Rapport de Projet

BTS CIEL (cybersécurité, informatique et réseaux, électronique)
option A informatique et réseaux

AirLockUnlock

Bilal TAOUFIK - Adel AICHI - Amine El MIR - Lakshan SANGARALINGAM

Sommaire

Contexte du projet

Ce projet a pour objectif de répondre aux problématiques liées à la gestion des accès dans le cadre de la location de logements, en remplaçant les méthodes traditionnelles de remise des clés par une solution numérique basée sur **les serrures connectées TapKey**.

Aujourd'hui, la remise physique des clés peut entraîner des contraintes logistiques, des retards ou encore des risques de perte ou de vol. Pour remédier à cela, notre plateforme intègre une gestion d'accès automatisée via une **application mobile**. Lorsqu'un utilisateur recherche une location, les logements équipés d'une serrure numérique sont clairement identifiés par un **symbole de cadenas**, permettant une reconnaissance rapide et intuitive.

Grâce à l'intégration des serrures TapKey, le **client peut ouvrir la porte directement depuis l'application**, sans avoir à entrer de code. L'accès est strictement **autorisé uniquement pendant la période de réservation** : si le locataire arrive trop tôt ou après la fin de son créneau, la serrure reste verrouillée. Ce système assure une gestion fine et sécurisée des horaires d'accès.

En résumé, AirLockUnlock propose une solution moderne, sécurisée et automatisée pour gérer l'accès aux locations, en offrant une expérience utilisateur fluide aussi bien pour les propriétaires que pour les locataires.

Objectifs

Le projet AirLockUnlock vise à concevoir un système complet de gestion de locations avec accès connecté, en combinant développement web (Front-end, Back-end), mobile et ioT.

Nos objectifs principaux sont :

- **Développer un site web** permettant aux propriétaires de publier des logements à louer et de gérer leurs biens et leurs réservations. Pour les clients de pouvoir réserver une location.

- **Concevoir une application mobile Android** destinée aux clients, leur permettant de consulter leurs réservations et d'ouvrir la serrure électronique du logement à l'aide de leur smartphone.

- **Intégrer une serrure connectée TapKey** avec un système de contrôle d'accès automatisé, activé uniquement aux horaires de réservation valides.

- **Mettre en place une base de données centralisée** pour gérer les utilisateurs, les logements, les réservations et les autorisations d'accès.

- Proposer une expérience fluide, sécurisée et fiable pour **les deux parties (propriétaire et client)**, sans nécessité de remise de clés physiques.

Problématique

Dans un monde de plus en plus connecté et orienté vers les solutions sans contact, **comment permettre à un client d'accéder à une location de manière autonome, sécurisée et automatisée**, sans passer par une remise de clés physique, tout en assurant aux propriétaires un contrôle total sur l'accès à leurs logements ?

I - Cahier des charges du projet

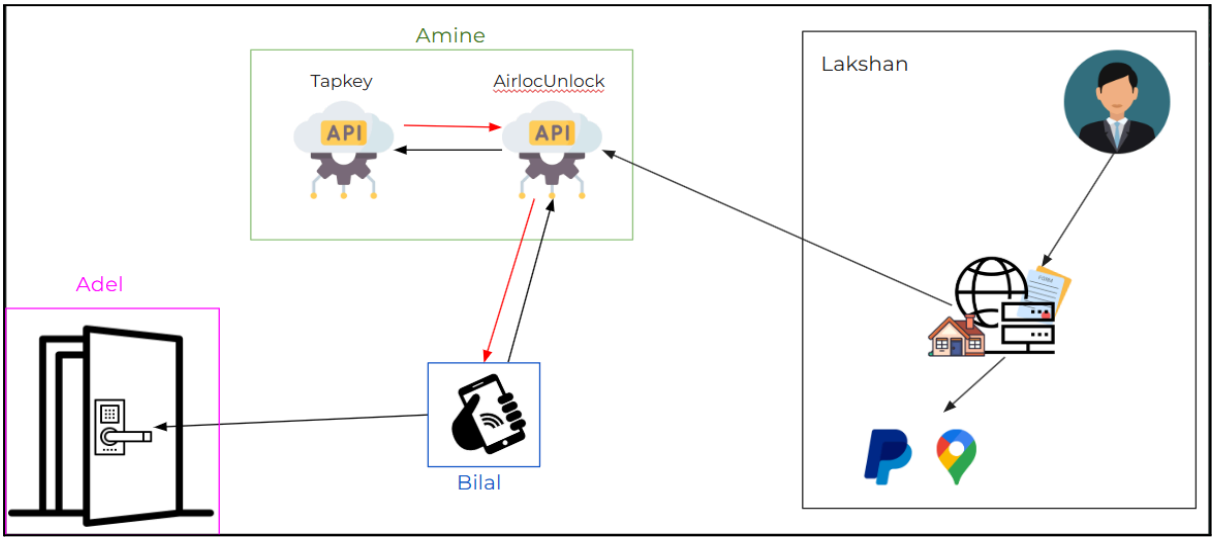
Besoin client

Le client souhaite disposer d'une plateforme complète de gestion de locations courte durée, avec la possibilité pour les locataires d'entrer dans leur logement sans remise de clés, grâce à une serrure connectée. Le système doit permettre aux **propriétaires** de publier des logements et gérer les réservations, et aux **clients** d'accéder à leur logement depuis une **application mobile**, uniquement pendant la période réservée.

Objectifs fonctionnels

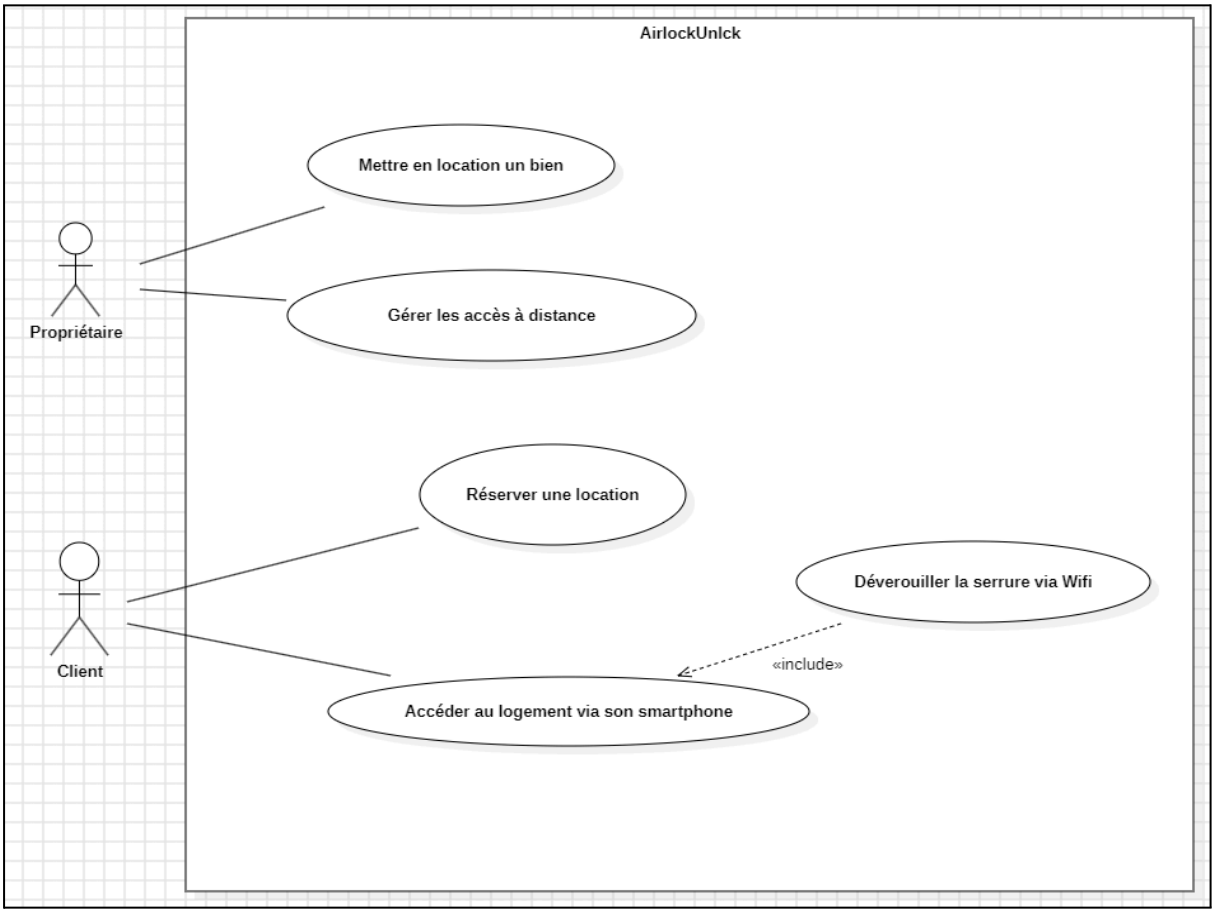
Objectif	Description
Gestion des comptes utilisateurs	Inscription et connexion pour les propriétaires et les clients
Publication de logements	Ajout de biens par les propriétaires
Réservation	Réservation d'un logement par un client avec choix des dates
Accès au logement	Ouverture de la serrure via l'application mobile, uniquement pendant les dates de réservation
Synchronisation IoT	Communication entre l'application et la serrure via l'ESP32

Le synoptique ci-dessous présente une vue d'ensemble du fonctionnement global du projet. Il permet de visualiser de manière claire et structurée les différentes interactions entre les composants matériels et logiciels du système. Ce schéma facilite la compréhension de l'architecture générale, notamment les échanges de données, les flux de communication, ainsi que le rôle de chaque élément impliqué dans le projet.



