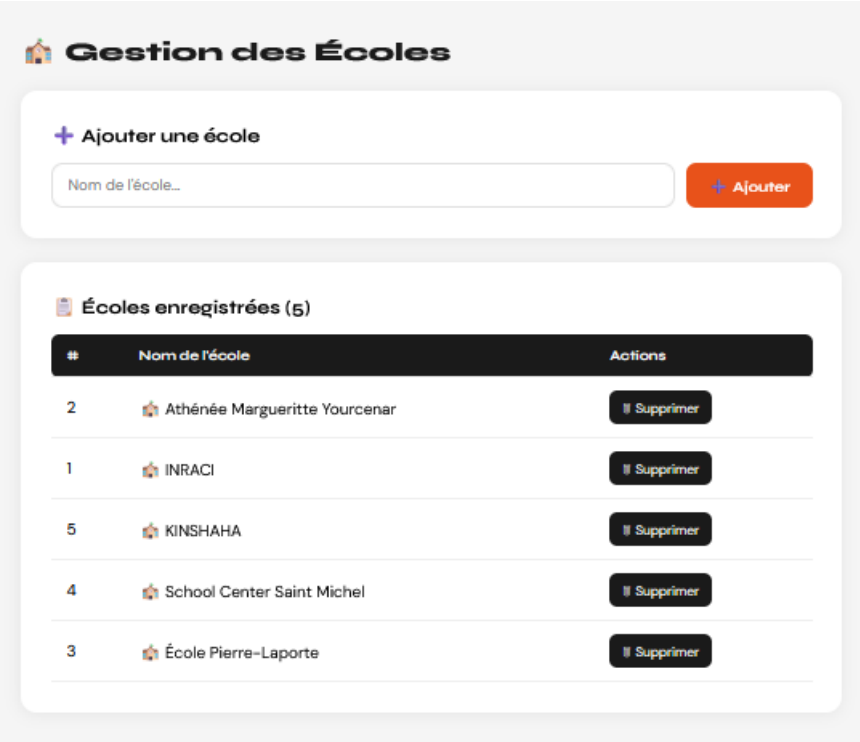
Ça permet de garder le menu à jour facilement.



4.Gestion des écoles :

Sur cette page, l'administrateur peut ajouter une école en écrivant simplement son nom.Il peut aussi voir toutes les écoles déjà enregistrées dans une liste.

Si besoin, il peut supprimer une école facilement. Ça permet de garder les informations à jour.

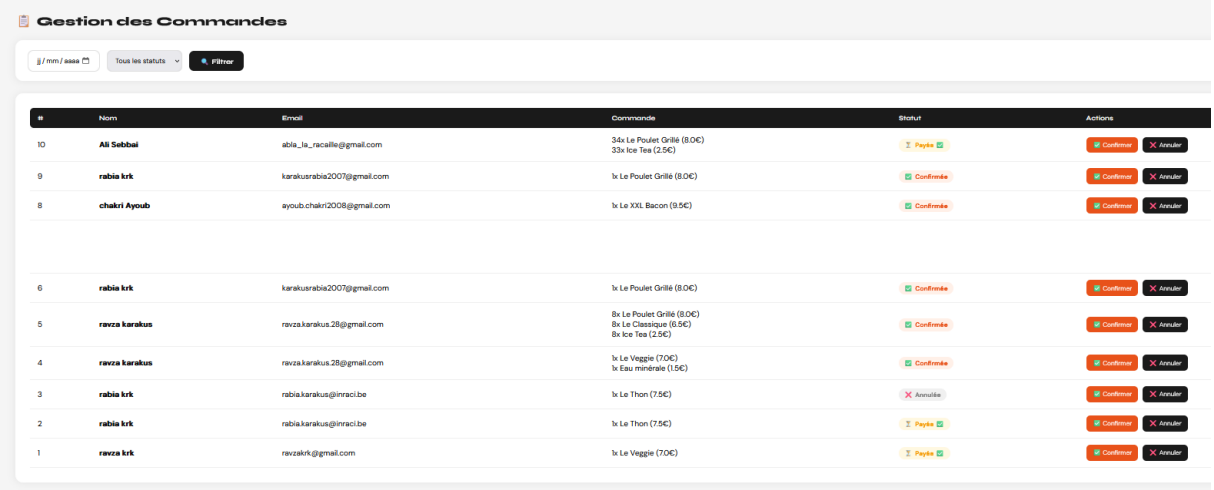


5) Gestion des commandes :

Sur cette page, l'administrateur peut voir toutes les commandes faites par les utilisateurs. Chaque commande montre le nom, l'email, les produits commandés et son état.

