Institut National de Radioélectricité et de Cinématographie

Enseignement technique secondaire de qualification

Accès aux études supérieures

Avenue Jupiter, 188

1190 Forest

Automatisation de terrain de FitFive

Projet personnel de Ikalaba Bebless

Pour l'obtention du certificat de qualification

Technicien en informatique

Année scolaire : 2025-2026

Remerciements

Avant de commencer ce rapport, je voudrais remercier les personnes qui m'ont soutenu et aidé tout au long de ce projet.

Tout d'abord, je tiens à exprimer ma reconnaissance à mon professeur de projet, Monsieur Bensallam. Ses conseils m'ont été d'une grande aide pour ne pas m'écarter de mon sujet et rester sur la bonne voie. Il m'a également beaucoup aidé pour la mise en place de mes bases de données.

Je souhaite ensuite remercier tout particulièrement **Monsieur Janah**, mon professeur de SML. Il m'a énormément aidé et conseillé dès le début de mon travail. Il a même eu la gentillesse de me prêter du matériel pour que je puisse réaliser ma maquette dans les meilleures conditions.

J'aimerais aussi remercier mes autres professeurs d'informatique, notamment Monsieur Mdiker et Monsieur Kajjoua, qui m'ont donné pas mal de conseils pour améliorer certains détails de mon projet.

Je remercie également mes parents, qui ont toujours été là pour me soutenir et m'encourager durant cette année.

Enfin, je tiens à remercier le centre de Fit Five. C'est en allant là-bas que j'ai trouvé toute l'inspiration nécessaire pour réaliser ce projet d'automatisation.

Table des matières

Introduction

Passionné par le football et l'informatique, je fréquente régulièrement des centres Fit Five. C'est en jouant là-bas que j'ai remarqué que la gestion des terrains était souvent peu optimisée, car elle repose entièrement sur la présence physique du personnel. C'est cette observation qui m'a donné l'idée de mon projet de Travail de Fin d'Études.

Actuellement, les complexes sportifs font face à plusieurs problèmes concrets : le personnel doit être présent en permanence pour encaisser les paiements et donner les accès aux terrains, ce qui représente un coût important en salaires. De plus, sans surveillance humaine constante, il est difficile de vérifier si les joueurs présents sur un terrain ont bien réservé leur créneau, ce qui peut entraîner des fraudes ou des occupations non autorisées. Même si certains centres proposent déjà une réservation en ligne ou par téléphone, il n'existe pas encore de système qui gère automatiquement l'accès physique au terrain sans intervention du personnel.

L'objectif de mon TFE est donc de concevoir une borne intelligente capable d'automatiser complètement la gestion d'un complexe sportif, pour le rendre totalement autonome et sécurisé. Pour répondre à ces besoins, j'ai développé une application en Python avec Tkinter qui permet au joueur de s'inscrire, de choisir son type de match (6, 8 ou 10 joueurs), de sélectionner un terrain et un horaire disponible, et de payer en ligne de manière sécurisée via l'API Stripe. L'application est connectée à une base de données MySQL hébergée sur AlwaysData, accessible depuis n'importe où via internet.

Une fois la réservation effectuée, le système génère automatiquement un code PIN unique à 4 chiffres, envoyé par email au joueur avec un reçu PDF. Le jour du match, le joueur tape ce code sur un keypad connecté à un Raspberry Pi 4 à l'entrée du terrain. Si le code est valide, un servo-moteur déverrouille la borne d'accès, des LEDs confirment visuellement l'accès et une caméra enregistre la séquence pour permettre à l'administrateur de vérifier à distance qu'il y a bien une activité sur le terrain. Quelques minutes avant la fin du créneau, un signal lumineux prévient les joueurs pour garantir que le terrain soit libéré à l'heure pour le groupe suivant.

1. Analyse des besoins

1.1 Description du problème

Le centre Fit Five Brussels propose des terrains de football synthétique à louer pour des matchs entre amis. Le problème principal que j'ai identifié, c'est que même si la réservation en ligne existe déjà, l'accès physique au terrain dépend encore entièrement du personnel sur place. Un employé doit être présent pour vérifier que le joueur a bien payé et pour lui ouvrir l'accès. Cela représente un coût en personnel et crée des risques : si personne n'est là pour surveiller, n'importe qui peut entrer sur un terrain sans avoir réservé.

Mon projet répond à ce problème en automatisant complètement le processus, de la réservation jusqu'à l'accès au terrain, sans qu'aucun employé ne soit nécessaire.

1.2 Public cible

Mon système s'adresse à deux types d'utilisateurs : le joueur et l'administrateur.

Le joueur est la personne qui souhaite réserver un terrain pour jouer un match. Pour cela, il doit d'abord créer un compte sur l'application, puis choisir le terrain souhaité, l'horaire ainsi que le nombre de joueurs. Une fois la réservation confirmée et le paiement effectué en ligne, il reçoit automatiquement un code d'accès par email. Ce code lui permettra, le jour de son match, d'accéder au terrain de manière simple et sécurisée.

L'administrateur, quant à lui, est le gestionnaire du centre sportif. Il dispose d'un accès à une page d'administration protégée par un mot de passe afin de garantir la sécurité des données. Depuis cette interface, il peut consulter toutes les réservations effectuées durant le mois, suivre le chiffre d'affaires généré par les réservations et vérifier quels codes d'accès ont déjà été utilisés. Cela lui permet de gérer plus facilement l'ensemble du système.

1.3 Cahier des charges

1.3.1 Matériels utilisés

Équipement	Description	Quantité	Prix
Raspberry Pi 4	Modèle B 4GB RAM.	1	170,00 €
Keypad 4x4	Matrice de 16 boutons poussoirs.	1	5,50 €
Servo-moteur	Modèle SG90 - 9g.	1	8,00 €
Capteur Ultrason	Module HC-SR04.	1	3,50 €
Bande LEDs	Ruban LED adressable RGB (1m).	2	6,10 €
Caméra USB	Module OV3660 5MP.	1	18,00 €
Écran LCD	Module 1602 avec interface I2C.	1	7,50 €

Le prix total de mes matériaux m'est revenu à 218,60 € pour pouvoir réaliser mon travail de fin d'étude

1.3.2 Logiciels utilisés

Application	Description	Prix
Visual Studio Code	Éditeur de code utilisé pour développer l'application Tkinter	0,00 €
MobaXterm	Terminal SSH pour se connecter au Raspberry Pi à distance	0,00 €
phpMyAdmin (AlwaysData)	Interface graphique pour gérer la base de données MySQL en ligne	0,00 €

1.3.3 le Raspberry pi 4

Le Raspberry Pi, c'est quoi ?

Le Raspberry Pi est un nano-ordinateur, c'est-à-dire un ordinateur de petite taille présenté sous la forme d'une carte électronique. Il est principalement utilisé dans des projets de programmation, d'électronique et d'automatisation. Malgré sa petite taille, il possède les composants essentiels d'un ordinateur classique, comme un processeur (CPU), une mémoire RAM et une carte microSD qui contient le système d'exploitation.

Il dispose de différentes connectivités comme 2 ports USB 2.0, 2 ports USB 3.0, un port Ethernet RJ45, un port jack, un port CSI pour la caméra, un port DSI pour l'affichage, ainsi que le Wi-Fi et le Bluetooth intégrés.

Le Raspberry Pi possède également des pins GPIO (General Purpose Input/Output) qui permettent de connecter et de contrôler des composants électroniques comme des capteurs, des LED ou des moteurs. Il dispose aussi de broches GND (masse) ainsi que d'alimentations en 3.3V et 5V, ce qui le rend très adapté aux projets d'automatisation et d'objets connectés.

Implantation dans mon projet : le Raspberry Pi est le cerveau du système de contrôle d'accès. C'est lui qui gère le keypad, le servo-moteur, les LEDs, le capteur ultrasonique et la caméra. Il se connecte à la base de données MySQL via internet pour vérifier si le code tapé par le joueur est valide et démarre automatiquement au boot grâce au crontab.

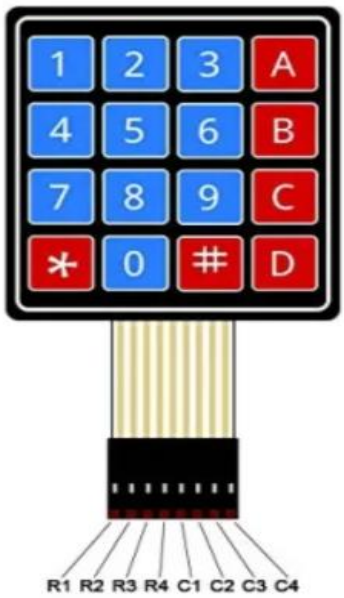
1.3.4 Le keypad 4x4

Le Keypad 4x4, c'est quoi ?

Le keypad 4x4 est un clavier matriciel composé de 16 touches, organisées en 4 lignes et 4 colonnes. Il est utilisé comme dispositif d'entrée pour permettre à un utilisateur d'introduire des données numériques ou des commandes dans un système électronique.

Son fonctionnement repose sur un système de matrice, où chaque touche correspond à l'intersection d'une ligne et d'une colonne. Lorsqu'une touche est pressée, une connexion électrique est créée entre ces deux éléments, ce qui permet au microcontrôleur ou au Raspberry Pi d'identifier la touche sélectionnée.