II - onception UMLs

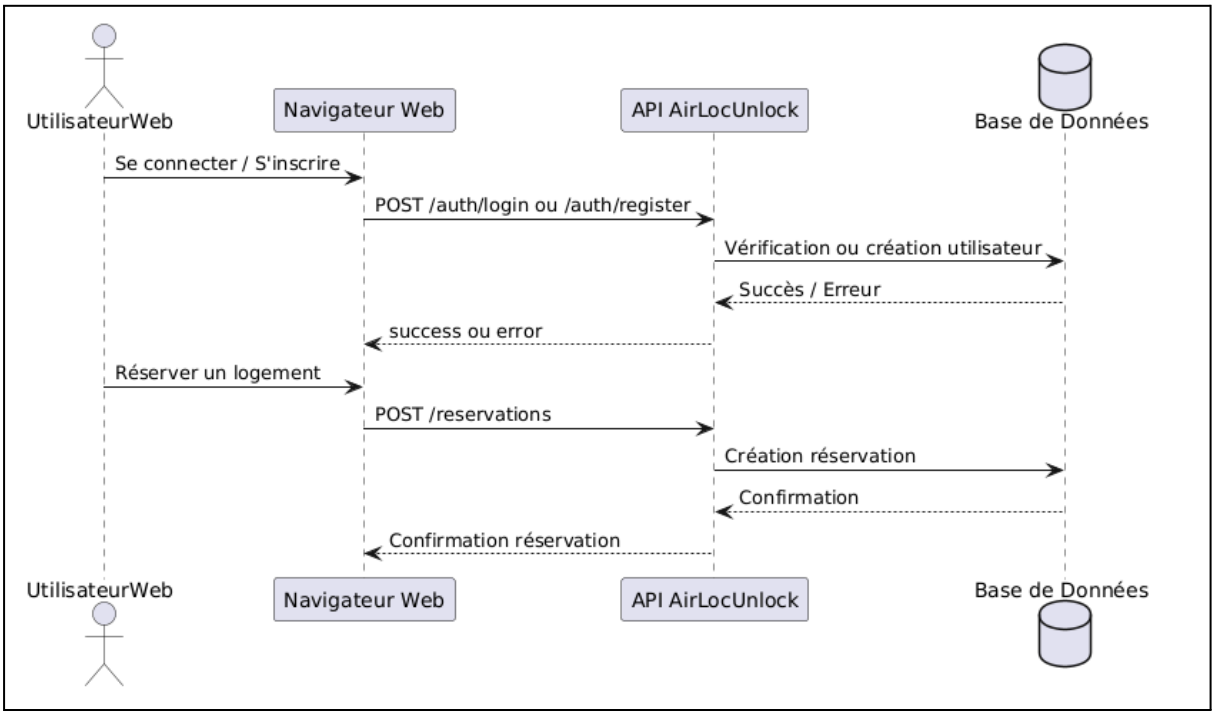
Diagrammes de **cas d'utilisations** :



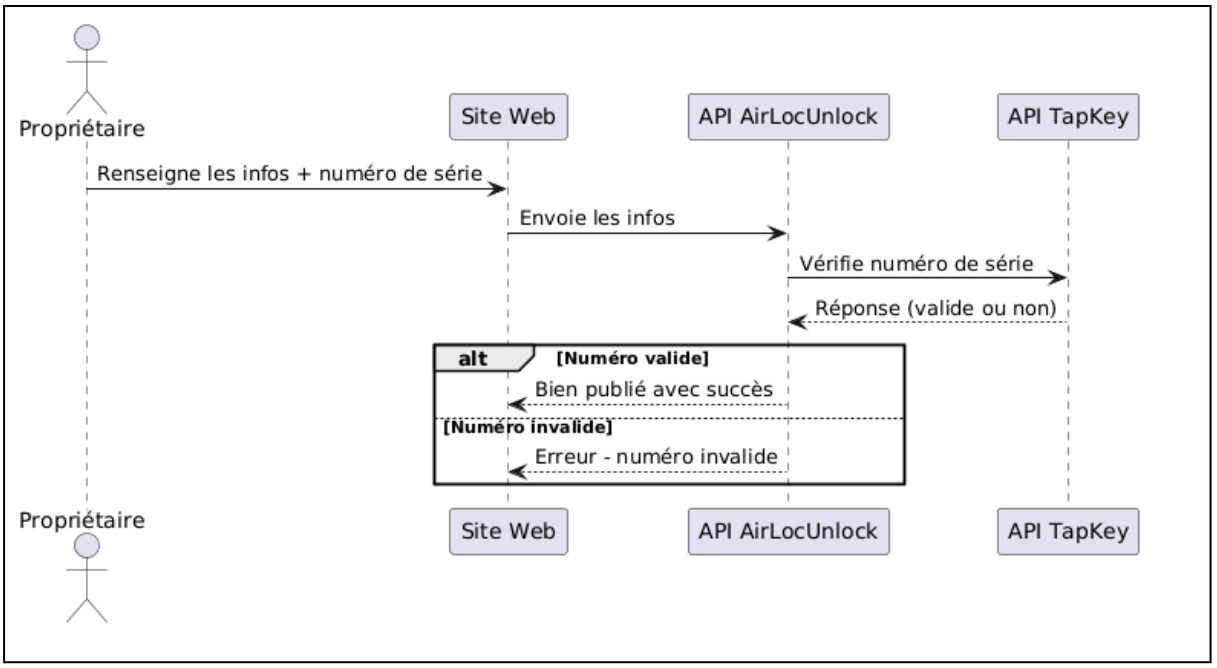
Deux types d'acteurs interagissent avec le système :

- **Le Propriétaire**, a la possibilité de :
    - Mettre en location un bien (ajouter une annonce)
    - Gérer les accès à son logement (en contrôlant les autorisations de déverrouillage)
- **Le Client**, peut :
    - Réserver un logement via la plateforme
    - Accéder au logement avec son smartphone, uniquement durant la période de réservation
    - Cette action inclut l'usage de **Wifi** pour **déverrouiller la serrure connectée**, comme indiqué dans le cas d'utilisation

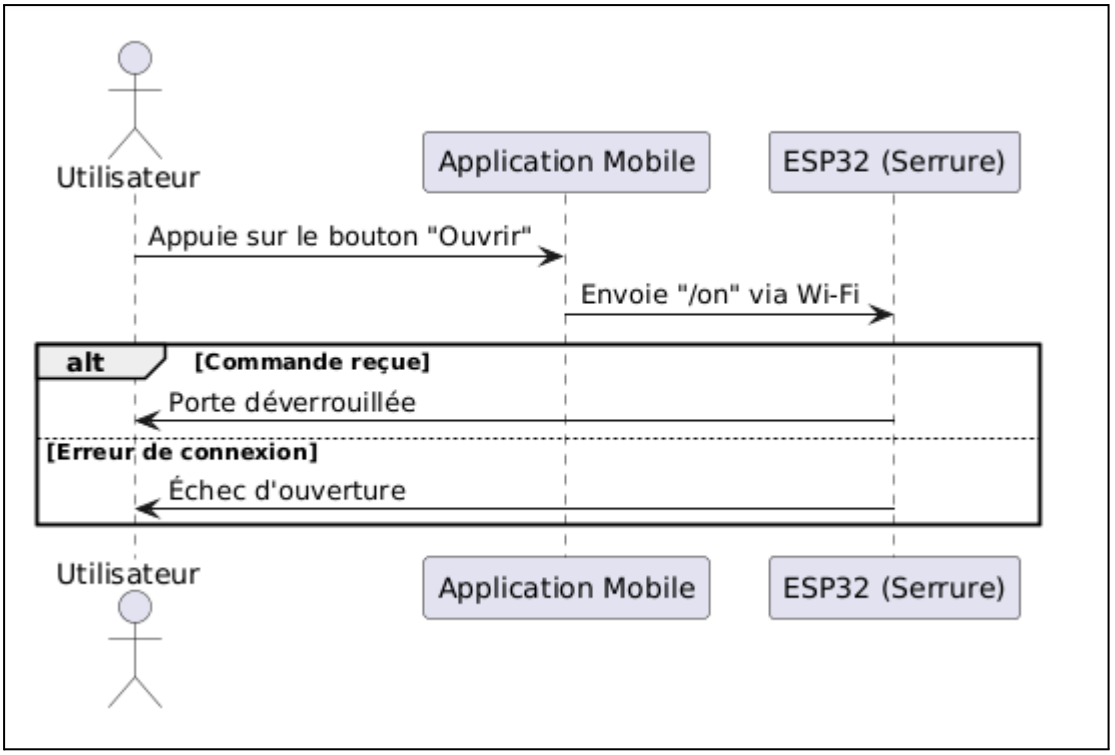
Diagrammes de séquence **Site Web** :



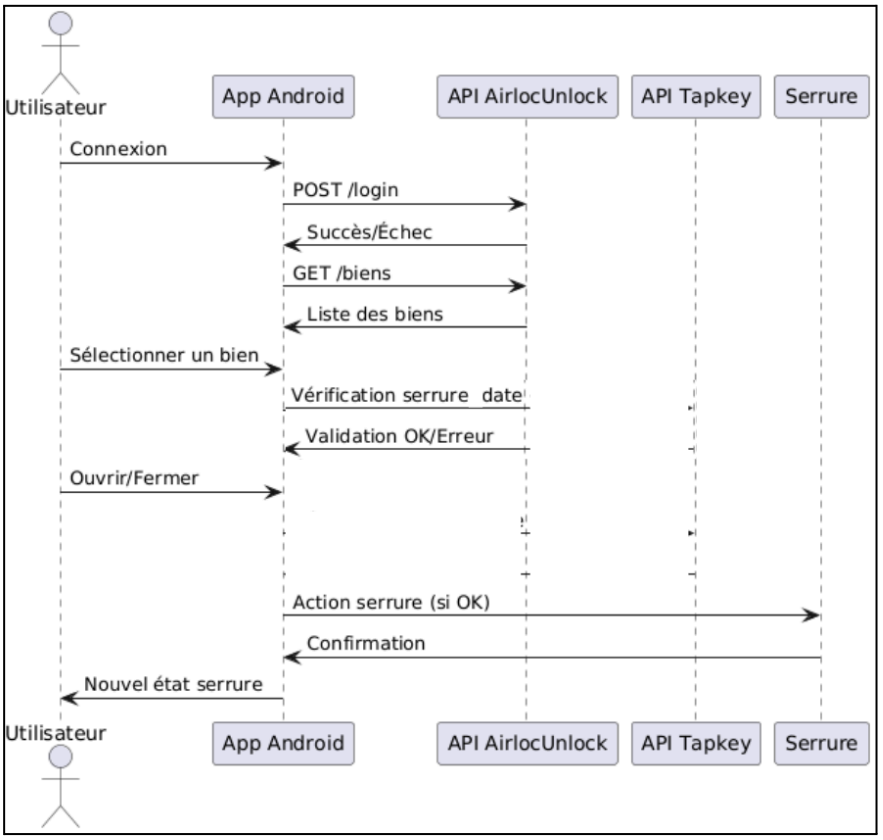
Diagrammes de séquence **APIs** :