On peut trier les commandes par état ou par date pour en retrouver une plus facilement. L'administrateur peut aussi confirmer ou annuler une commande.

Ça permet de bien gérer les commandes tous les jours.

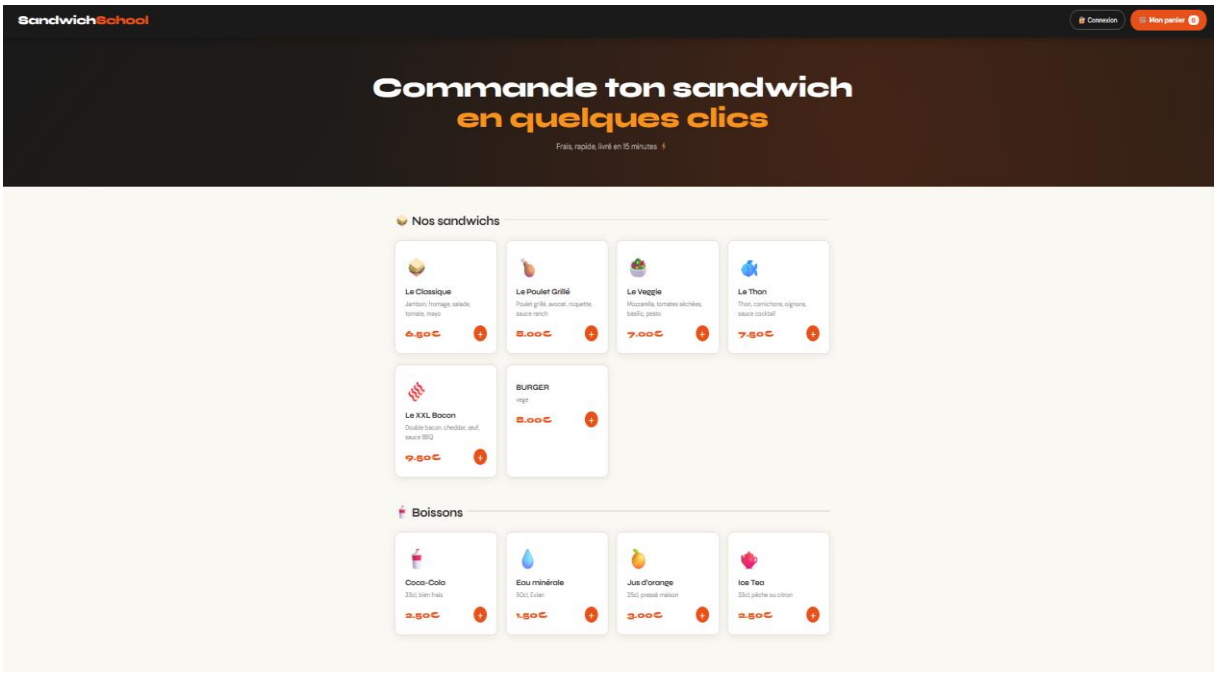


4.2 Coté client

1. Page de commande :

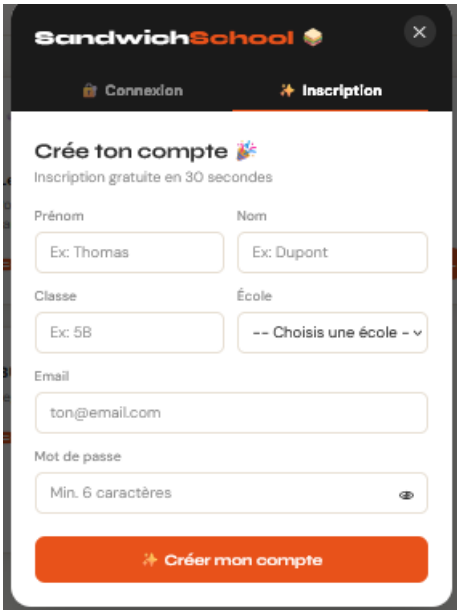
Sur cette page, l'utilisateur peut choisir les produits qu'il veut commander. Il y a différents sandwiches et boissons avec leur prix. Chaque produit peut être ajouté au panier en cliquant sur le bouton.