Ce type de clavier est apprécié dans les projets électroniques car il est compact, simple à connecter et nécessite peu de broches grâce à son fonctionnement matriciel.



Implantation dans mon projet : le joueur utilise le keypad pour taper son code d'accès à 4 chiffres. Il appuie sur # pour valider et sur * pour effacer en cas d'erreur. Le code saisi est ensuite vérifié en temps réel dans la base de données. Après 3 erreurs, le clavier se bloque et le joueur devra attendre 1 minute pour réessayer.

1.3.5 Le servo-moteur

Le servo-moteur, c’est quoi ?

Un servo-moteur est un moteur électrique qui peut effectuer un mouvement précis et contrôlé. Contrairement à un moteur classique qui tourne en continu, le servo-moteur peut se placer à un angle précis, souvent entre 0° et 180° selon le modèle.

Il est composé d’un moteur, d’engrenages et d’un capteur interne qui permet de connaître sa position et de la maintenir. Il fonctionne grâce à un signal PWM (Pulse Width Modulation) envoyé par un microcontrôleur ou un Raspberry Pi, ce qui permet de choisir exactement l’angle voulu.

Grâce à sa précision, le servo-moteur est très utilisé en robotique et en automatisation, quand il faut contrôler un mouvement de manière précise.

Implantation dans mon projet : Dès que le joueur saisit le code correct sur le keypad, le servo-moteur passe instantanément de 0° à 180° pour ouvrir la porte et permettre l'entrée sur le terrain. Une fois que les joueurs sont installés, la porte revient à 0° pour rester fermée pendant toute la durée de la rencontre (30, 45 ou 60 minutes). Enfin, dès que le match est terminé, le servo-moteur s'active à nouveau de 0° à 180° pour ouvrir la porte et permettre aux joueurs de quitter le terrain

1.3.6 Le capteur à ultrason

Le capteur à ultrason, c’est quoi ?

Le capteur à ultrasons HC-SR04 est un capteur de distance qui permet de mesurer la distance entre lui et un objet sans contact physique.

Il fonctionne en émettant des ondes ultrasonores, c’est-à-dire des sons à très haute fréquence, inaudibles pour l’oreille humaine. Ces ondes sont envoyées vers un obstacle, puis elles rebondissent et reviennent vers le capteur sous forme d’écho.

Le capteur mesure le temps entre l’émission du signal et la réception de l’écho. Grâce à la vitesse du son dans l’air, il calcule ensuite la distance qui le sépare de l’objet. Le résultat est généralement donné en centimètres.

Ce type de capteur est très utilisé en robotique et en automatisation pour détecter des obstacles ou mesurer des distances avec précision.

Implantation dans mon projet : L'activation du système repose sur un **capteur à** ultrasons qui agit comme un détecteur de présence intelligent. Le capteur mesure la distance en continu en envoyant des ondes sonores.

Dès qu'un joueur s'approche à moins de 50 cm, le capteur détecte l'obstacle et réveille instantanément le système qui demande alors la saisie du code sur le keypad.

1.3.7 Les Leds

Les Leds, c’est quoi ?

Une LED (Light Emitting Diode) est un composant électronique qui produit de la lumière quand un courant électrique passe à travers elle dans le bon sens.

Elle fonctionne parce que l’électricité est transformée en lumière à l’intérieur de la LED. Elle ne laisse passer le courant que dans un seul sens, donc il faut bien la brancher correctement (avec un côté positif et un côté négatif).

Les LED sont très utilisées en électronique parce qu’elles consomment peu d’énergie, durent longtemps et s’allument très vite. Elles existent en plusieurs couleurs comme rouge, jaune ou verte selon les matériaux utilisés.

Implantation dans mon projet : Les LEDs s’allument uniquement lorsque le joueur a entré le bon mot de passe. Elles servent ensuite de signal visuel et simulent l’éclairage du terrain. Elles restent allumées pendant toute la durée du match pour indiquer que le jeu est en cours.

À 2 minutes de la fin du match, seules les LEDs rouges s’allument pour prévenir les joueurs que la fin du temps est proche et qu’ils doivent se préparer à quitter le terrain.

Une fois le temps terminé, toutes les LEDs s’éteignent pour montrer que le match est fini.

1.3.8 L’écran oled I2C

L’écran oled, c’est quoi ?

L'OLED SSD1306 est un petit écran d'affichage très populaire en électronique. Le mot OLED signifie que ce sont les pixels (les petits points de l'écran) qui créent leur propre lumière. L'avantage, c'est qu'il n'y a pas de grosse lampe derrière l'écran : les zones noires sont donc totalement éteintes, ce qui permet de lire les messages très clairement, même dans le noir.

Le nom SSD1306 est simplement le modèle de la puce qui commande l'affichage. Enfin, il utilise le système I2C, une méthode de branchement très simple qui ne demande que deux fils pour envoyer toutes les informations (le texte, les dessins ou les chiffres).

Implantation dans mon projet : J'ai choisi d'utiliser deux écrans OLED identiques pour assurer une communication claire à chaque étape du parcours des joueurs :

Le premier écran est installé à l'entrée, juste au-dessus du keypad. Il sert d'interface de saisie : dès que le joueur tape son code, l'écran affiche des étoiles (****) pour confirmer la pression des touches tout en gardant le code secret. Il sert aussi à prévenir le joueur en cas d'erreur (mot de passe incorrect) ou de verrouillage du système pendant une minute (3 échecs : Système bloqué..).

Le deuxième écran est placé à l'intérieur, au niveau du terrain. Son rôle est d'informer les joueurs, il affiche en temps réel le chronomètre du match (1h, 45 ou 30 minutes) afin que les joueurs sachent exactement combien de temps il leur reste avant la fin du match tout simplement.

1.3.9 La caméra USB

La caméra USB, c’est quoi ?

La **caméra USB** est une webcam standard qui se branche directement sur l'un des ports USB du Raspberry Pi. Contrairement aux caméras spécifiques (comme le module caméra Pi) qui utilisent des nappes fragiles, celle-ci est très robuste et facile à installer. Une fois branchée, elle est automatiquement reconnue par le système Linux (le "cerveau" du Raspberry Pi) sous le nom de chemin **/dev/video0**, ce qui permet au programme de l'utiliser immédiatement pour capturer des images ou des vidéos.

Implantation dans mon projet : Dans mon projet, la caméra USB ne sert pas seulement à la sécurité, elle devient un outil de supervision en temps réel.

Son fonctionnement est directement lié au lancement de la session : dès que le bon code est saisi et que le match commence, la caméra s'active automatiquement pour filmer l'intégralité de la rencontre. Ce flux vidéo permet à l'administrateur de suivre le match à distance s'il le souhaite. Cela lui permet de vérifier en direct que tout se passe bien sur le terrain, de s'assurer du respect des règles et d'intervenir rapidement en cas de problème. La caméra s'arrête de filmer une fois le temps écoulé, garantissant ainsi un suivi précis de chaque réservation.

1.3.10 Les câbles GPIO

Les câbles GPIO, c’est quoi ?

Les câbles GPIO, aussi appelés câbles Dupont, sont de petits fils électriques utilisés en électronique pour relier facilement différents composants entre eux. Ils permettent de faire des connexions rapides et simples, sans avoir besoin de souder. Chaque câble possède des embouts mâles et/ou femelles qui s’adaptent aux broches des cartes comme le Raspberry Pi, ainsi qu’aux capteurs ou modules, ce qui facilite les montages sur une breadboard ou directement sur un circuit.

Implantation dans mon projet : Dans mon projet, les câbles GPIO permettent de relier les différents composants au Raspberry Pi. Ils servent à transmettre les données des capteurs vers le programme, à envoyer des commandes aux composants comme le servo-moteur ou les LEDs, mais aussi à fournir l’alimentation électrique nécessaire au bon fonctionnement de l’ensemble du système.

1.4 Logiciel utilisé

1.4.1 Visual studio code

Visual Studio Code, c’est quoi ?

Visual Studio Code est un programme gratuit créé par Microsoft qui sert à écrire du code informatique. On peut l’installer sur Windows, Linux et macOS.

Il permet de programmer dans plusieurs langages comme Python, JavaScript ou HTML. Mais également Il aide l’utilisateur en coloriant le code pour le rendre plus lisible, en proposant des suggestions pendant qu’on écrit et en indiquant les erreurs.

On peut aussi ajouter des extensions pour avoir plus de fonctionnalités selon ses besoins. C’est un outil très utilisé en informatique car il est simple, rapide et pratique pour coder.

Implantation dans mon projet : Dans mon projet, **Visual Studio Code** est l’environnement de développement (IDE) principal qui me permet d'écrire, organiser et structurer l’ensemble de mes scripts Python.

C’est dans ce logiciel que je développe mon interface graphique locale avec Tkinter pour gérer les fenêtres et l'interaction avec l'utilisateur. Ensuite, j'y programme la logique de mes composants matériels, comme la lecture du keypad, la mesure des distances avec le capteur ultrason HC-SR04 et le contrôle PWM du servo-moteur de la porte. Enfin, grâce à son terminal intégré, je peux exécuter mon programme en temps réel pour repérer et corriger rapidement les erreurs de syntaxe pendant le développement.

1.4.2 MobaXterm

Visual Studio Code, c’est quoi ?

MobaXterm est un logiciel qui permet de se connecter à distance à un autre appareil, comme un serveur ou un ordinateur, grâce à différents protocoles comme SSH.