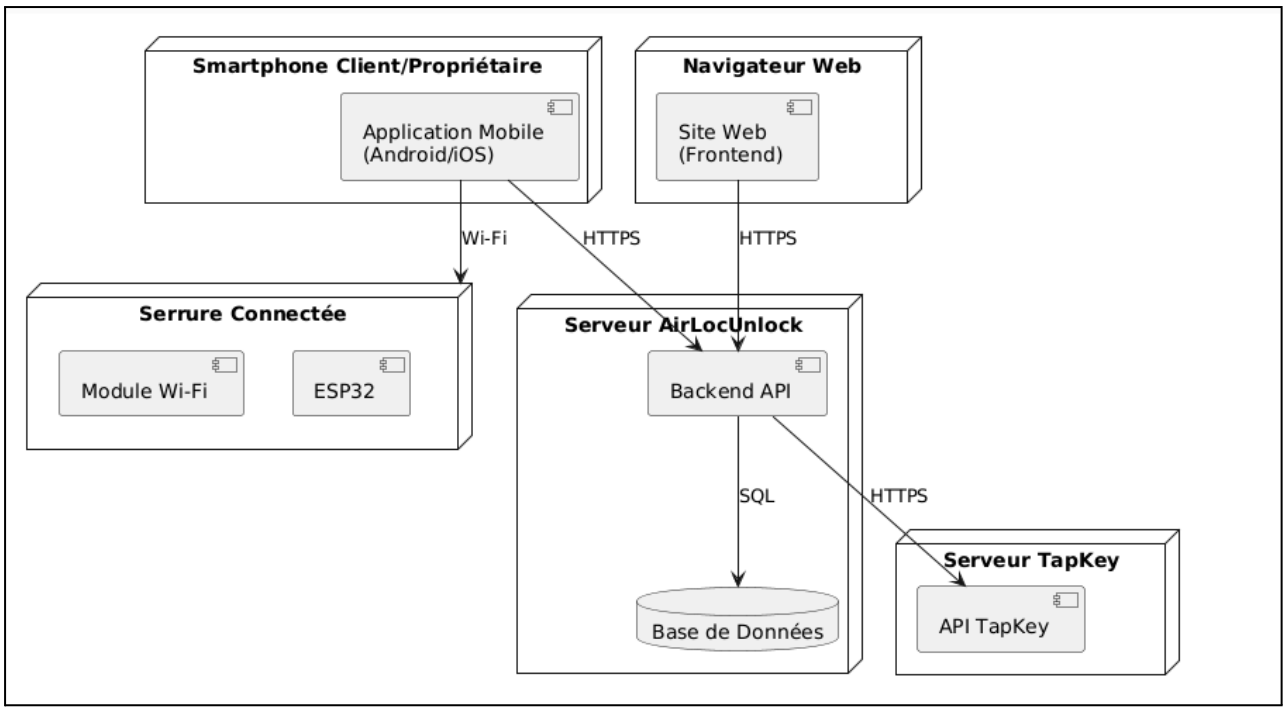
Diagrammes de séquence **IoT** :



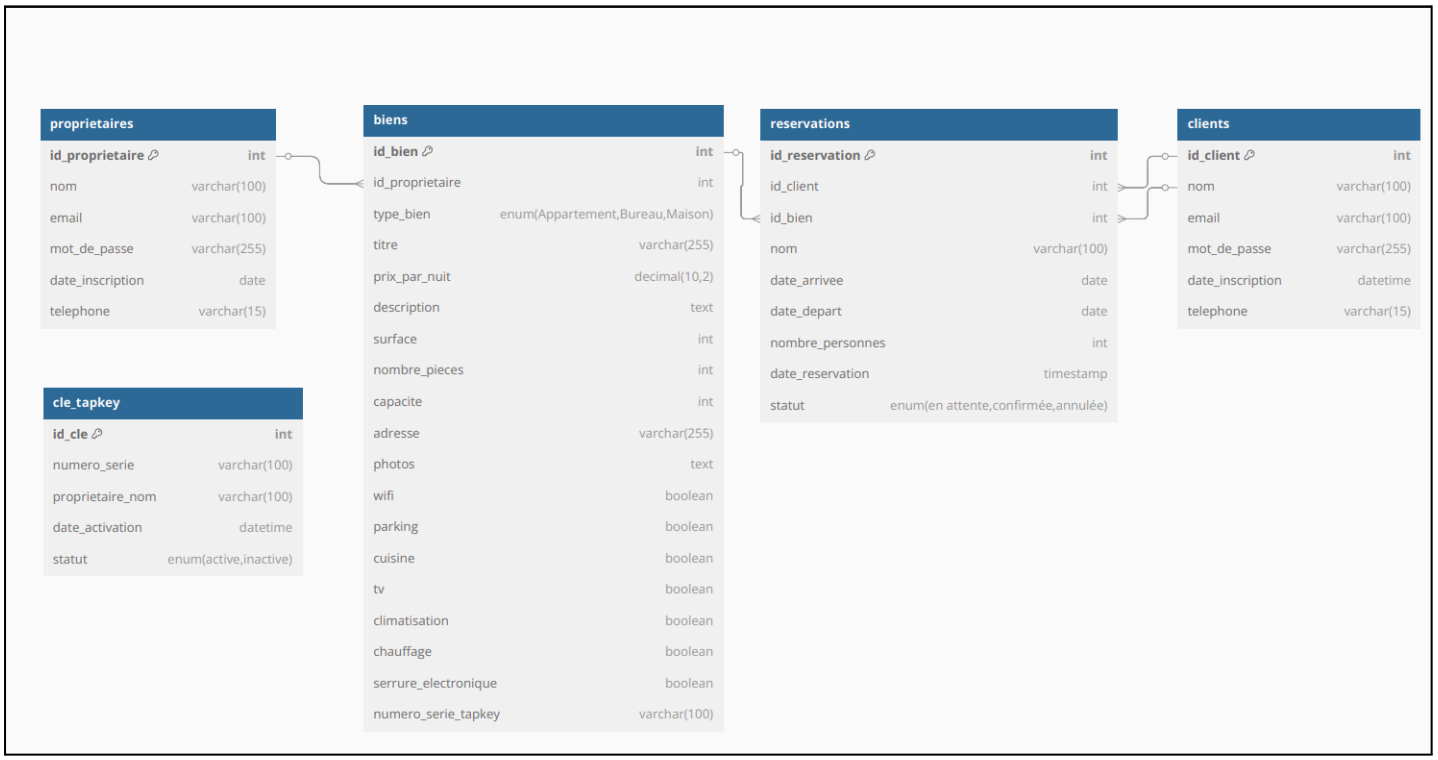
Diagrammes de séquence **Application Mobile** :



Diagrammes de **Déploiement** :



MCD (Modèle Conceptuel de Données) **:**



III - Répartition des tâches (4 étudiants)

Nom de l'étudiant	Rôle principal	Tâches principales
Lakshan SANGARALINGAM	**Site Web**	- Développement de l'interface pour les utilisateur - Gestion des logements
Amine EL MIR	**APIs**	- Création de la base de données SQL avec phpmyadmin - Développement des l'APIs AirlockUnlock et Tapkey en PHP
Adel AICHI	**IoT (ESP32)**	- Configuration de l'ESP32 - Contrôle de la serrure
Bilal TAOUFIK	**Application mobile**	- Développement de l'interface Android - Connexion à l'API - Affichage des réservations - Commande d'ouverture

IV - Méthodologie de travail

Pour réaliser le projet **AirLocUnlock**, nous avons suivi une méthode de travail organisée en plusieurs étapes. Nous avons avancé étape par étape tout en restant attentif aux contraintes ou problèmes menant à des changements, si nécessaires.

A - Analyse des besoins

Nous avons commencé par identifier les attentes des utilisateurs cibles : les clients et les propriétaires. Cette phase a permis de définir les principales fonctionnalités à mettre en place.

B - Conception des diagrammes

Nous avons réalisé différents schémas pour structurer le projet : cas d'utilisation, MCD, et maquettes visuelles sur Figma. Cela nous a permis d'avoir une vision claire de l'architecture du système.

C - Répartition des tâches

Chaque membre du groupe a pris en charge une partie du projet. En suivant l'avancement sur un TRELLO. (https://trello.com/b/P2KbRquX/projet-airlockunlock). Un Drive partager entre les membres du groupes, qui nous a servis se partager les documents nécessaire (images, diagramme, etc)

D - Développement

Chacun a développé sa partie en respectant les besoins définis. On a utilisé Github pour suivre les versions. (https://github.com/JR-CIEL-2-PROJETS/25-airlocunlock.git)

E - Tests

Nous avons effectué des tests fonctionnels sur l'ensemble des modules développés

F - Corrections

Après les tests, on a corrigé les bugs et amélioré certaines parties.

G - Préparation finale

Enfin nous avons préparé une démonstration, un diaporama et ce rapport pour la présentation finale.

V - Développement du projet

A - Outils utilisés

Dans le cadre de ce projet, nous avons utilisé plusieurs outils pour la gestion, le développement, le test et le déploiement :

Outils de développement :

- **Visual Studio Code** : logiciel pour coder (HTML/CSS, JavaScript, PHP).
- **Android Studio** : utilisé pour le développement de l'application mobile Android (Java).
- **Arduino** : pour la programmation du microcontrôleur ESP32 chargé de contrôler la serrure.

Outils web et mobile :

- **HTML / CSS / JavaScript** : pour créer l'interface web utilisateur.
- **PHP** : pour développer l'API côté serveur.
- **Java (Android)** : pour créer une application mobile qui interagit avec l'API et l'ESP32.

Outils backend :

- **Docker** : pour créer des environnements isolés (serveur web local, base de données).
- **phpMyAdmin** : interface de gestion de la base de données MySQL utilisée dans le projet.

Outils de test :

- **Postman :** pour simuler des réponses d'API pendant la phase de développement.
- **Wireshark** : pour tester les requêtes HTTP/HTTPS.

Outils de gestion de projet

- **Git / GitHub** : pour la gestion de version et la collaboration en équipe.
- **Trello** : pour organiser les tâches et suivre l'avancement du projet.
- **Figma** : visualisation de nos idées.

B - Développement individuel

Les APIs AirLockUnlock & Tapkey (Amine El mir)

Introduction

Mon projet a principalement porté sur le développement de l'API AirlockUnlock, qui constitue le cœur du système en assurant la communication entre le site web, l'application Android et l'API Tapkey. Cette API gère les interactions aussi bien du côté des clients que des propriétaires, en facilitant notamment la gestion des biens mis en location ainsi que l'intégration avec les serrures numériques via l'API Tapkey.

Gestion des utilisateurs (clients et propriétaires)

- Authentification : L'API vérifie l'identité des clients, une fonctionnalité accessible aussi bien sur l'application Android que sur le site web.

- Création et gestion des comptes : Elle permet l'enregistrement des nouveaux utilisateurs en stockant leurs données personnelles telles que nom, email, etc.

Gestion des biens

- Ajout et administration des logements : Les propriétaires peuvent ajouter, modifier ou supprimer leurs biens (titre, prix, nombre de chambres, etc.) via le site web.

- Accès aux détails des logements : L'API fournit des méthodes pour consulter les informations des biens en location, accessibles uniquement par le site web.

Intégration avec les serrures connectées

L'API AirlockUnlock s'appuie sur l'API Tapkey pour gérer l'intégration des serrures électroniques, garantissant une synchronisation efficace entre le système et les équipements connectés.