L'utilisateur peut donc créer sa commande facilement en quelques clics. En haut de la page, il peut aussi voir son panier ou se connecter à son compte.

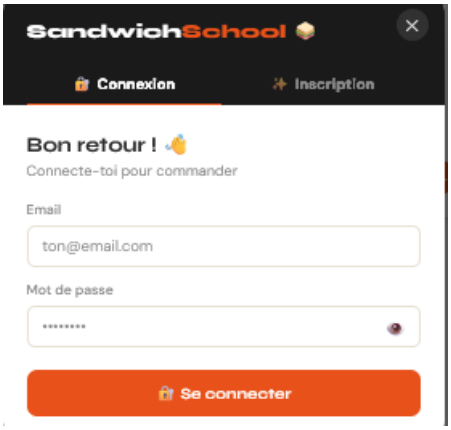


2) Page de connexion/inscription :

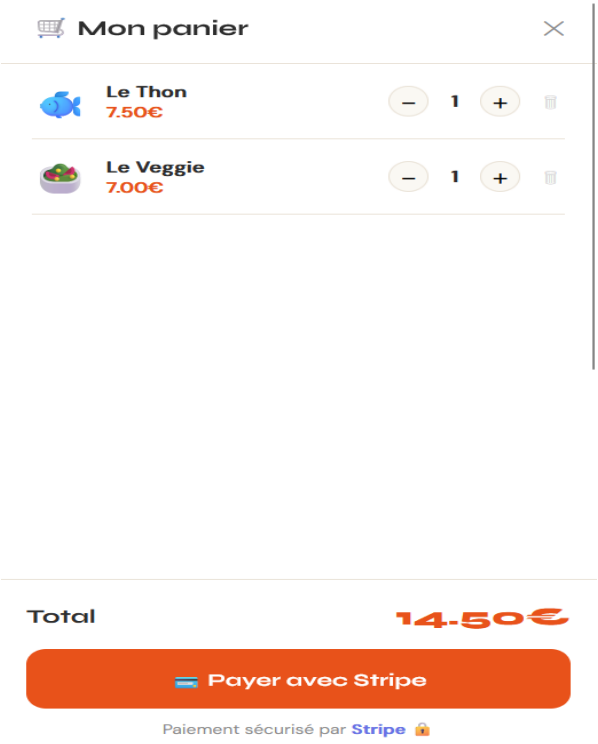
L'utilisateur peut créer un compte ou se connecter à un compte existant. Pour l'inscription, il doit remplir ses informations comme son prénom, nom, classe, école, email et mot de passe.



Une fois le compte créé, il peut se connecter avec son email et son mot de passe. Cela permet de garder un suivi des commandes et de faciliter l'utilisation du site.



3) Paiement :