Il offre un terminal intégré qui permet d’exécuter des commandes à distance et de gérer des fichiers plus facilement. Il est principalement utilisé en informatique et en réseau pour administrer des appareils sans devoir être physiquement devant eux.

C’est un outil très pratique car il regroupe plusieurs fonctions dans un seul logiciel et facilite le travail à distance.

Implantation dans mon projet : Dans mon projet, MobaXterm est l'outil de connexion à distance qui me permet de piloter et de gérer le Raspberry Pi directement depuis mon ordinateur de bureau. Comme mon nano-ordinateur fonctionne sans écran ni clavier dédiés, MobaXterm me sert de passerelle principale via le protocole SSH.

Grâce à son terminal en ligne de commande, je peux exécuter à distance mes scripts Python, interagir avec le système Linux et gérer directement ma base de données MySQL locale pour vérifier le stockage des utilisateurs ou des matchs. L'outil m'est également indispensable pour le transfert de fichiers en SFTP, ce qui me permet de glisser-déposer facilement mes codes sources de mon PC vers le Pi. Enfin, MobaXterm intègre un serveur X11 qui récupère le flux visuel du Raspberry Pi, ce qui est essentiel pour afficher et tester mon interface graphique Tkinter directement sur l'écran de mon ordinateur.

1.4.3 phpMyAdmin

PhpMyAdmin, c'est quoi ?

PhpMyAdmin est une interface graphique qui permet de gérer facilement une base de données MySQL ou MariaDB depuis un navigateur web, sans devoir utiliser uniquement des lignes de commande.

Grâce à cet outil, il est possible de créer des bases de données, ajouter ou supprimer des tables, modifier des données et exécuter des requêtes SQL de manière plus simple et plus rapide.

C'est un logiciel très utilisé en informatique, car il facilite la gestion des bases de données grâce à une interface claire et accessible.

Implantation dans mon projet : Dans mon projet, phpMyAdmin hébergé sur la plateforme Alwaysdata me sert d'interface de gestion pour ma base de données MySQL en ligne. C'est à cet endroit que j'ai structuré et créé mes tables pour stocker toutes les informations essentielles du système, comme les profils des utilisateurs, leurs codes d'accès au Keypad et le registre des réservations de terrains.

L'utilisation de cette plateforme en ligne me permet de centraliser les données de manière sécurisée et accessible. Via l'interface web de phpMyAdmin, je peux directement vérifier en temps réel les données enregistrées par mes scripts, insérer manuellement des utilisateurs pour faire des tests, ou exécuter des requêtes SQL sans devoir écrire de lignes de code Python. Cela m'offre un contrôle total et visuel sur l'état de ma base de données pendant toute la phase de développement.

1.5 Python

1.5.1 Python, c’est quoi ?

Python est un langage de programmation polyvalent, ce qui veut dire qu'on peut l'utiliser pour faire des choses très variées : créer des applications avec une interface graphique, développer des sites web, analyser des données, faire de l'intelligence artificielle ou encore contrôler des composants électroniques. Ce qui le rend populaire c'est sa syntaxe simple et lisible qui ressemble presque à du langage naturel, ce qui le rend accessible même pour les débutants. Python dispose également d'une très grande bibliothèque de modules prêts à l'emploi et fonctionne sur Windows, Linux et MacOS.

1.5.2 Pourquoi Python ?

Tout d’abord j’ai choisi le langage python pour plusieurs raisons : la première c'est que c'est le langage que je maîtrise le mieux, ce qui m'a permis de me concentrer sur la logique de mon projet plutôt que d'apprendre un nouveau langage en même temps.

La deuxième raison c'est que Python est parfaitement compatible avec toutes les bibliothèques dont j'avais besoin dans mon projet : Tkinter pour créer l'interface de réservation, RPi.GPIO pour contrôler le keypad, le servo-moteur, les LEDs et le capteur ultrasonique du Raspberry Pi, mysql-connector pour communiquer avec la base de données MySQL, ReportLab pour générer les reçus PDF, ou encore smtplib pour envoyer les emails automatiquement.

Le fait de pouvoir utiliser un seul et même langage pour les deux parties de mon projet, aussi bien sur le PC que sur le Raspberry Pi, m'a vraiment simplifié la tâche et m'a évité des problèmes de compatibilité entre différents langages.

2.6 Tkinter

2.6.1 Tkinter, c’est quoi ?

Tkinter est la bibliothèque graphique standard de Python. Elle permet de créer des interfaces graphiques avec des fenêtres, des boutons, des champs de texte, des listes déroulantes, des calendriers et bien d'autres éléments visuels. Ce qui la rend pratique c'est qu'elle est directement intégrée dans Python, ce qui veut dire qu'on n'a pas besoin de l'installer séparément. Tkinter fonctionne sur Windows, Linux et MacOS. Elle est principalement utilisée pour créer des applications de bureau, c'est-à-dire des applications qui tournent directement sur un ordinateur avec une interface que l'utilisateur peut voir et utiliser.

2.6.2 Pourquoi tkinter ?

J'ai choisi Tkinter pour créer l'interface graphique de ma borne car c'est une bibliothèque Python que je connaissais déjà et avec laquelle j'étais très à l'aise. Comme elle est déjà intégrée à Python, je n'ai pas eu besoin d'installer d'autres outils et j'ai pu tout coder au même endroit.

Avec Tkinter, j'ai réussi à concevoir une application complète qui utilise plusieurs fenêtres. J'ai pu y intégrer des choses simples et interactives : un calendrier pour choisir la date de réservation, des boutons colorés pour voir les heures disponibles, des formulaires pour l'inscription et la connexion des joueurs, une page de paiement et une page pour l'administrateur. C'était le choix idéal pour que les utilisateurs puissent naviguer facilement sur la borne tout en restant connecté à mes composants matériels.

2.7 FFmpeg

C'est quoi ? FFmpeg est un outil open source très puissant qui permet de traite des fichiers audio et vidéo. Il peut enregistrer, convertir et diffuser des vidéos dans pratiquement tous les formats existants. Il fonctionne en ligne de commande et est disponible sur Windows, Linux et MacOS."

Pourquoi ce choix ? Au départ j'utilisais fswebcam pour capturer des images depuis la caméra, mais fswebcam ne permet de prendre que des photos et pas d'enregistrer une vraie vidéo. J'ai donc cherché une alternative et j'ai trouvé ffmpeg qui me permettait d'enregistrer directement une vidéo MP4 depuis la caméra USB branchée sur le Raspberry Pi avec une seule ligne de commande. C'est gratuit, fiable et parfaitement compatible avec Linux.

2.8 Fonctionnalités attendues

En plus des éléments obligatoires du cahier des charges, j'ai développé plusieurs fonctionnalités supplémentaires pour rendre mon application beaucoup plus complète et réaliste.

Tout d'abord, j'ai mis en place un système d'annulation de réservation qui envoie automatiquement un e-mail de confirmation au joueur. Pour diversifier les offres de la borne, j'ai également intégré des forfaits événementiels, comme un "Pack Anniversaire" et un "Team Building", avec leur propre système de sélection de date et d'heure.

Ensuite, pour améliorer l'expérience des utilisateurs, l'accueil affiche désormais un bandeau de bienvenue personnalisé avec le nom du joueur connecté et les détails de son prochain match. J'ai aussi codé une sécurité "anti-doublon" très importante : elle empêche un joueur de réserver deux terrains différents le même jour à la même heure. Enfin, j'ai créé une page d'administration complète qui regroupe les statistiques du mois, permettant de suivre facilement le nombre total de réservations, le chiffre d'affaires généré et l'historique des codes d'accès utilisés.

2.9 Contraintes

Tout au long de ce projet, j’ai dû faire face à plusieurs contraintes techniques et matérielles que j'ai réussi à résoudre.

Premièrement, concernant la base de données en ligne, j'avais besoin d'un système accessible depuis Internet pour que l'application Tkinter sur mon PC et le Raspberry Pi puissent communiquer entre eux. La plateforme AlwaysData restreignait les

connexions distantes par adresse IP, ce qui a bloqué mes accès au départ. J'ai donc dû configurer correctement les autorisations d'accès pour régler ce problème.

Deuxièmement, j'ai rencontré une contrainte avec les broches GPIO du Raspberry Pi. Au démarrage de la carte, certains pins se mettent par défaut sur un état électrique "HIGH" (allumé), ce qui déclenchait mes LEDs de façon non voulue. J'ai dû analyser la documentation du Pi pour identifier ces broches particulières et les remplacer par des pins neutres au démarrage.

Troisièmement, au niveau matériel, l'outil de caméra fswebcam que j'utilisais au début ne permettait pas de capturer de vraies vidéos, mais seulement des photos successives. Pour obtenir des fichiers vidéo au format MP4 fonctionnels, j'ai dû remplacer cet outil par ffmpeg.

Enfin, j'ai eu une contrainte liée au démarrage automatique de la borne. Le script Python devait se lancer tout seul à chaque allumage du Raspberry Pi sans aucune intervention manuelle. J'ai utilisé le planificateur de tâches crontab pour automatiser cela, mais j'ai dû y intégrer un délai d'attente (un "sleep") suffisant pour s'assurer que le système et tous les composants soient bien initialisés avant que le code ne s'exécute.