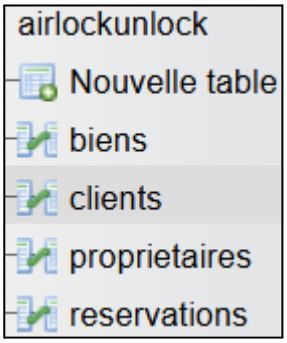
Mise en place des API (AirlockUnlock et Tapkey)

L'API AirlockUnlock est une interface centrale qui gère les inscriptions et les connexions des utilisateurs, qu'ils soient clients ou propriétaires, au sein de la plateforme. Elle permet une gestion sécurisée et fluide des accès à l'application.

1 - Inscription des utilisateurs (Propriétaires et Clients

L'API AirlockUnlock gère la création de compte client en exécutant une requête qui insère les informations fournies par l'utilisateur dans la base de données. Lorsqu'un utilisateur souhaite devenir client, les données envoyées via un formulaire POST (nom, email, mot_de_passe, téléphone) sont récupérées, validées, puis enregistrées dans la table clients de la base AirlockUnlock , permettant ainsi la création d'un nouveau profil client.

```php
$sql = "INSERT INTO clients (nom, email, mot_de_passe, telephone) VALUES (:nom, :email, :mot_de_passe, :telephone)";
try {
    $stmt = $pdo->prepare($sql);
    $stmt->bindParam(':nom', $nom);
    $stmt->bindParam(':email', $email);
    $stmt->bindParam(':mot_de_passe', $mot_de_passe_hash);
    $stmt->bindParam(':telephone', $telephone);
```



id_client	nom	email	mot_de_passe	date_inscription	telephone
1	Client Admin	client@admin.com	$2y$10$GEq6aEZHERl/tHB2b0fBr.nqfQ9SxvtFybmgWWbU2ny...	2025-04-08 09:27:45	0614589324
2	Bilal Taoufik	bilal@gmail.com	$2y$10$Noi/ZpDGHQM/VPpOcBJW1eKhocXGg7yMeUgEgnPMZnU...	2025-04-08 09:31:07	0626266252
3	Adel Aichi	adel@gmail.com	$2y$10$1VZPPJ9t3feNhFQE3bPVGOGt7lYaKlNm24CxDbPj.HQ...	2025-04-08 09:31:07	0624589357
5	toto	toto@lycee-jeanrostand.fr	$2y$10$vGYrKL5l2U0Uf1vmLgWXZ.35PK2BNvMQBaqNRoMNxX/...	2025-04-10 13:36:07	0102030405

L'API AirlockUnlock gère également la création de compte pour les utilisateurs souhaitant devenir propriétaires.

En effet, ils suivent le même protocole que pour les clients : les informations saisies via un formulaire (nom, email, mot de passe, téléphone) sont envoyées par POST, puis récupérées, traitées et validées.

Cependant, ces données sont ensuite insérées dans la base AirlockUnlock, mais dans une table distincte appelée propriétaires.

Ainsi, cette organisation permet de distinguer clairement les profils clients des profils propriétaires, tout en utilisant un processus d'inscription commun. Par conséquent, la gestion des utilisateurs selon leur rôle sur la plateforme est simplifiée et optimisée.

2 - Connexion des utilisateurs (Propriétaires et Clients)

L'API AirlockUnlock gère la connexion des utilisateurs qu'ils soient clients ou propriétaires. grâce à un système d'authentification basé sur un token JWT.

```php
$stmt = $pdo->prepare("SELECT id_client, nom, email, mot_de_passe FROM clients WHERE email = :email");
$stmt->bindParam(':email', $email);
$stmt->execute();
$client = $stmt->fetch(PDO::FETCH_ASSOC);

if (!$client || !password_verify($mot_de_passe, $client['mot_de_passe'])) {
    echo json_encode(['status' => 'error', 'message' => 'Identifiants incorrects.']);
    exit();
}
```

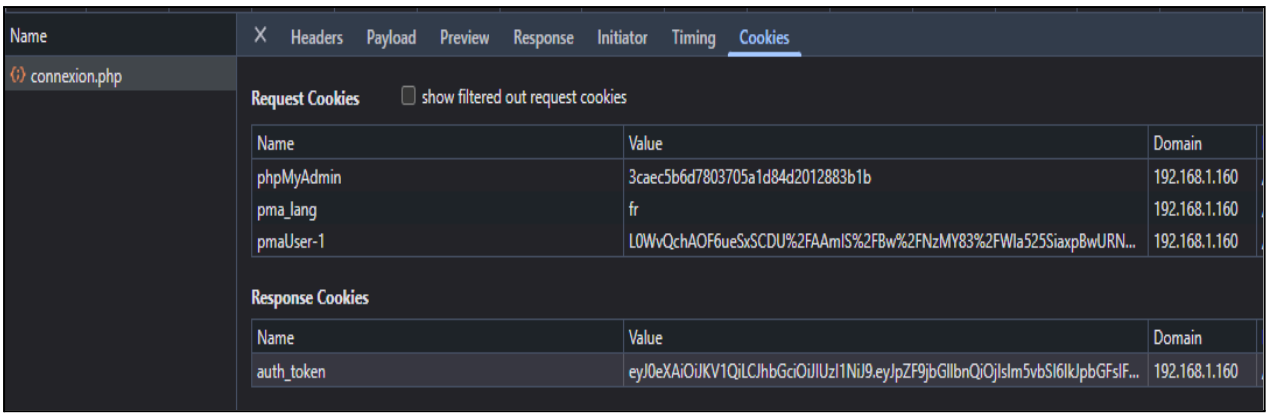
Cette partie de code interroge la base de données pour trouver un client correspondant à l'adresse email fournie, puis vérifie la validité de ses identifiants. Si aucune correspondance n'est trouvée ou si les informations sont incorrectes, une erreur est renvoyée et le processus est interrompu.

Ensuite lorsque l'API AirlockUnlock vérifie ces informations dans la base de données.

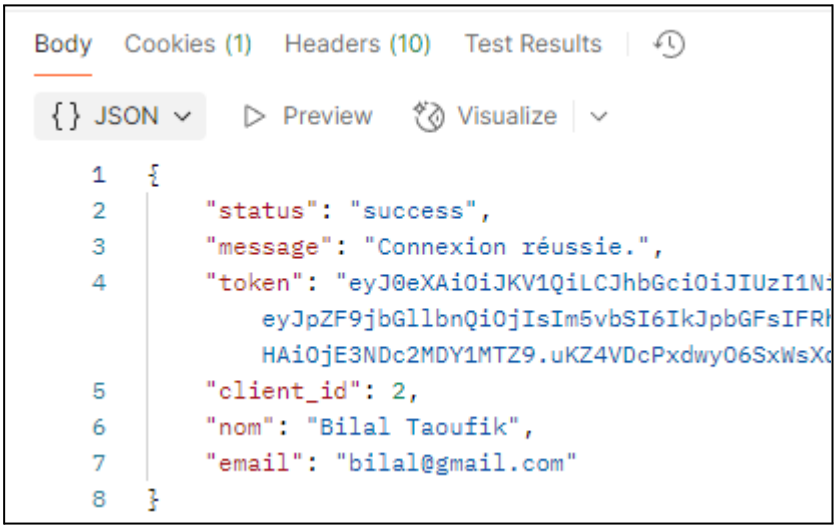
```php
$key = getenv('JWT_SECRET_KEY');
if (!$key) {
    echo json_encode(['status' => 'error', 'message' => 'Erreur : Clé secrète manquante.']);
    exit();
}

$iat = time();
$exp = $iat + 3600;
$payload = [
    'id_client' => $client['id_client'],
    'nom' => $client['nom'],
    'role' => 'client',
    'iat' => $iat,
    'exp' => $exp
];

$jwt = JWT::encode($payload, $key, 'HS256');
```

Si elles sont correctes, le serveur récupère une clé secrète pour créer un token JWT. Ce token contient l'identifiant unique du client, son nom, son rôle, ainsi que les dates de création et d'expiration (valide une heure).

Une fois généré, ce token est stocké dans un cookie nommé auth_token qui expire également au bout d'une heure. Ce cookie permet au navigateur de transmettre automatiquement le token à chaque nouvelle requête, assurant une authentification continue sans que l'utilisateur ait besoin de ressaisir ses identifiants.



Enfin, l'API renvoie une réponse confirmant la réussite de la connexion avec le token et les informations du client.



Ce protocole est également effectué pour les propriétaires.

3 - Publication d'un bien par le propriétaire

Lorsqu'un propriétaire souhaite publier un bien, l'API prend en compte toutes les informations fournies, telles que le type de bien, le prix, les équipements, et plus particulièrement la présence éventuelle d'une serrure électronique.

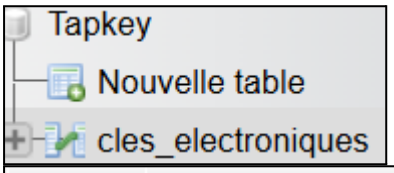
Lorsqu'un propriétaire soumet un bien, l'API AirlockUnlock récupère toutes les informations via la méthode POST :

```php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $id_proprietaire = $_POST['id_proprietaire'];
    $type_bien = $_POST['type_bien'];
    $titre = $_POST['titre'];
    $prix_par_nuit = $_POST['prix_par_nuit'];
    $description = $_POST['description'];
    $surface = $_POST['surface'];
    $nombre_pieces = $_POST['nombre_pieces'];
    $capacite = $_POST['capacite'];
    $adresse = $_POST['adresse'];
    $wifi = isset($_POST['wifi']) ? 1 : 0;
    $parking = isset($_POST['parking']) ? 1 : 0;
    $cuisine = isset($_POST['cuisine']) ? 1 : 0;
    $tv = isset($_POST['tv']) ? 1 : 0;
    $climatisation = isset($_POST['climatisation']) ? 1 : 0;
    $chauffage = isset($_POST['chauffage']) ? 1 : 0;
    $serrure_electronique = isset($_POST['serrure_electronique']) ? 1 : 0;
    $numero_serie_tapkey = $_POST['numero_serie_tapkey'];
```

```php
if (!empty($numero_serie_tapkey)) {
    $numero = trim($numero_serie_tapkey);
    $stmt = $pdo_tapkey->prepare("SELECT COUNT(*) FROM Tapkey.cles_electroniques WHERE numero_serie = :numero");
    $stmt->execute([':numero' => $numero]);
    if ($stmt->fetchColumn() == 0) {
        echo json_encode(['error' => 'Ce numéro de série Tapkey n'existe pas.']);
        exit();
    }
    $stmt = $pdo->prepare("SELECT COUNT(*) FROM biens WHERE numero_serie_tapkey = :numero");
    $stmt->execute([':numero' => $numero]);
    if ($stmt->fetchColumn() > 0) {
        echo json_encode(['error' => 'Ce numéro de série Tapkey est déjà utilisé.']);
        exit();
```

Lorsque le propriétaire indique une serrure électronique, l'API AirlockUnlock interroge l'API Tapkey pour vérifier si le numéro de série fourni existe bien et n'est pas déjà utilisé par un autre bien. Si le numéro est inexistant ou déjà attribué, une erreur est immédiatement renvoyée.