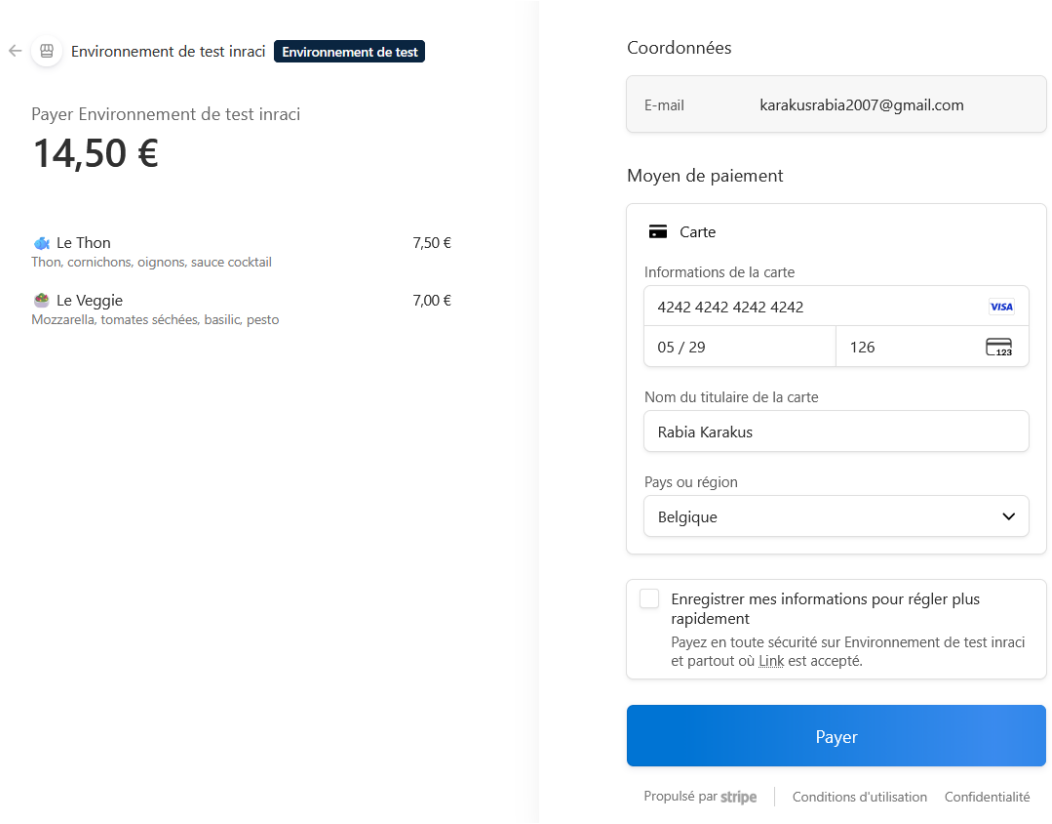


Dans le panier, l'utilisateur peut voir les produits qu'il a ajoutés avec leur prix et la quantité.

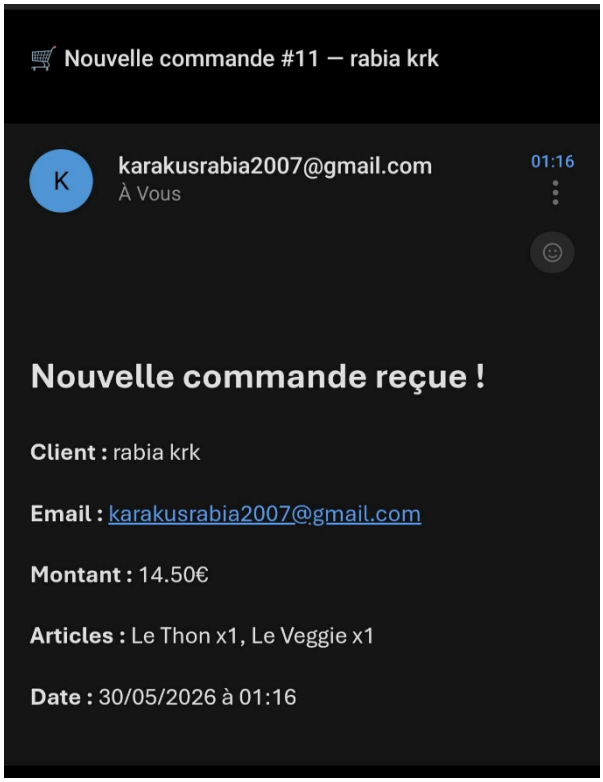
Il peut modifier la quantité ou supprimer un produit facilement.

Le total de la commande est calculé automatiquement.

Pour finaliser la commande, l’utilisateur peut payer en ligne grâce à Stripe en entrant ses informations de carte bancaire. Le paiement est sécurisé, ce qui permet de commander en toute confiance.



4) Confirmation de commande :



Après le paiement, une page de confirmation s'affiche.

L'utilisateur y voit un message indiquant que sa commande est confirmée, avec un récapitulatif des produits commandés et le prix total.

Un email de confirmation est envoyé à l'utilisateur.

Une fois la commande passée, l'administrateur reçoit également un email.

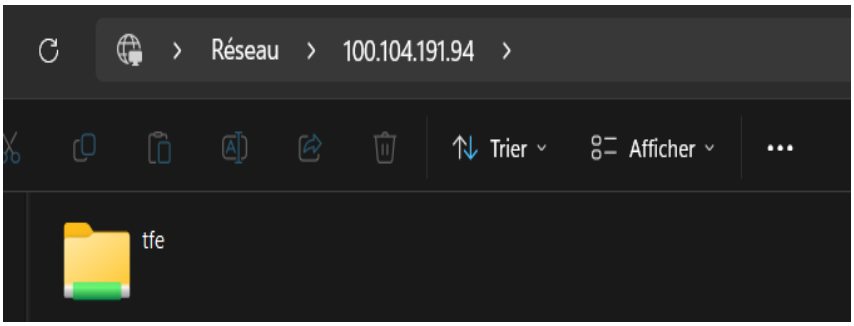
Ce dernier contient toutes les informations importantes de la commande : le nom du client, son adresse email, les articles commandés, le montant total, ainsi que la date et l'heure.