3. Etude préalable

Avant même que je commencer à coder quoi que ce soit, j'ai d'abord bien pris le temps de faire des recherches pour m'assurer de faire les bons choix technologiques. Cette étape était importante pour ne pas me retrouver bloqué en cours de projet à cause d'un mauvais choix dès le départ.

3.1 Solutions existantes

Il existe déjà des solutions sur le marché pour gérer les réservations de terrains sportifs. On peut par exemple citer des applications comme Playtomic ou Koala, qui

permettent de réserver et de payer en ligne depuis un smartphone. Ces solutions sont bien développées et très utilisées dans les centres sportifs professionnels.

Cependant, le point commun de toutes ces solutions c'est qu'elles s'arrêtent à la réservation et au paiement en ligne. Aucune d'entre elles ne gère l'accès physique au terrain de manière automatique. Le joueur doit toujours passer par un employé ou un agent de sécurité pour accéder au terrain. C'est exactement ce manque que mon projet vient combler en ajoutant un système de contrôle d'accès physique automatisé directement lié à la réservation.

3.2 Comparaison des technologies

3.2.1 Choix du microcontrôleur

Pour la partie contrôle d'accès, j'avais le choix entre plusieurs microcontrôleurs. Les deux options les plus connues dans le monde des projets électroniques sont l'Arduino et le Raspberry Pi.

Critère	Arduino	Raspberry Pi 4
Système d'exploitation	Non (pas d'OS)	Oui (Linux)
Langage principal	C / C++	Python
Connexion internet	Non (sans shield)	Oui (Wi-Fi intégré)
Connexion base de données	Difficile	Facile via Python
Puissance de calcul	Faible	Élevée
Prix	~20 €	~60 €
Facilité d'utilisation	Moyenne	Élevée

J'ai choisi le Raspberry Pi car il tourne sous Linux, supporte Python nativement et dispose d'une connexion Wi-Fi intégrée. Ces trois points étaient essentiels pour mon projet puisque j'avais besoin de me connecter à une base de données en ligne et d'exécuter un script Python complexe.

3.2.2 Choix de la base de données

Pour stocker les réservations, les utilisateurs et les codes d'accès, j'ai comparé trois solutions différentes : SQLite, MySQL et Firebase.

Critère	SQLite	MySQL	Firebase
Accessibilité depuis internet	Non (local uniquement)	Oui	Oui
Langage de requête	SQL	SQL	NoSQL
Hébergement gratuit	Oui	Oui (AlwaysData)	Oui (limité)
Compatibilité Python	Oui	Oui	Oui
Facilité d'utilisation	Très facile	Facile	Moyenne
Adapté à plusieurs appareils	Non	Oui	Oui

C’est quoi une base de données ?

Une base de données est un ensemble de données organisées et stockées sur un ordinateur, un peu comme un grand tableau numérique. Par exemple, elle peut contenir une liste d’utilisateurs avec leur nom, leur mot de passe et leurs réservations. Ces informations peuvent ensuite être facilement recherchées, ajoutées, modifiées ou supprimées.

J'ai éliminé SQLite dès le départ car c'est une base de données locale — elle ne peut pas être accessible depuis plusieurs appareils via internet. Or dans mon projet, l'application Tkinter sur PC et le Raspberry Pi devaient tous les deux accéder à la même base de données. Firebase aurait pu fonctionner mais c'est une base de données NoSQL, ce qui veut dire qu'elle ne fonctionne pas avec des requêtes SQL classiques et aurait nécessité d'apprendre un nouveau système. J'ai donc choisi MySQL, hébergé gratuitement sur AlwaysData, qui m'offrait tout ce dont j'avais besoin : accessibilité depuis internet, requêtes SQL que je connaissais déjà et compatibilité parfaite avec Python.

3.3 Justification des choix technologiques

3.3.1 smtplib / Gmail SMTP

C'est quoi ? smtplib est une bibliothèque intégrée dans Python qui permet d'envoyer des emails via le protocole SMTP (Simple Mail Transfer Protocol). Le protocole SMTP c'est le protocole standard utilisé pour l'envoi d'emails sur internet. Dans mon projet, j'utilise smtplib avec le serveur SMTP de Gmail pour envoyer les emails automatiquement.

Pourquoi ce choix ? J'ai choisi smtplib avec Gmail car c'est une solution simple, gratuite et directement intégrée dans Python. Il suffit d'avoir un compte Gmail et de générer un mot de passe d'application pour pouvoir envoyer des emails depuis son code Python sans passer par un service externe payant.

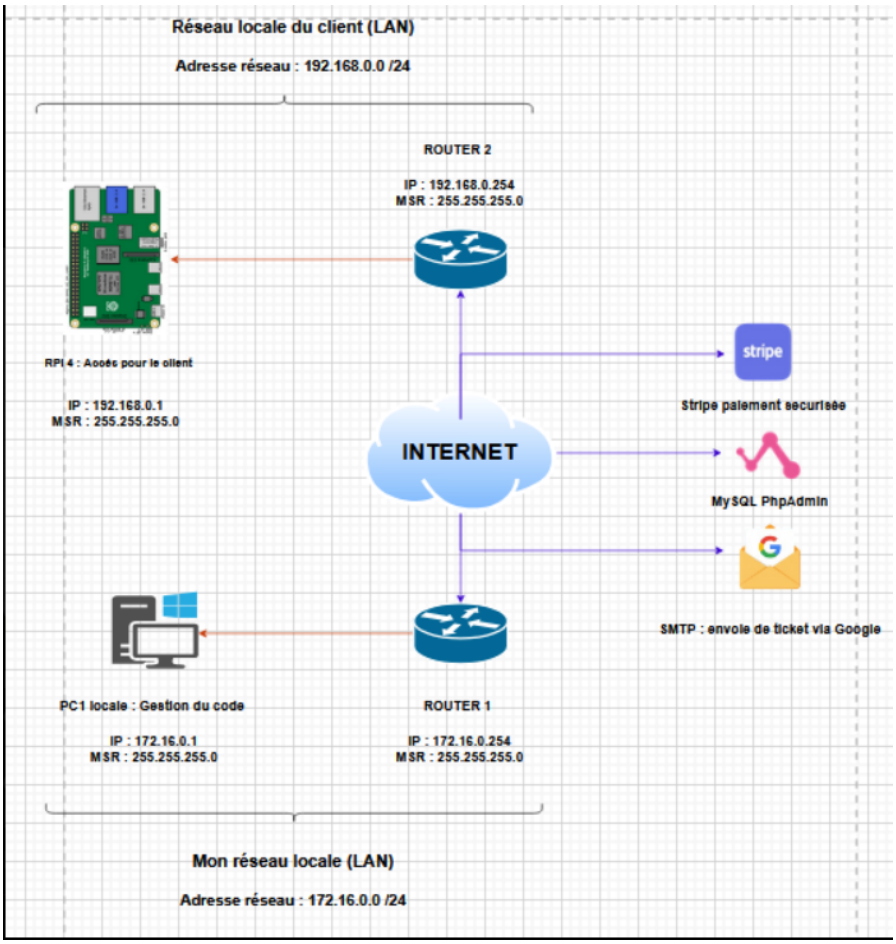
3.3.2 Stripe

C'est quoi ? Stripe est une plateforme de paiement en ligne qui propose une API permettant d'intégrer un système de paiement par carte bancaire dans une application. Elle est utilisée par des milliers d'entreprises dans le monde et est reconnue pour sa fiabilité et sa sécurité.

Pourquoi ce choix ? J'ai choisi Stripe car c'est une des API de paiement les plus utilisées et les mieux documentées. Elle propose un environnement de test qui m'a permis de tester les paiements sans utiliser une vraie carte bancaire, ce qui était pratique pendant le développement.

4. Conception du projet

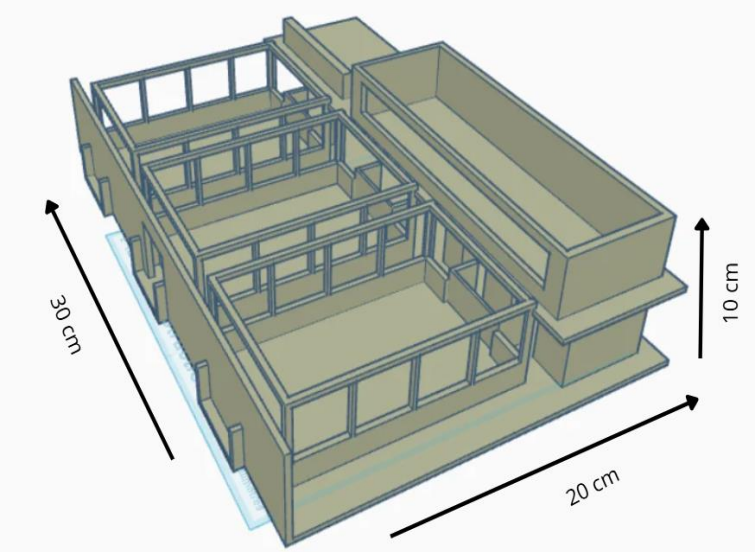
4.1 Topologie réseau



Explication de la topologie

Cette topologie présente deux réseaux locaux distincts, interconnectés via Internet et exploitant des services Cloud tiers. En haut, le réseau du client (192.168.0.0/24) héberge un Raspberry Pi 4 (192.168.0.1) connecté au Router 2, tandis qu'en bas, votre réseau de développement (172.16.0.0/24) comprend un poste de travail dédié à la gestion du code (172.16.0.1) derrière le Router 1. L'ensemble de l'infrastructure transite par Internet pour interagir avec une base de données MySQL (PhpMyAdmin), un système de paiement sécurisé Stripe et un serveur SMTP Google chargé de l'envoi automatisé de tickets.