L'API Tapkey s'assure de la validité du numéro de série fourni par le propriétaire en le comparant aux enregistrements de sa table cle_electroniques, où il est référencé sous le champ numero_serie.



id_cle	numero_serie	proprietaire_nom	date_activation	statut
1	TK-001-ADMIN	Propriétaire Admin	2025-04-08 09:38:37	active
2	TK-002-XYZ	Lakshan Sangaralingam	2025-04-08 09:38:37	active
3	TK-003-TEST	Amine El mir	2025-04-08 09:38:37	active

Après avoir validé les informations, notamment la vérification du numéro de série de la serrure électronique avec l'API Tapkey, l'API AirlockUnlock insère les données du bien dans la table biens.

id_bien	id_proprietaire	type_bien	titre	prix_par_nuit
1	1	Maison	Villa en bord de mer	120.00

serrure_electronique	numero_serie_tapkey
1	TK-001-ADMIN

On constate qu'un bien enregistré dans la base AirlockUnlock renseigne l'ID du propriétaire, l'ID du bien et le numéro de série Tapkey si une serrure électronique est sélectionnée.

4 - Réservation d'un bien par le client

Lorsqu'un client souhaite réserver un bien, il doit d'abord indiquer plusieurs informations essentielles : la date d'arrivée, la date de départ, ainsi que le nombre de personnes qui séjourneront. Ces données permettent de définir précisément la période et les besoins liés à la réservation.

```php
$id_bien = $_POST['id_bien'];
$date_arrivee = $_POST['date_arrivee'];
$date_depart = $_POST['date_depart'];
$nombre_personnes = $_POST['nombre_personnes'];
```

Ces données sont envoyées en POST à l'API AirlockUnlock, qui les traite et enregistre la réservation dans la base de données.

```php
$sql = "INSERT INTO reservations (id_client, id_bien, date_arrivee, date_depart, nombre_personnes, statut)
        VALUES (:id_client, :id_bien, :date_arrivee, :date_depart, :nombre_personnes, 'confirmée')";
```

L'API AirlockUnlock reçoit les informations de réservation envoyées par le client.

Ensuite, elle exécute cette requête SQL pour enregistrer la réservation dans la base de données.
L'API associe ainsi l'identifiant du client et du bien, enregistre les détails de la réservation, et définit automatiquement le statut à « confirmer » lorsque toutes les conditions sont respectées.

id_reservation	id_client	id_bien	nom	date_arrivee	date_depart	nombre_personnes	date_reservation	statut
1	1	1	Client Admin	2025-04-15	2025-04-20	4	2025-04-08 09:35:08	confirmée

La table Réservations stocke alors l'identifiant du client, son nom, l'identifiant du bien réservé, ainsi que l'identifiant unique de la réservation avec les autres informations.

5 - Réservations et biens (Propriétaire et clients)

Les clients peuvent consulter toutes leurs réservations directement depuis leur interface sur le site grâce à cette requête traitée par l'API AirlockUnlock.

```
$sql = "SELECT
            r.id_reservation,
            r.date_arrivee,
            r.date_depart,
            r.nombre_personnes,
            r.statut,
            b.titre,
            b.photos
        FROM reservations r
        INNER JOIN biens b ON r.id_bien = b.id_bien
        WHERE r.id_client = :id_client
        ORDER BY r.date_arrivee DESC";
```

Cette requête sert à récupérer toutes les réservations faites par un client. Elle cherche dans la base de données toutes les réservations liées à ce client, puis rassemble aussi des informations sur les biens réservés, comme leur titre et leurs photos.

Les résultats sont ensuite classés pour afficher en premier les réservations avec les dates d'arrivée les plus proches ou récentes.

Les propriétaires peuvent également visualiser leurs biens mis en location grâce à cette requête. Elle récupère tous les biens dans la table biens qui appartiennent au propriétaire connecté identifié.

```
$sql = "SELECT * FROM biens WHERE id_proprietaire = :id_proprietaire";
```

L'API AirlockUnlock permet aux propriétaires de consulter toutes les informations détaillées de leurs biens mis en location, incluant notamment la présence ou non d'une serrure électronique.

6 - Sécurité

Dans le cadre du développement de l'API AirlockUnlock, plusieurs mesures de sécurité ont été mises en œuvre afin de protéger les données des utilisateurs et sécuriser les échanges entre le client et le serveur.

Passage en HTTPS

Pour assurer la confidentialité des échanges entre le client (navigateur ou application) et le serveur, l'API a été configurée pour fonctionner en HTTPS via la méthode OpenSSL. Cela garantit que toutes les données transmises sont chiffrées et protégées contre l'interception (attaque de type "man-in-the-middle").

- L'API écoute sur le port sécurisé 421.
- Le protocole HTTPS empêche la lecture ou la modification des données pendant leur transit.

Hachage des mots de passe

Pour protéger les mots de passe des utilisateurs (propriétaires et clients), ceux-ci ne sont jamais stockés en clair dans la base de données. À la place, un algorithme de hachage sécurisé est utilisé :

```
$mot_de_passe_hash = password_hash($mot_de_passe, PASSWORD_BCRYPT);
```

La fonction password_hash utilise l'algorithme sécurisé BCRYPT pour transformer les mots de passe en une suite chiffrée unique, intégrant un sel aléatoire propre à chaque mot de passe.

Cette méthode protège efficacement les mots de passe stockés, rendant leur récupération quasi impossible même en cas de compromission de la base de données.

Lors de la connexion, la fonction password_verify compare le mot de passe saisi par l'utilisateur avec ce hachage pour valider son identité.

Gestion des sessions et authentification via JWT (token)

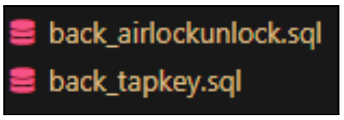
```php
$key = getenv('JWT_SECRET_KEY');
if (!$key) {
    echo json_encode(['status' => 'error', 'message' => 'Erreur : Clé secrète manquante.']);
    exit();
}

$iat = time();
$exp = $iat + 3600;
$payload = [
    'id_client' => $client['id_client'],
    'nom' => $client['nom'],
    'role' => 'client',
    'iat' => $iat,
    'exp' => $exp
];

$jwt = JWT::encode($payload, $key, 'HS256');
```

Pour sécuriser l'authentification et les sessions, l'API utilise des JSON Web Tokens (JWT). Ces tokens encodent des informations essentielles sur l'utilisateur (ID, nom, rôle, date d'émission et d'expiration ici 1 heure) et sont signés avec une clé secrète stockée en environnement. Cette signature garantit que le token est authentique et n'a pas été altéré.

Le token JWT est signé avec une clé secrète connue uniquement de l'API, ce qui garantit que seul l'API peut générer et valider un token valide. En plus, comme les échanges se font en HTTPS, toutes les données, y compris le token, sont chiffrées lors de la transmission. Cela empêche toute interception ou modification du token par un attaquant pendant son transfert. Ainsi, la signature combinée à l'usage du protocole HTTPS assure une protection robuste contre les falsifications et les attaques.

Le JWT est stocké dans un cookie sécurisé, configuré pour être transmis uniquement via HTTPS (attribut Secure) et inaccessible au JavaScript côté client (attribut HttpOnly), ce qui protège contre les attaques courantes comme le vol de token par XSS. À chaque requête, le token est vérifié pour autoriser l'accès aux ressources protégées.

Dockerisation



Pour sauvegarder mes bases de données, je génère deux fichiers SQL back_airlockunlock.sql et back_tapkey.sql grâce à ces commandes renseigné dans mon fichier README :

```
sauvegarder la bsdonne

docker exec -i mysql-container mysqldump -u root -proot airlockunlock > back_airlockunlock.sql
docker exec -i mysql-container mysqldump -u root -proot Tapkey > back_tapkey.sql
```

Ces fichiers contiennent toutes les données et la structure des bases Airlockunlock et Tapkey. Ils servent donc de copies complètes pour conserver les informations importantes.

Pour restaurer ces bases sur un autre poste, je commence par lancer les conteneurs Docker avec la commande renseigné dans mon fichier README :

```
lancer les containers

docker compose up -d
```

Ensuite, j'utilise les fichiers d'export pour importer leur contenu dans la base MySQL grâce à des commandes spécifiques, que j'ai également détaillées dans mon fichier README.

```
recuperer ma base de donnés sur un autre pc

docker exec -i mysql-container mysql -u root -proot airlockunlock < back_airlockunlock.sql
docker exec -i mysql-container mysql -u root -proot Tapkey < back_tapkey.sql
```

Cela permet de récupérer toutes les données, même si on change de machine ou de réseau.

Le Site Web (Lakshan Sangaralingam)

Introduction

Dans le cadre du projet AirLockUnlock, j'ai été chargé de développer l'intégralité du site web. Il permet aux clients de réserver des logements, et aux propriétaires de publier et gérer leurs biens.

Le site est connecté à trois services externes : l'API AirlockUnlock pour les données, l'API Tapkey pour les serrures connectées, et PayPal pour les paiements.

Mise en place du site web

1 - Connexion et rôle utilisateur

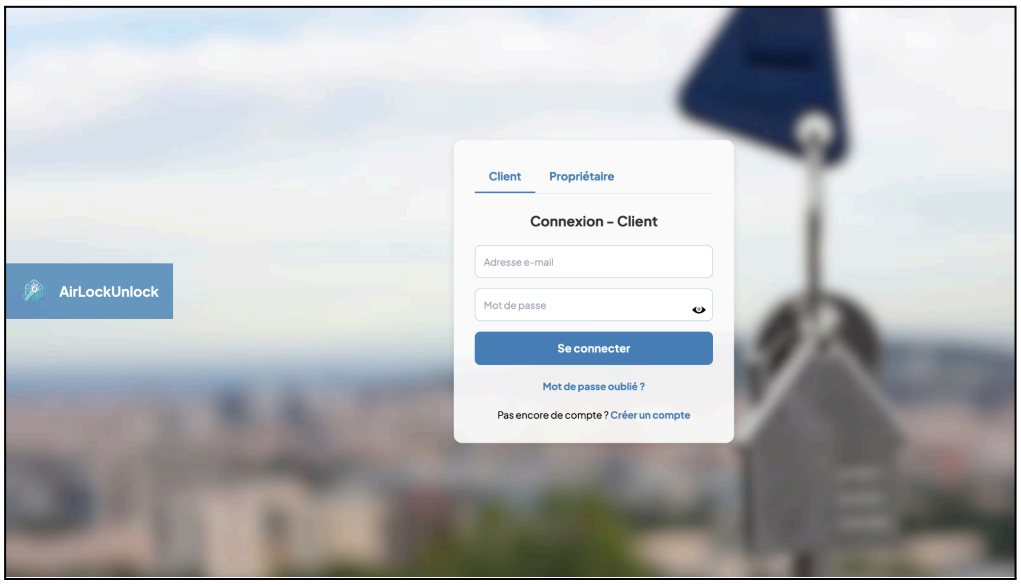
L'utilisateur arrive sur une page de connexion unique avec deux champs : email et mot de passe.
Lors de l'inscription (réalisée depuis une autre page), il choisit son rôle : client ou propriétaire.