5) Le serveur samba

C’est quoi ?

Un serveur Samba a été mis en place pour faciliter le partage de fichiers sur le réseau local. Il permet d'accéder aux ressources du Raspberry Pi depuis d'autres ordinateurs, surtout sous Windows, comme si c'était un dossier partagé classique.

Cela simplifie le transfert de fichiers, la gestion du projet et la maintenance de l'application.



Le dossier partagé via Samba est accessible uniquement aux appareils connectés à mon tunnel VPN. Ainsi, seuls les dispositifs autorisés peuvent y accéder, et aucun appareil extérieur au tunnel ne peut consulter ce dossier.

```
[TFE]

path = /home/TFE
browsable = yes
writable = yes
valid users = @smbgrp
guest ok = no
```

C'est la configuration que j'ai faite dans le fichier « smb.conf » pour transférer mes fichiers vers le Raspberry Pi 4.

.

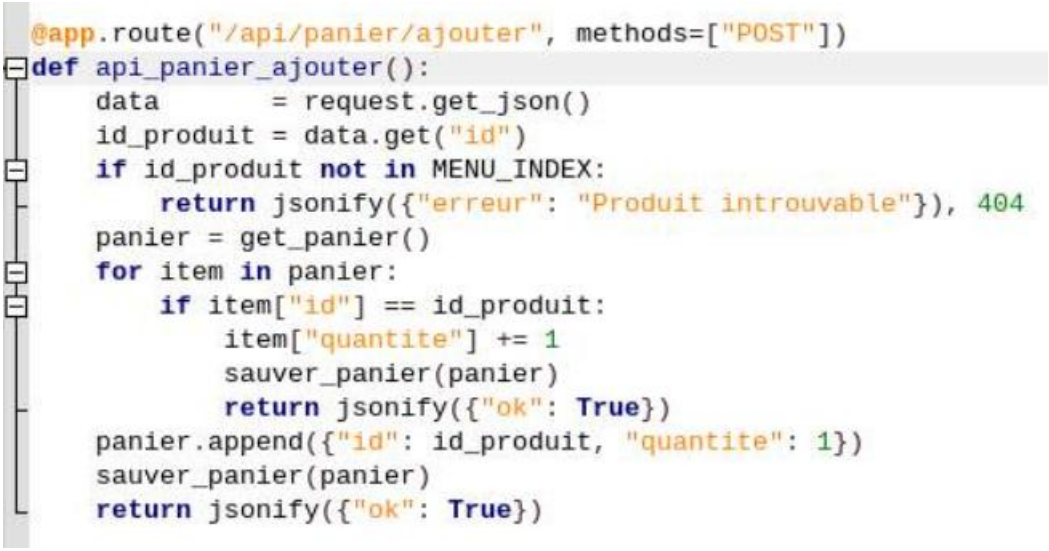
6) Réalisation

6.1 Explication du fonctionnement

La partie frontend permet à l'utilisateur de parcourir l'application et de commander ses sandwiches. En arrière-plan, le backend assure la gestion et le bon fonctionnement de l'application. Celui-ci a été développé en Python à l'aide du framework Flask.

6.1.1 Système de panier

Ce bout de code sert à ajouter un produit dans le panier de l'utilisateur. C'est une route API, créée avec Flask, qui est appelée quand on envoie une requête de type POST.



```python
@app.route("/api/panier/ajouter", methods=["POST"])
def api_panier_ajouter():
    data       = request.get_json()
    id_produit = data.get("id")
    if id_produit not in MENU_INDEX:
        return jsonify({"erreur": "Produit introuvable"}), 404
    panier = get_panier()
    for item in panier:
        if item["id"] == id_produit:
            item["quantite"] += 1
            sauver_panier(panier)
            return jsonify({"ok": True})
    panier.append({"id": id_produit, "quantite": 1})
    sauver_panier(panier)
    return jsonify({"ok": True})
```

D'abord, le programme récupère les données envoyées, surtout l'ID du produit. Ensuite, il vérifie si ce produit existe bien dans le menu. Si ce n'est pas le cas, il renvoie une erreur avec un message.

Si le produit existe, le code récupère le panier actuel. Il vérifie si le produit est déjà dedans : si oui, il augmente simplement la quantité de 1 ; sinon, il ajoute le produit avec une quantité de 1.