4.2 Schéma mécanique



Vue en perspective :

Cette vue en perspective permet d'avoir une vision globale de la maquette. On peut clairement distinguer les 3 terrains de Fit Five ainsi que l'ensemble de la structure. Chaque terrain est séparé et dispose de son propre accès.

Vue arrière :

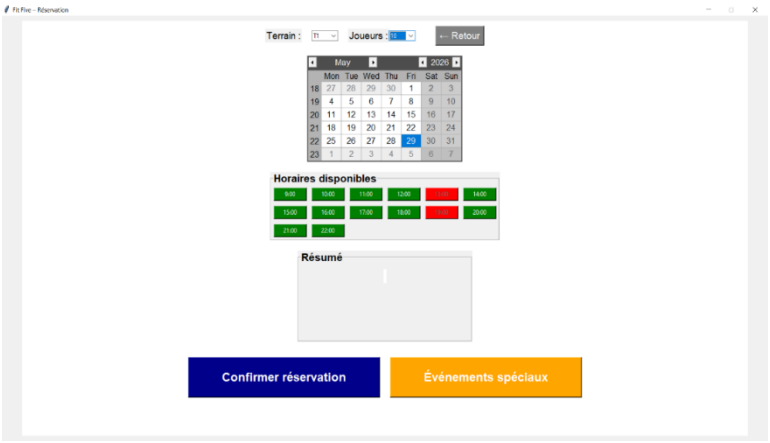
Cette vue arrière montre l'arrière du bâtiment. On peut y voir les fenêtres, les portes, les installations techniques ainsi que la profondeur de la construction.

5. Réalisation TFE

5.1. Explication du fonctionnement de l'application frontend

1)La page d'accueil:

L'application invite l'utilisateur à naviguer facilement grâce aux boutons **Réserver un terrain**, **Mes réservations** et **Quitter**, lui permettant d'accéder rapidement aux fonctionnalités principales de Fit Five.



2) La page de réservation :

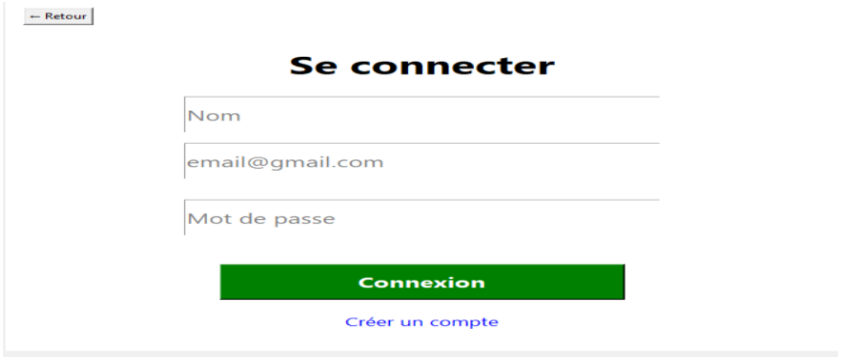
L'utilisateur sélectionne selon ses préférences le terrain souhaité, le nombre de joueurs, la date et l'horaire disponible afin de finaliser sa réservation au complexe Fit Five.

Les boutons rouges indiquent qu'un créneau est indisponible, soit parce qu'il est déjà réservé, soit parce que la date est passée. L'utilisateur ne peut donc pas le sélectionner, assurant ainsi une réservation fiable et sans erreur.

3) La page Forfaits Événementiels (propose deux offres spéciales) :

Le Pack Anniversaire à 180€ inclut 1h30 de match, un espace goûter de 30 minutes, un gâteau et des boissons, un cadeau surprise, le tout pour un maximum de 12 joueurs. **Le Team Building à 350€** inclut la privatisation de 2 terrains pendant 3h00, un tournoi organisé, un apéro et drink de fin, pour un maximum de 20 participants. Chaque forfait possède un bouton **« Réserver ce forfait »** qui redirige l'utilisateur vers un formulaire de réservation adapté à l'offre choisie. L'utilisateur peut également revenir à la page précédente grâce au bouton **"Retour"** en haut à gauche.

Après avoir cliqué sur "Réserver ce forfait", une fenêtre s'ouvre permettant à l'utilisateur de sélectionner une date et une heure de début avant de confirmer sa réservation.

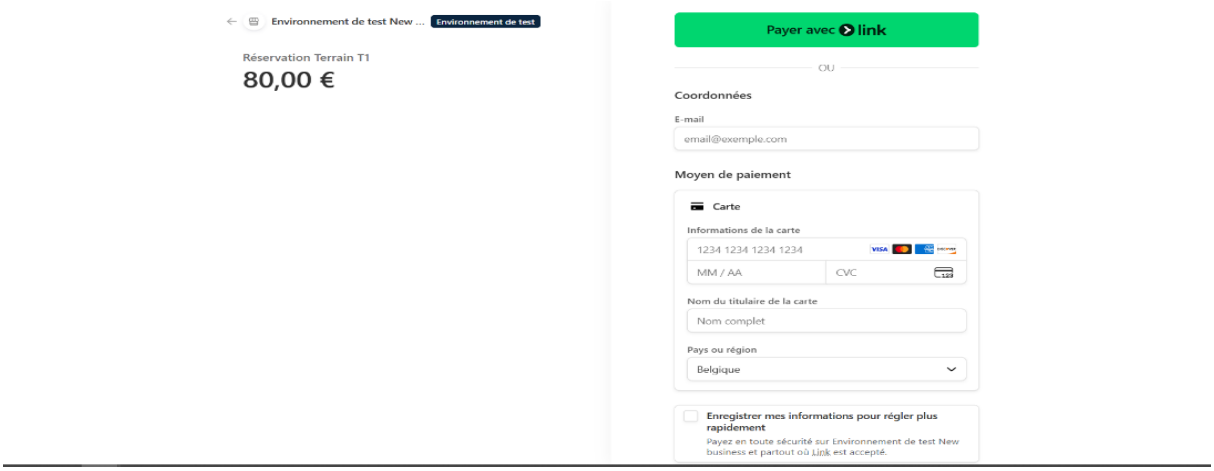


4) La page de connexion :

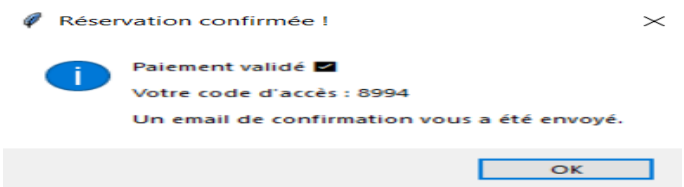
permet à l'utilisateur de s'identifier en entrant son nom, son email et son mot de passe. Le mot de passe doit respecter certains caractères spéciaux pour être valide. Une fois les informations saisies, l'utilisateur clique sur le bouton "Connexion" pour accéder à son compte. S'il n'a pas encore de compte, il peut cliquer sur "Créer un compte" pour s'inscrire.

5) Page de paiement sécurisé :

La page de paiement sécurisé affiche le total à payer et permet à l'utilisateur de procéder au paiement en cliquant sur le bouton "Procéder au paiement". Une fois cliqué, l'utilisateur est redirigé vers la page de paiement Stripe, où il peut choisir de payer avec son compte Link ou bien en entrant manuellement ses informations bancaires telles que son numéro de carte, sa date d'expiration, son code CVC ainsi que le nom du titulaire de la carte.

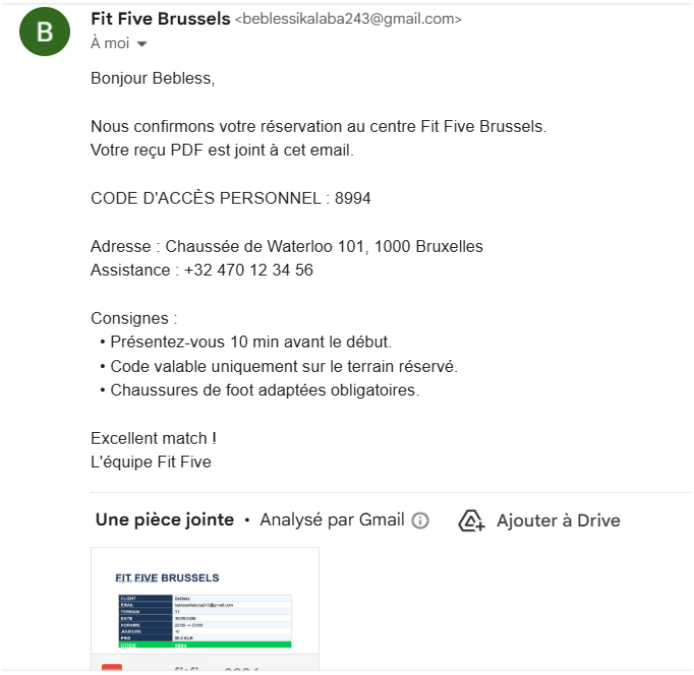


Une fois le paiement effectué avec succès, l'utilisateur est redirigé vers une page de confirmation affichant le message "Paiement Réussi !" accompagné d'une icône verte. Un message lui indique qu'il peut fermer cette page et retourner à l'application Fit Five.