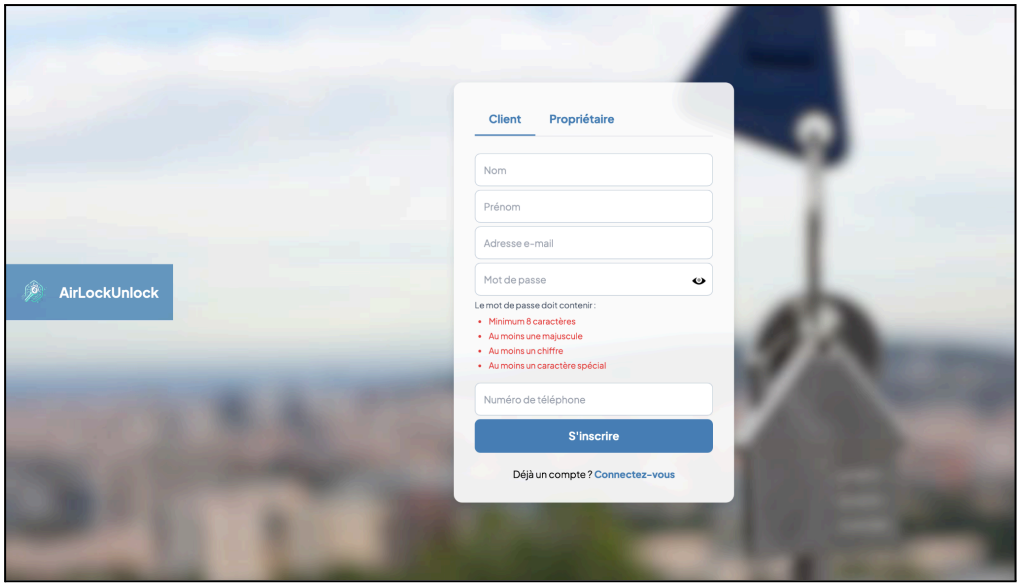
Une fois connecté :

- Un client est redirigé automatiquement vers la page d'accueil avec la liste des logements.
- Un propriétaire accède à un tableau de bord personnalisé.

Fonctionnement technique :

- Envoi d'une requête POST avec fetch à l'API d'authentification.
- Si la réponse est positive : le token est stocké dans les cookies, un message s'affiche, puis redirige.






```
fetch('http://192.0.0.2:91/AirlockUnlock/client/connexion.php', {
    method: 'POST',
    body: formData
})
.then(async (response) => {
    const contentType = response.headers.get("content-type");
    if (!contentType || !contentType.includes("application/json")) {
        throw new Error("La réponse du serveur n'est pas au format JSON.");
    }
    return response.json();
})
.then(data => {
    console.log('Client Response:', data); // Log the response
    if (data.status === 'success') {
        // Stockage du token
        localStorage.setItem('token', data.token);
        messageElement.textContent = 'Connexion réussie ! Redirection...';
        messageElement.style.display = 'block';
        setTimeout(() => {
            window.location.href = 'accueil.html';
        }, 1000);
    } else {
        messageElement.textContent = data.message;
        messageElement.style.display = 'block';
    }
})
```

Ce code JavaScript permet de gérer la connexion d'un client via une requête fetch vers l'API :

- Envoi des identifiants (email et mot de passe) saisis dans un formulaire à l'API client/connexion.php.
- Vérifie si la réponse du serveur est bien au format JSON.
- Si la réponse indique un succès :
    - Le token d'authentification est stocké dans les cookies.
    - Un message de confirmation s'affiche à l'utilisateur.
    - Une redirection automatique vers la page accueil.html est déclenchée après 1 seconde.
- En cas d'échec, un message d'erreur est affiché.

Ce mécanisme garantit une connexion sécurisée, avec gestion du retour API et affichage dynamique des messages.

```
    fetch('http://172.16.15.63:443/AirlockUnlock/proprietaire/connexion.php', {
        method: 'POST',
        body: formData
    })
    .then(async (response) => {
        const contentType = response.headers.get("content-type");
        if (!contentType || !contentType.includes("application/json")) {
            throw new Error("La réponse du serveur n'est pas au format JSON.");
        }
        return response.json();
    })
    .then(data => {
        console.log('Owner Response:', data); // Log the response
        if (data.status === 'success') {
            // Stockage du token
            localStorage.setItem('token', data.token);
            messageElement.textContent = 'Connexion réussie ! Redirection...';
            messageElement.style.display = 'block';
            setTimeout(() => {
                window.location.href = 'accueilpro.html';
            }, 1000);
        } else {
            messageElement.textContent = data.message;
            messageElement.style.display = 'block';
        }
    })
    .catch(error => {
        console.error('Erreur:', error);
        messageElement.textContent = 'Erreur lors de la connexion.';
        messageElement.style.display = 'block';
    });
});
```

Ce bloc JavaScript gère la connexion d'un propriétaire :

- Envoie les identifiants saisis (email et mot de passe) à l'API propriétaire/connexion.php via une requête POST(fetch).
- Vérifie que la réponse est bien au format JSON.
- Si la connexion est un succès :
    - Le token est sauvegardé dans les cookies.
    - Affiche un message de succès.
    - Rédige automatiquement vers accueilpro.html après 1 seconde.
- Sinon, un message d'erreur est affiché à l'écran.
- En cas de problème réseau ou serveur, un message générique s'affiche dans la page.

Cela permet une connexion fluide et sécurisée pour les propriétaires, avec retour visuel et gestion des erreurs.

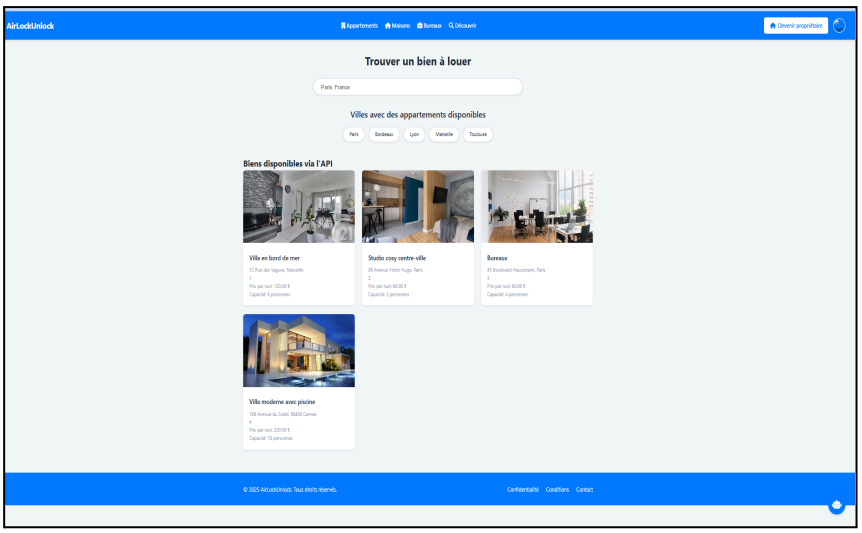
2 - Affichage des logements disponibles

Après la connexion, la page d'accueil affiche l'ensemble des logements disponibles. Les données sont récupérées dynamiquement via l'API AirLockUnlock.

Chaque logement contient :

- Un titre, une image, un prix, une adresse.
- Une icône spécifique si le logement est équipé d'une serrure Tapkey.
- Un bouton ou lien pour afficher les détails du bien.

Un système de filtrage par zone géographique est intégré pour faciliter la recherche.




```
    function displayProperties(biens) {
        const propertiesContainer = document.getElementById('api-properties-list');
        propertiesContainer.innerHTML = ''; // Effacer le contenu existant

        biens.forEach(bien => {
            const propertyDiv = document.createElement('div');
            propertyDiv.className = 'property';
            propertyDiv.innerHTML = `
                <img src="${bien.photo_url}" alt="${bien.titre}">
                <div class="property-details">
                    <h3>${bien.titre}</h3>
                    <p>${bien.adresse}</p>
                    <p>${bien.id_bien}</p>
                    <p>Prix par nuit: ${bien.prix_par_nuit} €</p>
                    <p>Capacité: ${bien.capacite} personnes</p>
                </div>
            `;

            // Ajouter un événement de clic pour rediriger vers la page détaillée de
            propertyDiv.addEventListener('click', () => {
                window.location.href = `appartement.html?id_bien=${bien.id_bien}`;
            });

            propertiesContainer.appendChild(propertyDiv);
        });
    }

    // Appeler la fonction pour récupérer les biens lorsque la page est chargée
    document.addEventListener('DOMContentLoaded', fetchProperties);
</script>
```

Ce code sert à afficher automatiquement tous les logements disponibles sur la page d'accueil.

Quand la page se charge :

- Il récupère les logements depuis l'API.
- Pour chaque logement, il crée une carte avec la photo, le titre, l'adresse, le prix et la capacité.
- Si on clique sur une carte, on est redirigé vers la page de détails du logement.

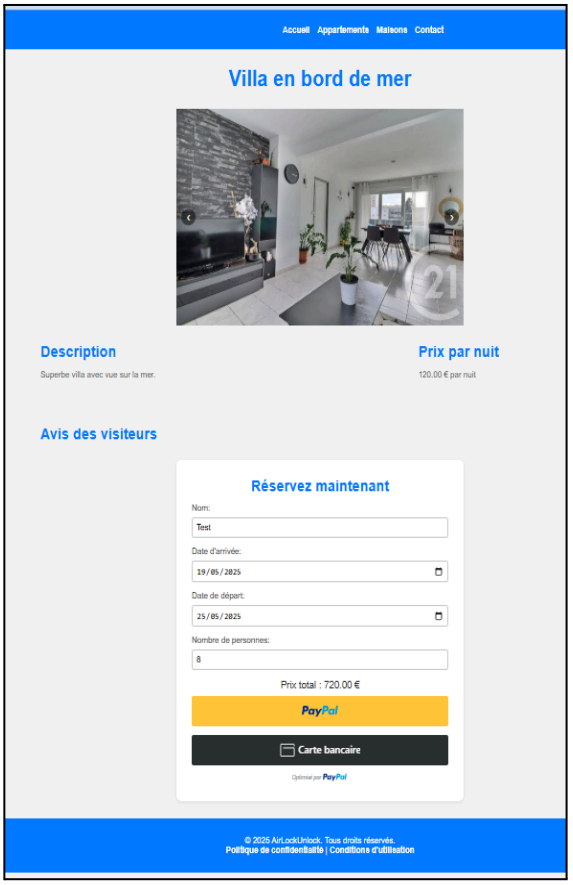
Ce système permet d'afficher les logements dynamiquement, sans avoir besoin de les coder un par un dans la page.