À la fin, le panier est sauvegardé et le programme renvoie une réponse pour indiquer que tout s'est bien passé.

6.1.2 Authentification des clients

Ce code gère l'inscription d'un utilisateur dans l'application. C'est une route API qui est appelée quand quelqu'un remplit le formulaire d'inscription.

D'abord, le programme récupère toutes les informations envoyées : nom, prénom, classe, école, email et mot de passe.

Il vérifie ensuite que tous les les champs sont remplis. Si ce n'est pas le cas, il renvoie une erreur. Ensuite, il vérifie que le mot de passe contient au moins 6 caractères et que l'email n'est pas déjà utilisé par un autre utilisateur.

Après cela, le code vérifie si l'école existe déjà dans la base de données. Si elle n'existe pas, elle est créée automatiquement. Un nouvel utilisateur est ensuite créé avec les informations données. Le mot de passe est sécurisé (il est transformé en une version protégée).

L'utilisateur est enregistré dans la base de données. Enfin, les informations de l'utilisateur sont stockées dans la session, ce qui le maintien connecté.

Le programme renvoie alors un message de bienvenue pour confirmer que l'inscription a réussi

```python
@app.route("/api/inscription", methods=["POST"])
def inscription():
    data     = request.get_json()
    nom      = data.get("nom",      "").strip()
    prenom   = data.get("prenom",   "").strip()
    classe   = data.get("classe",   "").strip()
    ecole    = data.get("ecole",    "").strip()
    email    = data.get("email",    "").strip().lower()
    password = data.get("password", "")

    if not all([nom, prenom, classe, ecole, email, password]):
        return jsonify({"erreur": "Tous les champs sont obligatoires"}), 400
    if len(password) < 6:
        return jsonify({"erreur": "Mot de passe trop court (6 caractères min)"}), 400
    if User.query.filter_by(email=email).first():
        return jsonify({"erreur": "Cet email est déjà utilisé"}), 400

    # AJOUTER : Créer l'école en BDD si elle n'existe pas
    ecole_obj = Ecole.query.filter_by(nom=ecole).first()
    if not ecole_obj:
        ecole_obj = Ecole(nom=ecole)
        db.session.add(ecole_obj)
        db.session.commit()

    new_user = User(
        nom=nom, prenom=prenom, classe=classe,
        ecole=ecole, email=email,
        password=generate_password_hash(password)
    )
    db.session.add(new_user)
    db.session.commit()

    session["user_id"]     = new_user.id
    session["user_prenom"] = new_user.prenom
    session["user_nom"]    = new_user.nom

    return jsonify({"message": f"Bienvenue {new_user.prenom} ! [illegible]"})
```

```python
@app.route("/api/connexion", methods=["POST"])
def connexion():
    data     = request.get_json()
    email    = data.get("email",    "").strip().lower()
    password = data.get("password", "")

    user = User.query.filter_by(email=email).first()
    if not user or not check_password_hash(user.password, password):
        return jsonify({"erreur": "Email ou mot de passe incorrect"}), 401

    session["user_id"]     = user.id
    session["user_prenom"] = user.prenom
    session["user_nom"]    = user.nom

    return jsonify({"message": f"Bon retour {user.prenom} !", "prenom": user.prenom})
```

Ce code gère la connexion d'un utilisateur. C'est une route API utilisée quand quelqu'un essaie de se connecter à son compte.

Le programme commence par récupérer les données envoyées : l'email et le mot de passe.

Ensuite, il cherche dans la base de données si un utilisateur avec cet email existe. Si l'utilisateur n'existe pas ou si le mot de passe est incorrect, le programme renvoie une erreur indiquant que les informations sont incorrectes.

Si les informations sont correctes, les données de l'utilisateur sont enregistrées dans la session. Cela permet de le maintenir connecté sur le site. Enfin, le programme renvoie un message de bienvenue avec le prénom de l'utilisateur pour confirmer que la connexion a réussi.

6.1.3 Paiement avec Stripe

Ce code gère le paiement du panier avec Stripe. C'est une route API qui est appelée quand l'utilisateur veut payer sa commande.

D'abord, le programme vérifie si l'utilisateur est bien connecté. Si ce n'est pas le cas, il renvoie une erreur.

Ensuite, il récupère les informations de l'utilisateur, comme son nom et son email. Après cela, le code récupère le panier. Si le panier est vide, une erreur est renvoyée. Ensuite, le programme prépare les produits pour Stripe.