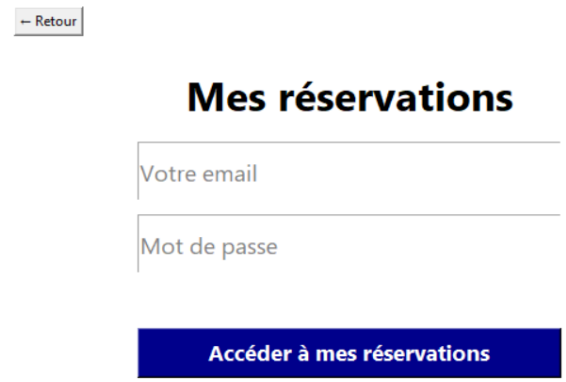
6) **Confirmation de réservation :**

Une fois le paiement validé, une fenêtre de confirmation s'affiche indiquant que la réservation est confirmée. L'utilisateur reçoit un code d'accès unique « (ici 8994) » qui lui permettra d'accéder au terrain réservé. Un email de confirmation lui est également envoyé automatiquement. L'utilisateur peut ensuite fermer cette fenêtre en cliquant sur "OK".



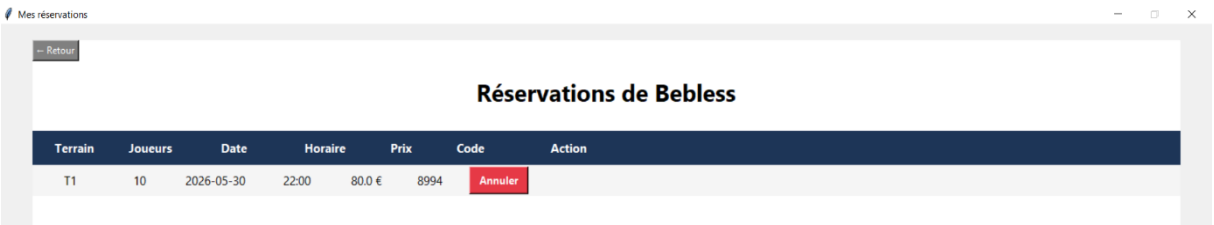
7) Email de confirmation :

Après la réservation, l'utilisateur reçoit automatiquement un email de confirmation envoyé par Fit Five Brussels, contenant son code d'accès personnel (ici 8994), l'adresse du centre, un numéro d'assistance ainsi que des consignes à respecter le jour du match. Un reçu PDF est également joint à cet email avec tous les détails de la réservation.



8) Mes réservations :

La page "Mes réservations" permet à l'utilisateur de consulter ses réservations en cours ou d'annuler sa réservation en se connectant avec son email et son mot de passe.



Une fois connecté, il peut voir l'ensemble des détails de sa réservation tels que le terrain, le nombre de joueurs, la date, l'horaire, le prix ainsi que son code d'accès. En cas d'imprévu, l'utilisateur a également la possibilité d'annuler sa réservation directement en cliquant sur le bouton "Annuler".

5.2 Interface d'administration

L'application dispose d'une interface d'administration dédiée, permettant au gérant de Fit Five Brussels de suivre en temps réel l'activité de son centre.

Notifications par email :

Dès qu'une nouvelle réservation est validée et payée par un client, l'administrateur reçoit automatiquement un email d'alerte contenant toutes les informations essentielles de la transaction, à savoir le nom du client, son adresse email, le terrain réservé, la date, l'horaire, le prix ainsi que le code d'accès généré.

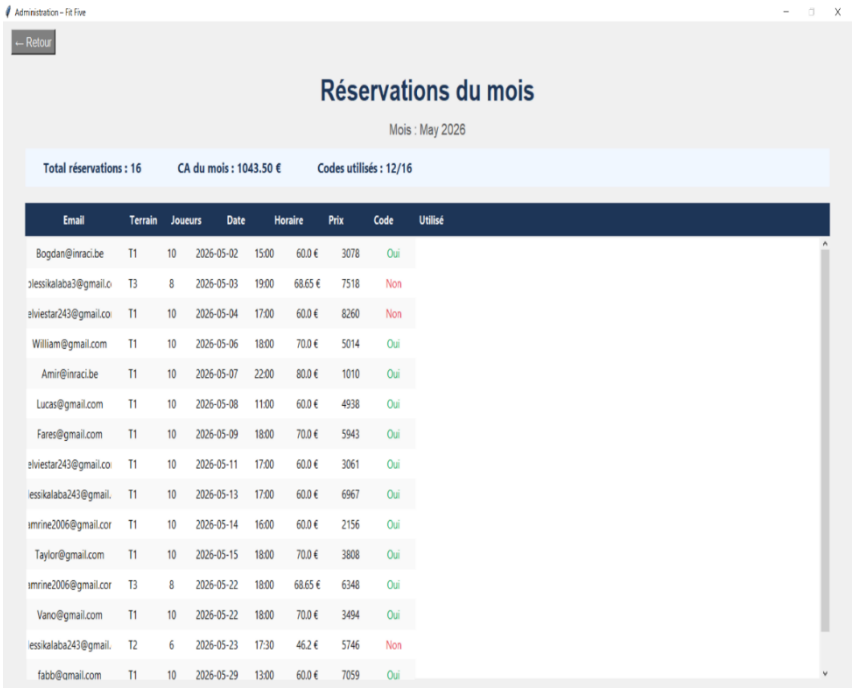


Tableau de bord des réservations :

La page "Réservations du mois" offre à l'administrateur une vue complète sur les réservations du mois en cours. Un résumé affiche le nombre total de réservations, le chiffre d'affaires ainsi que le nombre de codes utilisés. Le tableau liste chaque réservation en détail, avec une colonne "Utilisé" indiquant en vert **Oui** ou en rouge **Non** selon que le client s'est présenté ou non.

6. Fonctionnement interne de l'application (backend)

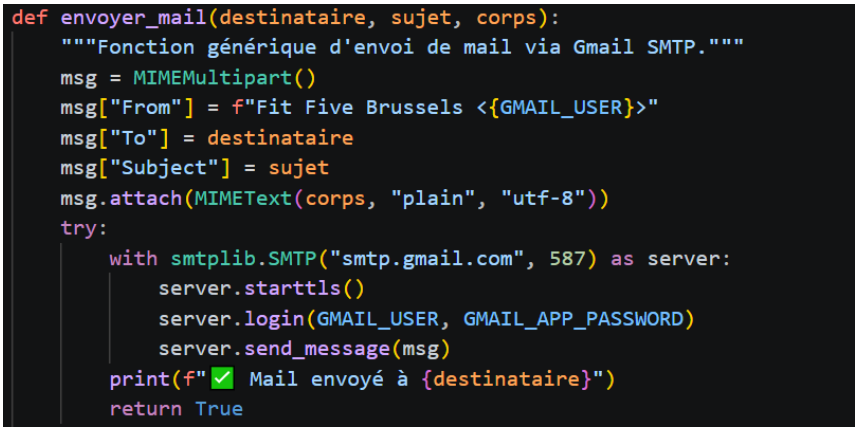
Le fonctionnement interne de l'application repose sur un ensemble de mécanismes techniques qui travaillent ensemble en arrière-plan, allant de la connexion à la base de données jusqu'à l'envoi automatique des mails de confirmation, en passant par le calcul des prix, la gestion des paiements et la sécurisation des accès utilisateurs.

6.1 Connexion Base de Données

```
# CONFIGURATION BASE DE DONNÉES
# ================================================
DB_CONFIG = {
    "host": "mysql-bebless.alwaysdata.net",
    "user": "bebless",
    "password": "Psaume73",
    "database": "bebless_tfe_foot",
    "auth_plugin": "mysql_native_password",
    "connection_timeout": 10
}
```

L'application se connecte à une base de données MySQL hébergée sur AlwaysData avec un délai maximum de 10 secondes. Une fonction get_cursor() gère la reconnexion automatique en cas de déconnexion. Cette base de données est le pilier central : sans elle, aucune fonctionnalité ne fonctionne (connexion, réservations, affichage des créneaux, administration).

6.2 Système d'envoi de mails



```
def envoyer_mail(destinataire, sujet, corps):
    """Fonction générique d'envoi de mail via Gmail SMTP."""
    msg = MIMEMultipart()
    msg["From"] = f"Fit Five Brussels <{GMAIL_USER}>"
    msg["To"] = destinataire
    msg["Subject"] = sujet
    msg.attach(MIMEText(corps, "plain", "utf-8"))
    try:
        with smtplib.SMTP("smtp.gmail.com", 587) as server:
            server.starttls()
            server.login(GMAIL_USER, GMAIL_APP_PASSWORD)
            server.send_message(msg)
        print(f"✅ Mail envoyé à {destinataire}")
        return True
```

La fonction envoyer_mail() gère l'envoi de tous les mails via Gmail SMTP. Elle crée un objet MIMEMultipart() et remplit les champs du message : l'expéditeur, le destinataire, le sujet et le contenu en UTF-8. Ensuite, elle établit une connexion sécurisée avec le serveur SMTP de Gmail sur le port 587 avec TLS. L'authentification se fait via GMAIL_USER et GMAIL_APP_PASSWORD.La fonction envoie le message, affiche une confirmation et retourne True en cas de succès.