3 - Réservation et paiement

En cliquant sur un logement, l'utilisateur accède à une fiche complète.
Cette fiche contient une description détaillée et un formulaire de réservation.

Processus :

- L'utilisateur remplit ses dates de séjour.
- Le paiement est effectué via le SDK PayPal intégré.
- Une fois le paiement validé, les données sont envoyées à l'API pour enregistrer la réservation.



```
async function fetchPropertyDetails() {
    const id_bien = getIdFromUrl();
    if (!id_bien) {
        document.getElementById('property-title').innerText = 'Aucun bien sélectionné.';
        return;
    }

    try {
        const response = await fetch(`https://172.16.15.74:421/AirlockUnlock/bien/biens.php?
        if (!response.ok) throw new Error('Erreur lors de la récupération du bien.');

        const biens = await response.json();
        if (!Array.isArray(biens) || biens.length === 0) {
            document.getElementById('property-title').innerText = 'Bien introuvable.';
            return;
        }

        currentProperty = biens[0];
        pricePerNight = parseFloat(currentProperty.prix_par_nuit);
        displayProperty(currentProperty);
        loadImages(currentProperty);
        loadAmenities(currentProperty);
        loadReviews(currentProperty);
    } catch (error) {
        console.error('Erreur:', error);
        document.getElementById('property-title').innerText = 'Une erreur est survenue.';
    }
}
```

Chargement des détails d'un logement

Lorsque l'utilisateur arrive sur la page de détails d'un logement, le site récupère automatiquement l'identifiant du logement (id_bien) dans l'URL. Cet identifiant permet de faire une requête à l'API pour obtenir toutes les informations liées à ce bien.

Si l'API répond correctement, les détails du logement (comme le titre, le prix, les photos, les équipements ou encore les avis) sont affichés à l'aide de différentes fonctions, notamment displayProperty() pour les informations principales.

En cas d'erreur (logement introuvable, identifiant invalide ou problème avec l'API), un message d'erreur s'affiche à l'écran pour informer l'utilisateur.

Ce fonctionnement permet d'afficher dynamiquement les informations d'un logement et d'assurer une bonne expérience utilisateur.

```
    fetch('https://172.16.15.74:421/AirlockUnlock/bien/reserver.php', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
            'Authorization': 'Bearer ' + localStorage.getItem('token')
        },
        body: new URLSearchParams(reservationData)
    })
    .then(response => response.json())
    .then(data => {
        console.log('Success:', data);
        localStorage.setItem('token', data.token);
        alert('Réservation enregistrée.');
    })
    .catch((error) => {
        console.error('Error:', error);
        alert('Erreur lors de l\'enregistrement de la réservation.');
    });
}

document.addEventListener('DOMContentLoaded', async function () {
    await fetchPropertyDetails();
});
```

Enregistrement d'une réservation

Lorsqu'un utilisateur connecté remplit le formulaire de réservation et le valide, le site déclenche un processus permettant d'enregistrer sa demande auprès du serveur. Ce processus repose sur l'envoi d'une requête HTTP de type POST vers le fichier reserver.php de l'API.

Les informations de la réservation, regroupées dans un objet appelé reservationData, sont incluses dans le corps de cette requête. Pour garantir la sécurité de l'opération, le token d'authentification de l'utilisateur est également ajouté dans les en-têtes. Ce token permet de vérifier que la demande provient bien d'un utilisateur autorisé.

Si la réservation est acceptée par le serveur, une réponse positive est renvoyée. Le système en profite alors pour mettre à jour le token dans les Cookies, ce qui permet de maintenir la session de manière sécurisée. Un message de confirmation est ensuite affiché à l'écran afin d'informer l'utilisateur que sa réservation a bien été prise en compte.

En revanche, si une erreur survient — que ce soit à cause de données incorrectes, d'un token invalide ou d'un problème technique — un message d'erreur s'affiche à l'utilisateur pour l' avertir.

L'objectif de ce code est double : permettre à un utilisateur authentifié de réserver un logement facilement, et s'assurer que cette opération se déroule de façon sécurisée grâce à l'utilisation d'un token. Le tout vise à offrir une expérience utilisateur fluide, claire et fiable.

```
paypal.Buttons({
    createOrder: function(data, actions) {
        // Envoyer les données à l'API avant de créer l'ordre PayPal
        sendReservationToAPI();
        return actions.order.create({
            purchase_units: [{
                amount: {
                    value: totalPrice.toFixed(2) // Montant total calculé
                }
            }]
        });
    },
    onApprove: function(data, actions) {
        return actions.order.capture().then(function(details) {
            // Redirection vers PayPal
            window.location.href = 'https://www.paypal.com';
            alert('Transaction completed by ' + details.payer.name.given_name);
        });
    }
}).render('#paypal-button-container');
```

Chargement des détails d'un logement

Lorsque l'utilisateur accède à la page de détails d'un logement, le site déclenche automatiquement un processus destiné à afficher toutes les informations relatives à ce bien. Pour cela, l'identifiant du logement (id_bien) est d'abord extrait de l'URL.

À partir de cet identifiant, une requête est envoyée à l'API afin de récupérer les données complètes du logement correspondant. Si la réponse est positive, ces données sont ensuite affichées à l'écran. La fonction displayProperty() se charge d'afficher les informations principales, telles que le titre, la description ou le prix. D'autres fonctions sont également utilisées pour charger et afficher les photos du logement, les équipements disponibles ainsi que les avis laissés par les utilisateurs.

En cas de problème — si le logement n'existe pas, si l'identifiant est invalide ou si l'API ne répond pas correctement — un message d'erreur est affiché pour en informer l'utilisateur.

Ce système permet d'afficher dynamiquement les informations d'un logement à chaque chargement de page, tout en assurant une expérience fluide et fiable. Il garantit aussi la robustesse de l'application en prenant en compte les éventuelles erreurs liées aux données ou à la connexion avec l'API.

```
paypal.Buttons({
    createOrder: function(data, actions) {
        // Envoyer les données à l'API avant de créer l'ordre PayPal
        sendReservationToAPI();
        return actions.order.create({
            purchase_units: [{
                amount: {
                    value: totalPrice.toFixed(2) // Montant total calculé
                }
            }]
        });
    },
    onApprove: function(data, actions) {
        return actions.order.capture().then(function(details) {
            // Redirection vers PayPal
            window.location.href = 'https://www.paypal.com';
            alert('Transaction completed by ' + details.payer.name.given_name);
        });
    }
}).render('#paypal-button-container');
```

Paiement en ligne via PayPal

Ce code permet d'intégrer un système de paiement sécurisé avec PayPal, afin que l'utilisateur puisse régler sa réservation directement en ligne. Lorsqu'un utilisateur clique sur le bouton "Payer", plusieurs étapes se déclenchent automatiquement.

Tout d'abord, les informations liées à la réservation sont envoyées à l'API grâce à la fonction sendReservationToAPI(). Cela permet d'enregistrer la réservation avant même de procéder au paiement. Ensuite, une commande est créée auprès de PayPal, en précisant le montant total à régler (totalPrice).

Une fois que l'utilisateur valide le paiement via l'interface PayPal, la transaction est confirmée. Le site affiche alors un message de succès personnalisé, incluant le prénom de la personne ayant effectué le paiement. L'utilisateur peut ensuite être redirigé — ici vers la page de PayPal, mais cela peut être adapté selon les besoins du projet.

Ce système permet d'assurer un paiement sécurisé tout en automatisant l'ensemble du processus : de l'enregistrement de la réservation à la validation de la transaction. Il garantit également que la réservation est bien prise en compte avant que le paiement ne soit lancé, afin d'éviter tout problème de synchronisation ou de réservation non confirmée.

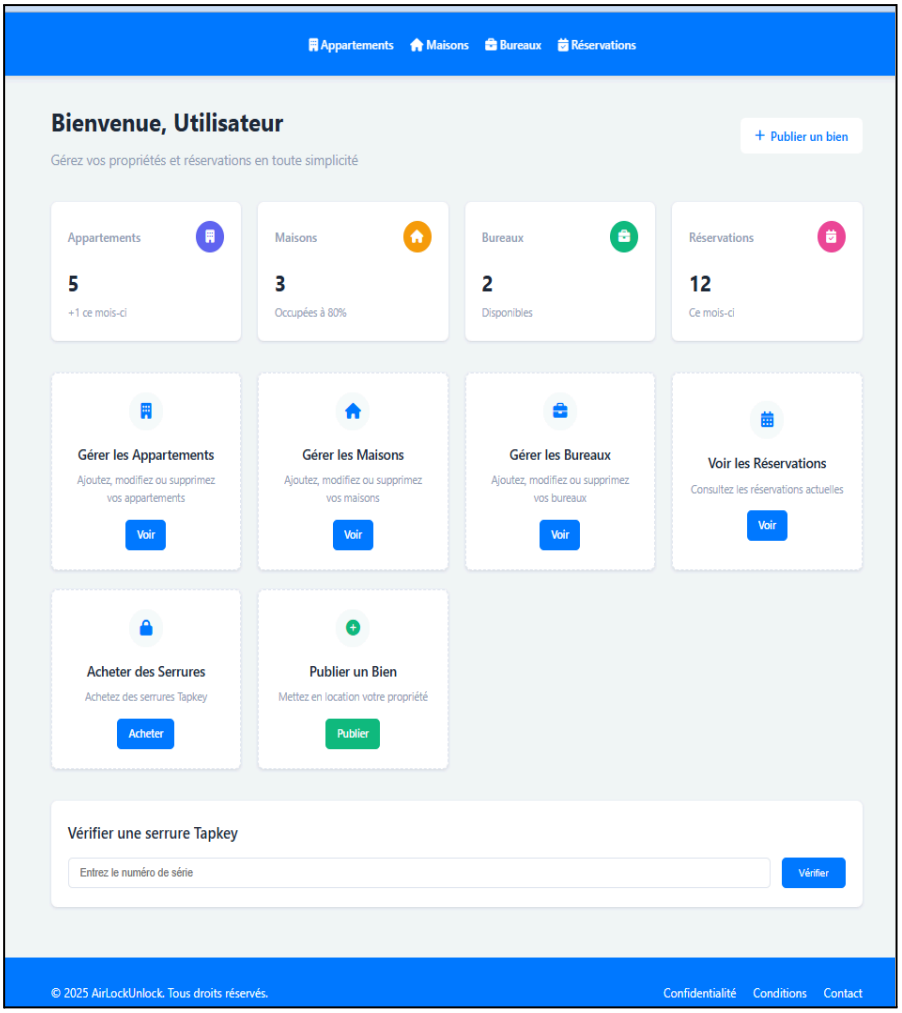
4 - Tableau de bord propriétaire

Cette interface est réservée aux propriétaires connectés.