Pour chaque produit du panier, il récupère son nom, sa description, son prix et la quantité. Il convertit aussi le prix en centimes, car c'est comme ça que fonctionne Stripe.

Avant de lancer le paiement, le panier est sauvegardé dans la session. Cela permet de garder une trace des produits même après la redirection vers Stripe.

Ensuite, une session de paiement Stripe est créée avec toutes les informations nécessaires : les produits, l'email du client et les liens de retour en cas de succès ou d'annulation.

Enfin, le programme renvoie une URL de paiement. Si une erreur se produit, un message d'erreur est renvoyé.

```python
@app.route("/api/paiement/creer-session", methods=["POST"])
def creer_session_paiement():
    if "user_id" not in session:
        return jsonify({"erreur": "Tu dois être connecté pour payer"}), 401

    user = User.query.get(session["user_id"])
    if not user:
        return jsonify({"erreur": "Utilisateur introuvable"}), 404

    client_nom   = f"{user.prenom} {user.nom}"
    client_email = user.email

    panier = get_panier()
    if not panier:
        return jsonify({"erreur": "Panier vide"}), 400

    line_items = []
    for item in panier:
        produit = MENU_INDEX.get(item["id"])
        if produit:
            line_items.append({
                "price_data": {
                    "currency": "eur",
                    "product_data": {
                        "name":        f"{produit['emoji']} {produit['nom']}",
                        "description": produit["description"]
                    },
                    "unit_amount": int(produit["prix"] * 100)
                },
                "quantity": item["quantite"],
            })
```

```python
    session["client_nom"]   = client_nom
    session["client_email"] = client_email
    # Sauvegarder le panier dans la session avant redirection Stripe
    session["panier_snapshot"] = json.dumps([
        {
            "nom":      MENU_INDEX[i["id"]]["nom"],
            "quantite": i["quantite"],
            "prix":     MENU_INDEX[i["id"]]["prix"]
        }
        for i in panier if i["id"] in MENU_INDEX
    ])

    try:
        checkout_session = stripe.checkout.Session.create(
            payment_method_types=["card"],
            line_items=line_items,
            mode="payment",
            success_url=request.host_url + "paiement/succes?session_id={CHECKOUT_SESSION_ID}",
            cancel_url=request.host_url  + "paiement/annule",
            customer_email=client_email,
            metadata={"client_nom": client_nom, "client_email": client_email}
        )
        return jsonify({"url": checkout_session.url})
    except Exception as e:
        return jsonify({"erreur": str(e)}), 500
```

6.1.4 Envoie de l’email et confirmation

Ce code sert à envoyer un email automatiquement depuis l’application.

La fonction reçoit l’adresse du destinataire, le sujet et le contenu du message. Ensuite, elle crée un email avec tout ça.

Le message est en format HTML, donc ça permet d’avoir un email plus propre et mieux présenté.

Après, le programme se connecte à Gmail grâce au serveur SMTP pour pouvoir envoyer le message avec une adresse et un mot de passe.

Si tout se passe bien, la fonction renvoie "True". Sinon, elle affiche une erreur et renvoie "False".

Ce système peut être utilisé par exemple pour envoyer un mail de confirmation après un paiement ou une commande.

```
def envoyer_email(destinataire, sujet, corps_html):
    try:
        msg = MIMEMultipart("alternative")
        msg["Subject"] = sujet
        msg["From"]    = EMAIL_SENDER
        msg["To"]      = destinataire
        msg.attach(MIMEText(corps_html, "html"))
        with smtplib.SMTP_SSL("smtp.gmail.com", 465) as smtp:
            smtp.login(EMAIL_SENDER, EMAIL_PASSWORD)
            smtp.sendmail(EMAIL_SENDER, destinataire, msg.as_string())
        return True
    except Exception as e:
        print(f"Erreur email : {e}")
        return False
```

7) Conclusion

Pour conclure, ce projet m'a vraiment permis de créer quelque chose de concret.

L'idée de SandwichSchool était surtout d'aider les élèves à gagner du temps et de mettre fin aux files interminables à la cafétéria.

Tout au long du développement, j'ai énormément appris, que ce soit pour coder, organiser mon travail ou utiliser les bons outils.

Ça n'a pas toujours été simple : j'ai eu quelques moments de blocage où rien ne fonctionnait comme prévu, mais c'est justement ça qui m'a fait progresser.

Même si l'application est fonctionnelle, je sais qu'elle peut encore évoluer, par exemple en ajoutant de nouvelles options ou en rendant son interface encore plus fluide.