6.3 Calcul du prix

```
361     def calcul_prix(horaire):
362         try:
363             h = int(horaire.split(":")[0])
364             supplement = 20 if h >= 20 else 10 if h >= 18 else 0
365             terrain = combo_terrain.get()
366             joueurs = int(combo_joueurs.get())
367             coef_t = TERRAIN_COEF.get(terrain, 1.0)
368             coef_j = JOUEURS_COEF.get(joueurs, 1.0)
369             return round(PRIX_BASE * coef_t * coef_j + supplement, 2)
370         except Exception as e:
371             print(f"Erreur calcul_prix : {e}")
372             return PRIX_BASE
373
```

La fonction calcul_prix() détermine le prix de la réservation. Elle extrait d'abord l'heure de l'horaire sélectionné, puis ajoute un supplément : 20€ après 20h, 10€ après 18h, sinon 0€. Ensuite, elle récupère le terrain et le nombre de joueurs choisis. Elle applique un coefficient au prix de base selon le terrain (T1 : 1.0, T2 : 1.1, T3 : 1.15) et selon le nombre de joueurs (10 : 1.0, 8 : 0.85, 6 : 0.7). Finalement, elle calcule : **Prix = 60€ × coefficient_terrain × coefficient_joueurs + supplément**.

6.4 Annulation de réservation

```
1138 def confirmer_annulation(row, email, nom, parent):
1139     """Popup de confirmation avant annulation."""
1140     rid, terrain, joueurs, date_r, horaire, prix, code = row
1141     date_aff = date_r.strftime("%d/%m/%Y") if hasattr(date_r, 'strftime') else str(date_r)
1142
1143     rep = messagebox.askyesno(
1144         "Confirmer l'annulation",
1145         f"Voulez-vous vraiment annuler :\n\n"
1146         f"Terrain : {terrain}\n"
1147         f"Date    : {date_aff} à {horaire}\n"
1148         f"Prix    : {prix} €\n\n"
1149         f"Cette action est irréversible."
1150     )
1151     if not rep:
1152         return
1153
1154     cur = get_cursor()
1155     if cur is None:
1156         messagebox.showerror("Erreur", "DB non connectée.")
1157         return
1158
1159     try:
1160         cur.execute("DELETE FROM reservations WHERE id=%s", (rid,))
1161         db.commit()
```

La fonction confirmer_annulation() affiche d'abord une boîte de dialogue demandant au client de confirmer son annulation, en lui rappelant le terrain, la date, l'horaire et le prix de sa réservation. Si le client clique sur "Non", la fonction s'arrête sans rien faire. Si il confirme, la fonction vérifie que la base de données est bien connectée. Ensuite, une requête SQL DELETE est exécutée pour supprimer définitivement la réservation correspondante grâce à son identifiant unique, puis la suppression est validée avec db.commit().

6.5 Sécurisation de l'accès administrateur

```
# ============================================================================
# MODULE PAGE ADMIN
# ============================================================================
def ouvrir_admin():
    """Page admin protégée par mot de passe."""
    pwd = tk.simpledialog.askstring(
        "Accès Admin", "Mot de passe admin :", show="*"
    )
    if pwd != "AdminFitFive2026":
        if pwd is not None:
            messagebox.showerror("Refusé", "Mot de passe incorrect.")
        return
```

La fonction ouvrir_admin() protège l'accès à la page d'administration par un mot de passe. Lorsqu'un utilisateur clique sur le bouton "Administration", une boîte de dialogue s'ouvre et lui demande de saisir un mot de passe (masqué par des étoiles). Si le mot de passe entré ne correspond pas au mot de passe attendu (AdminFitFive2026), un message d'erreur s'affiche et l'accès est refusé. Seul l'administrateur connaissant le bon mot de passe peut accéder à cette page.

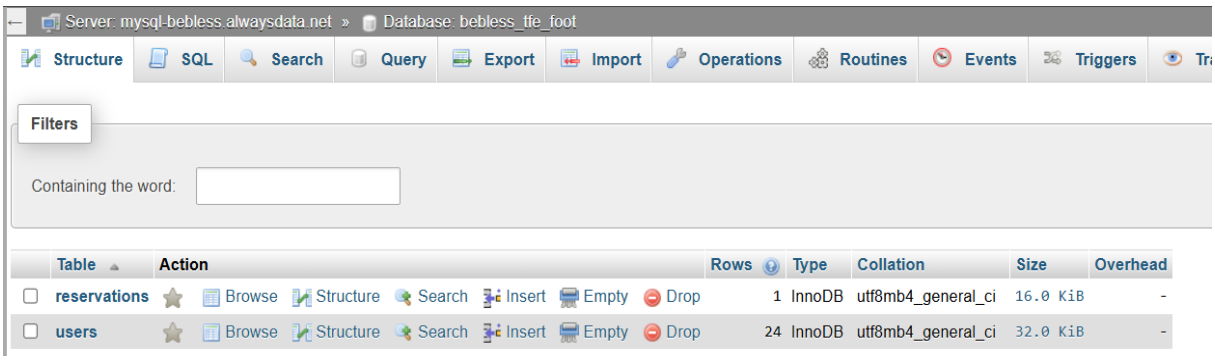
6.6 Gestion des créneaux horaires selon le nombre de joueurs

```
DUREE_JOUEURS = {10: 60, 8: 45, 6: 30}

def calcul_heure_fin(horaire, joueurs):
    try:
        h, m = map(int, horaire.split(":"))
        duree = DUREE_JOUEURS.get(int(joueurs), 60)
        total_min = h * 60 + m + duree
        return f"{total_min // 60:02d}:{total_min % 60:02d}"
    except Exception:
        return "??"

btn_actif = [None]

def select_horaire(h, btn):
    nonlocal selected_horaire
    if btn_actif[0] and btn_actif[0] != btn:
        try:
            btn_actif[0].config(bg="green")
        except Exception:
            pass
    selected_horaire = h
    btn.config(bg="#005500")
    btn_actif[0] = btn
    maj_resume()
```

Le dictionnaire DUREE_JOUEURS définit la durée d'une partie selon le nombre de joueurs. La fonction calcul_heure_fin() calcule automatiquement l'heure de fin de la réservation. La fonction select_horaire() permet à l'utilisateur de choisir un horaire, met en évidence le bouton sélectionné et met à jour le récapitulatif de la réservation.

7. Base de données

Pour ce projet, j'ai créé une base de données MySQL nommée « bebles_tfe_foot ». Elle contient 2 tables : users et reservations



7.1 Table users

id	nom	email	mdp
39	Bebless	beblessikalaba243@gmail.com	Inraci#20321

Cette table contient les informations des utilisateurs inscrits sur le site. Chaque utilisateur possède un identifiant unique (id), un nom, une adresse e-mail utilisée pour se connecter ainsi qu'un mot de passe (mdp) permettant de sécuriser l'accès à son compte.

7.2 Table reservation

id	email	terrain	joueurs	date_resa	horaire	prix	code	code_utilise
64	beblessikalaba243@gmail.com	T1	10	2026-05-30	22:00	80	8994	0

Cette table enregistre toutes les réservations effectuées par les utilisateurs sur le site. Pour chaque réservation, elle stocke un identifiant unique, l'adresse e-mail de l'utilisateur ayant effectué la réservation, le terrain choisi, le nombre de joueurs, la date et l'horaire sélectionnés. Elle contient également le prix calculé pour la réservation, un code de confirmation généré automatiquement ainsi qu'une information indiquant si ce code a déjà été utilisé ou non.

8. Conclusion

Ce Travail de Fin d'Études m'a permis de créer une borne intelligente pour gérer et contrôler l'accès à un complexe de football appelé **Fit Five**. Le projet se divise en deux grandes parties : une application développée en Python qui permet de faire des réservations en ligne, de payer via Stripe et de recevoir un email de confirmation automatique, et une partie hardware basée sur un Raspberry Pi 4 qui contrôle plusieurs composants électroniques comme le keypad, le servo-moteur, les LEDs, le capteur à ultrasons et l'écran OLED. J'ai aussi construit une maquette physique pour montrer concrètement comment le système fonctionne.

Les objectifs de départ ont bien été atteints. L'utilisateur peut s'inscrire, se connecter, réserver un terrain, choisir un horaire, payer en ligne et recevoir un code PIN unique par email. Ce code lui permet ensuite d'accéder au terrain en le tapant sur le keypad connecté au Raspberry Pi. Le système vérifie aussi que les créneaux ne se chevauchent pas et propose une interface administrateur pour suivre les réservations et le chiffre d'affaires. Seule une version mobile n'a pas été développée, mais tout le reste fonctionne correctement.

Ce projet m'a permis d'apprendre beaucoup de nouvelles choses. J'ai découvert des technologies que je ne connaissais pas avant comme l'API Stripe pour les paiements, Flask pour le serveur web, smtplib pour les emails automatiques, FFmpeg pour l'enregistrement vidéo et MobaXterm pour me connecter à distance au Raspberry Pi. J'ai aussi progressé dans la gestion d'une base de données MySQL hébergée en ligne, dans le câblage des composants électroniques via les broches GPIO et dans la gestion d'un projet complet du début à la fin.

Pour terminer, plusieurs améliorations pourraient être faites dans le futur. La plus importante serait de créer une application mobile pour que les joueurs puissent réserver depuis leur smartphone. Il serait aussi intéressant d'ajouter un système de reconnaissance faciale pour remplacer le code PIN et mieux sécuriser les accès. Enfin, l'interface administrateur pourrait être améliorée avec des statistiques plus détaillées comme des graphiques ou des rapports mensuels automatiques.