Au final, ce projet m'a montré ce que c'est de gérer un projet de A à Z, et c'est une expérience qui me sera très utile pour la suite de mes études.

8) Bibliographie

Pallets Projects. (s.d.). *Flask – Un micro framework web léger*. Consulté le 10 novembre 2024, de https://flask.palletsprojects.com/

Python Software Foundation. (s.d.). *Documentation Python 3.12.5*. Consulté le 10 novembre 2025, de https://docs.python.org/3/

SQLite Development Team. (s.d.). *Format de fichier SQLite*. Consulté le 10 novembre , de https://sqlite.org/

Stripe. (s.d.). *Tableau de bord*. Consulté le 25 avril 2026, de https://dashboard.stripe.com/acct_1TM0QoH7Tce7VxUA/test/dashboard

Cloudflare. (s.d.). *Connecter des réseaux – Documentation Cloudflare One*. Consulté le 1 mai 2026, de https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/

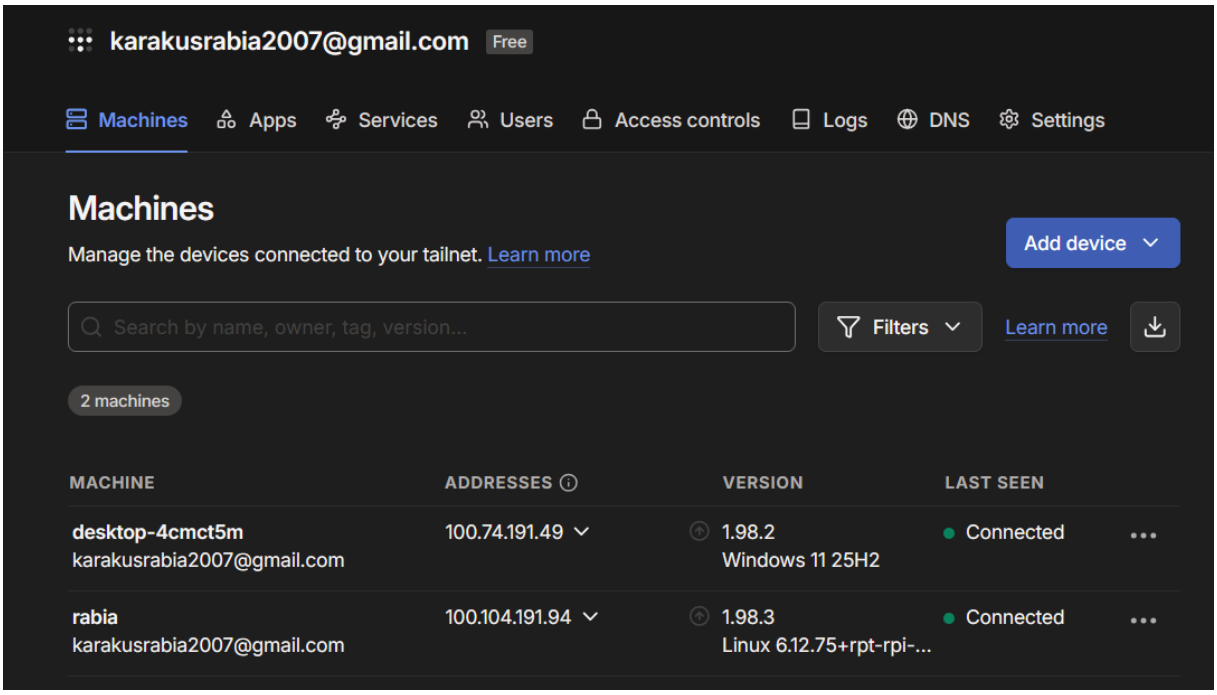
Tailscale. (s.d.). *Documentation Tailscale*. Consulté le 1 mai 2026, de https://tailscale.com/kb/

Raspberry Pi Foundation. (s.d.). *Documentation Raspberry Pi*. Consulté le 25 mai 2026, de https://www.raspberrypi.com/documentation/

belginux.com. (s.d.). *Créer un partage Samba*. Consulté le 27 mai 2026, de https://belginux.com/creer-un-partage-samba/

9) Annexes

Les machines présentes dans mon tunnel VPN.



Voici mon lien github avec tous les fichier source pour la réalisation de mon projet de fin d'étude

https://github.com/rabia-boop/SandwichSchool