9. Sitographie

9.1 Documents

FitFive : le 1er réseau de foot 5/5 et padel en Belgique. (2025, 3 février). FitFive. https://www.fitfive.be/ Consulté le 12 décembre 2025.

SunFounder. (s.d.). Pour les utilisateurs de Raspberry Pi [Documentation]. SunFounder Universal Maker Sensor Kit. https://docs.sunfounder.com/projects/umsk/fr/latest/05_raspberry_pi/raspberry_pi.html Consulté le 12 décembre 2025.

Mastering Date Selection in Python Tkinter. (2024, 3 novembre). YouTube. https://www.youtube.com/watch?v=qUK_rktHrCU Consulté le 8 janvier 2026

Alwaysdata. (s.d.). MariaDB / MySQL [Documentation]. Alwaysdata Help. https://help.alwaysdata.com/en/databases/mariadb/ Consulté le 6 février 2026

de Langen, J. (s.d.). Sending emails with Python. Real Python. https://realpython.com/python-send-email/ Consulté le 6 février 2026

Raspberry Pi Ltd. (s.d.). *Raspberry Pi 4 Model B specifications*. Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ Consulté le 6 mars 2026

Raspberry Pi Ltd. (s.d.). *GPIO documentation*. Raspberry Pi. https://www.raspberrypi.com/documentation/computers/raspberry-pi.html Consulté le 6 mars

Python Software Foundation. (s.d.). *gpiozero – Servo*. Read The Docs. https://gpiozero.readthedocs.io/en/stable/api_output.html#servo Consulté le 8 mars

Pattabiraman, K. (s.d.). *How to set up a keypad on an Arduino*. Circuit Basics. https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/ Consulté le 8 mars

Stripe. (s.d.). *Stripe API documentation*. Stripe. https://stripe.com/docs Consulté le 23 mars 2026

Python Software Foundation. (s.d.). *smtplib — SMTP protocol client*. Python Documentation. https://docs.python.org/3/library/smtplib.html Consulté le 10 avril 2026

FFmpeg. (s.d.). *Ffmpeg documentation*. Ffmpeg. https ://ffmpeg.org/documentation.html Consulté le 27 avril 2026

Passion Électronique. (s.d.). *Afficheur OLED i2c Arduino (avec contrôleur SSD1306 monochrome 128×64 pixels) : branchement, librairie, fonctions graphiques, et exemples de code*. Passion Électronique. https ://passionelectronique.fr/ecran-oled-i2c-arduino/ Consulté le 11 mai 2026

10. Annexe

Code source de l'application (VS Code)

Importations des bibliothèques nécessaires.

```python
import tkinter as tk
from tkinter import ttk, messagebox
from tkcalendar import Calendar
from PIL import Image, ImageTk
import random
from datetime import datetime
import mysql.connector
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import requests
import webbrowser
import os
```

Vérification des créneaux occupés

```python
def get_creneaux_occupes(terrain, date_str):
    """Récupère les horaires déjà réservés en DB pour ce terrain/date."""
    cur = get_cursor()
    if cur is None:
        return []
    try:
        date_mysql = datetime.strptime(date_str, "%d/%m/%Y").strftime("%Y-%m-%d")
        cur.execute(
            "SELECT horaire FROM reservations WHERE terrain = %s AND date_resa = %s",
            (terrain, date_mysql)
        )
        return [row[0] for row in cur.fetchall()]
    except Exception:
        return []
```

Module Paiement

```python
def lancer_paiement():
    status_label.config(text="⏳ Connexion au serveur de paiement...", fg="orange")
    pay.update()
    try:
        response = requests.post(
            "http://localhost:5000/create-session",
            json={"prix": prix_total, "terrain": terrain, "email": email},
            timeout=5
        )
        response.raise_for_status()
        data = response.json()
        webbrowser.open(data["url"])
        status_label.config(text="🌍 Paiement ouvert dans le navigateur.", fg="blue")
        verifier_paiement(data["session_id"])
```

Vérification du paiement

```
938     def verifier_paiement(session_id):
939         def check():
940             try:
941                 resp = requests.get(
942                     f"http://localhost:5000/check-status/{session_id}",
943                     timeout=2
944                 )
945                 if resp.json().get("status") == "succeeded":
946                     finaliser_reservation()
947                 else:
948                     pay.after(2000, check)
949             except Exception:
950                 pay.after(2000, check)
951         check()
```

Importations et configuration du serveur de paiement

```
1   from flask import Flask, request, jsonify
2   import stripe
3
4   app = Flask(__name__)
5
6   stripe.api_key = "sk_test_51T011WFRsdcJhZgHaII1hEjv173H1etdHeFiQB1oBLcgAZ31AC4mQnGEsAaHFFjNqPAAqO71FC
7
8   SESSION_STORE = {}
9
10
```

Vérification du statut de paiement

```
46  @app.route('/check-payment/<session_id>', methods=['GET'])
47  def check_payment(session_id):
48      try:
49          session = stripe.checkout.Session.retrieve(session_id)
50          if session.payment_status == 'paid':
51              SESSION_STORE[session_id] = {'status': 'succeeded'}
52              return jsonify({'status': 'succeeded'})
53          else:
54              return jsonify({'status': 'pending'})
55      except Exception as e:
56          return jsonify({'error': str(e)}), 500
```

Lancement du serveur

```
239 if __name__ == '__main__':
240     print("=" * 50)
241     print("🚀 SERVEUR DE PAIEMENT STRIPE - FIT FIVE")
242     print("📡 Écoute sur : http://localhost:5000")
243     print("⚠️  NE FERMEZ PAS CETTE FENÊTRE !")
244     print("=" * 50)
245
246     app.run(host='127.0.0.1', port=5000, debug=False)
247
```

Code Raspberry Pi (main.py)

Connexion à la base de données

```
DB_CONFIG = {
    "host": "mysql-bebless.alwaysdata.net",
    "user": "bebless",
    "password": "Psaume73",
    "database": "bebless_tfe_foot",
    "auth_plugin": "mysql_native_password",
    "connection_timeout": 10
}

db = None

def connecter_db():
    global db
    try:
        db = mysql.connector.connect(**DB_CONFIG)
        print("✅ Connexion DB réussie.")
    except mysql.connector.Error as e:
        print(f"❌ Erreur DB : {e}")
        db = None

def get_cursor():
    global db
    try:
        if db is None or not db.is_connected():
            connecter_db()
        db.ping(reconnect=True, attempts=3, delay=2)
        return db.cursor()
    except Exception as e:
        print(f"❌ Curseur indisponible : {e}")
        return None

connecter_db()
```

Détection de présence (capteur ultrason)

Le **d**éclencheur du système

```
def distance():
    GPIO.output(TRIG, False)
    time.sleep(0.01)
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    debut = time.time()
    fin = time.time()
    timeout = time.time() + 0.1

    while GPIO.input(ECHO) == 0:
        debut = time.time()
        if debut > timeout:
            return 999

    while GPIO.input(ECHO) == 1:
        fin = time.time()
        if fin > timeout:
            return 999

    return (fin - debut) * 34300 / 2
```

Vérification du code en base de données

```
def verifier_code_db(code_saisi):
    cur = get_cursor()
    if not cur: return False, None
    try:
        cur.execute(
            "SELECT id FROM reservations WHERE code = %s AND code_utilise = 0 LIMIT 1",
            (int(code_saisi),)
        )
        res = cur.fetchone()
        if res:
            cur.execute("UPDATE reservations SET code_utilise = 1 WHERE id = %s", (res[0],))
            db.commit()
            return True, res[0]
        return False, None
    except Exception as e:
        print(f"❌ Erreur SQL: {e}")
        return False, None
```

Saisie du code sur le clavier matriciel (keypad 4x4)

```
def get_code():
    code = ""
    print("🔒 En attente du code (# pour valider) :")
    oled_afficher("Entrez votre", "code + #", "* pour effacer")

    while True:
        for i, row in enumerate(ROWS):
            GPIO.output(row, GPIO.LOW)
            for j, col in enumerate(COLS):
                if GPIO.input(col) == 0:
                    key = KEYS[i][j]

                    GPIO.output(row, GPIO.HIGH)
                    print(f"Touche pressée : {key}")
                    # Attend patiemment que l'utilisateur relâche la touche du cla
                    while GPIO.input(col) == 0:
                        time.sleep(0.05)
```

Gestion de la session après validation de l'accès

```
def activation_success(res_id):
    print(f"✅ ACCÈS VALIDÉ (#{res_id})")
    oled_afficher("ACCES AUTORISE", "Bienvenue !", "Borne ouverte")

    set_angle(180)
    leds_toutes_on()
    threading.Thread(target=record_video, args=("success",), daemon=True).start()

    for t in range(120, -1, -1):
        if t == 30:
            print("⏰ 30s restantes")
            leds_rouges_seulement()
            oled_afficher("Attention !", "30 sec restantes", "Terminez svp")

        if t == 0:
            print("🔒 Fermeture.")
            leds_off()
            set_angle(0)
            oled_afficher("Session terminee", "Merci !", "")
            time.sleep(3)
            oled_effacer()

        if t % 10 == 0:
            print(f"⏱ Temps restant : {t}s")

        time.sleep(1)
```