Fonctionnalités disponibles :

- Affichage des biens publiés.
- Ajout d'un nouveau bien (titre, prix, localisation, photos…).
- Modification ou suppression d'un logement existant.
- Association d'une serrure Tapkey (optionnelle).
- Vérification de la propriété de la serrure via appel à l'API Tapkey.

Les propriétaires peuvent également accéder à la page client pour effectuer des réservations.



```
    <script>
        document.addEventListener('DOMContentLoaded', function() {
            // Message de bienvenue par défaut
            document.getElementById('welcome-message').textContent = 'Bienvenue, Utilisateur';
        });

        // Vérification du numéro de série
        document.querySelector('.verify-btn').addEventListener('click', function() {
            const serialNumber = document.querySelector('.verify-input').value;
            if (serialNumber.trim() === '') {
                alert('Veuillez entrer un numéro de série');
                return;
            }

            // Simulation de vérification
            alert(`Numéro de série ${serialNumber} vérifié avec succès!`);
        });

        // Animation au survol des cartes
        document.querySelectorAll('.stat-card').forEach(card => {
            card.addEventListener('mouseenter', function() {
                this.style.transform = 'translateY(-5px)';
                this.style.boxShadow = '0 10px 15px -3px rgba(0, 0, 0, 0.1)';
            });

            card.addEventListener('mouseleave', function() {
                this.style.transform = '';
                this.style.boxShadow = '';
            });
        });
    </script>
</body>
```

Code JavaScript du tableau de bord (dashboard)

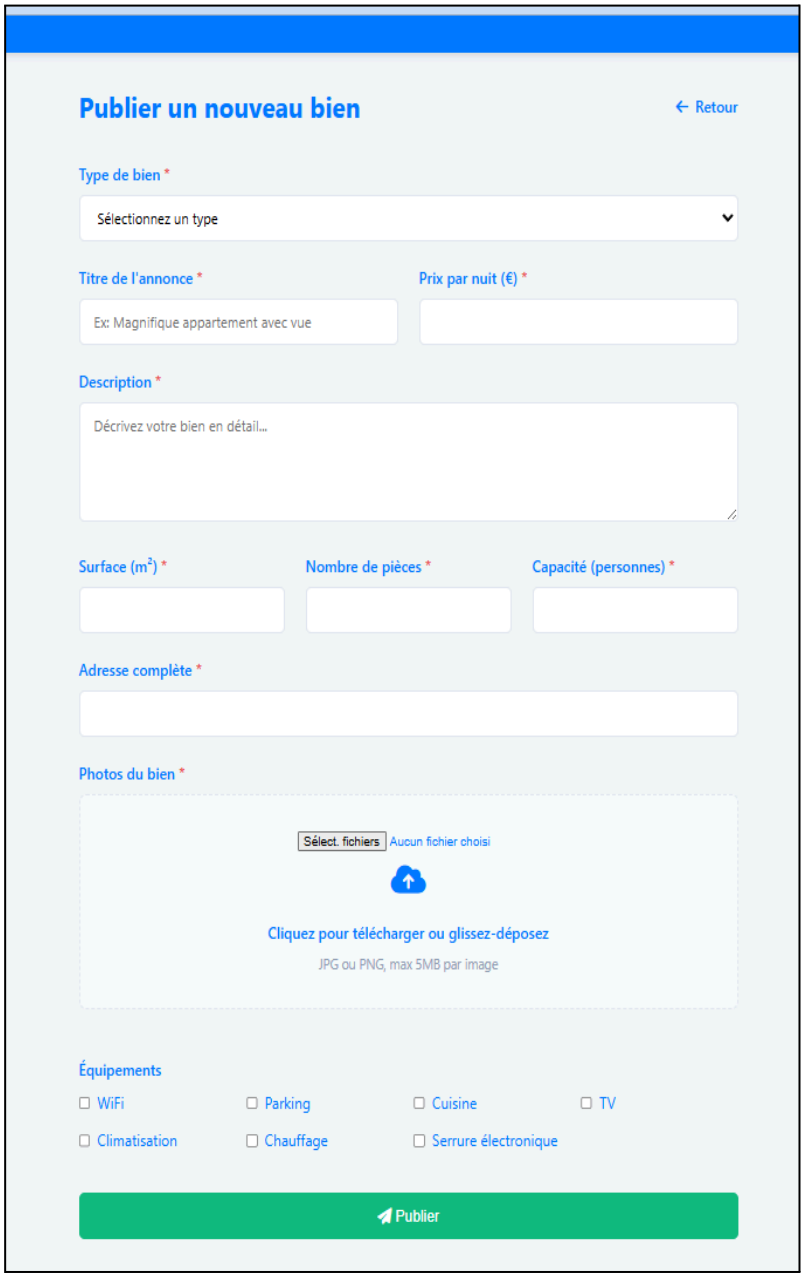
Ce code permet d'améliorer l'interactivité et l'expérience utilisateur sur le tableau de bord, une fois l'utilisateur connecté. Il introduit plusieurs fonctionnalités à la fois pratiques et visuelles.

Tout d'abord, dès que la page est chargée, un message de bienvenue par défaut s'affiche en haut de l'interface, indiquant "Bienvenue, Utilisateur". Ce message peut être personnalisé ultérieurement pour refléter le prénom ou l'identité de l'utilisateur connecté.

Ensuite, une fonctionnalité permet de vérifier un numéro de série, par exemple pour une clé Tapkey. Lorsqu'un utilisateur clique sur le bouton de vérification, le code vérifie si un numéro a été saisi. Si ce n'est pas le cas, une alerte s'affiche pour inviter l'utilisateur à entrer un numéro. Dans le cas contraire, une alerte simule une confirmation de vérification réussie.

Enfin, une animation visuelle est appliquée aux cartes de statistiques du tableau de bord. Lorsqu'on passe la souris sur une carte, celle-ci se soulève légèrement (grâce à un effet de translation verticale), et une ombre apparaît pour créer une impression de relief et de dynamisme. Lorsque la souris quitte la carte, l'effet visuel disparaît et la carte reprend son apparence initiale.

L'objectif de ce code est de rendre le tableau de bord plus interactif, plus accueillant et plus intuitif. Les animations et les alertes offrent un retour immédiat à l'utilisateur, renforçant ainsi la fluidité et l'ergonomie de l'interface.


Publier un nouveau bien
← Retour
Type de bien *
Sélectionnez un type
Titre de l'annonce *
Ex: Magnifique appartement avec vue
Prix par nuit (€) *
Description *
Décrivez votre bien en détail...
Surface (m²) *
Nombre de pièces *
Capacité (personnes) *
Adresse complète *
Photos du bien *
Sélect. fichiers
Aucun fichier choisi
Cliquez pour télécharger ou glissez-déposez
JPG ou PNG, max 5MB par image
Équipements
WiFi
Parking
Cuisine
TV
Climatisation
Chauffage
Serrure électronique
Publier

```
<script>
    // Gestion de l'affichage du champ Tapkey
    document.getElementById('serrure_electronique').addEventListener('change', function() {
        const tapkeyField = document.getElementById('tapkeyField');
        if (this.checked) {
            tapkeyField.style.display = 'block';
        } else {
            tapkeyField.style.display = 'none';
            document.getElementById('numero_serie_tapkey').value = ''; // Réinitialise la valeur
            document.getElementById('numero_serie_tapkey').classList.remove('error'); // Enlève la classe d
            document.getElementById('tapkeyError').style.display = 'none'; // Masque le message d'erreur
        }
    });

    document.getElementById('publishForm').addEventListener('submit', async function(e) {
        e.preventDefault(); // Empêche la soumission classique du formulaire

        const formElement = e.target;
        const formData = new FormData(formElement);

        try {
            // Affiche toutes les données avant l'envoi (debug)
            for (let [key, value] of formData.entries()) {
                console.log(key, value);
            }

            const response = await fetch('https://172.16.15.74:421/AirlockUnlock/bien/publier-bien.php', {
                method: 'POST',
                body: formData,
            });

            const contentType = response.headers.get("content-type");

            if (!response.ok) {
                throw new Error(`Erreur HTTP : ${response.status}`);
            }
```

```
            if (contentType && contentType.includes("application/json")) {
                const result = await response.json();
                if (result.error) {
                    document.getElementById('numero_serie_tapkey').classList.add('error'); // A
                    document.getElementById('tapkeyError').textContent = result.error;
                    document.getElementById('tapkeyError').style.display = 'block';
                } else {
                    document.getElementById('numero_serie_tapkey').classList.remove('error');
                    document.getElementById('tapkeyError').style.display = 'none'; // Masque le
                    alert(result.message || "Bien publié avec succès !");
                }
            } else {
                const text = await response.text();
                console.warn("Réponse non JSON :", text);
                alert("Réponse inattendue : " + text);
            }

        } catch (error) {
            console.error('Erreur lors de la requête API:', error);
            alert('Erreur lors de la publication : ' + error.message);
        }
    });
```

Ajout d’un bien avec option Tapkey

Ce code permet à un propriétaire de publier un bien via le formulaire du tableau de bord, avec une option pour ajouter une serrure connectée Tapkey.

Si l’utilisateur coche la case "serrure électronique", un champ apparaît pour saisir le numéro de série Tapkey. Si la case est décochée, ce champ est masqué et réinitialisé.

Lors de l’envoi du formulaire, les données sont transmises à l’API (publier-bien.php). En cas d’erreur liée à Tapkey (ex : numéro invalide), un message s’affiche sous le champ. Sinon, le message d’erreur disparaît et une confirmation apparaît : « Bien publié avec succès ! ».

Objectif : simplifier l’ajout de bien, gérer dynamiquement l’option Tapkey, et fournir un retour clair à l’utilisateur.

5 - Sécurité et configuration

Le site met en place plusieurs mesures de sécurité :

- Authentification avec gestion des rôles.
- Appels API sécurisés en HTTPS.
- Protection des routes sensibles côté front-end et back-end.
- Validation des champs côté client et serveur.
- Gestion des tokens via **cookies sécurisés**.

Aucune configuration manuelle de l’adresse API ou d’un appareil n’est requise côté site web, à la différence de l’application mobile

Problème rencontré

Lors du développement du site web, le principal problème a été la communication avec l'API. Le site devait s'appuyer uniquement sur l'API pour toutes les données (affichage des biens, réservation, etc.), ce qui demandait une intégration parfaite.

Plusieurs difficultés sont survenues :

Certaines routes de l'API n'étaient pas toujours bien documentées ou leur fonctionnement pas clair.

Le format des données échangées ne correspondait pas toujours à ce que le site envoyait ou recevait, ce qui a nécessité plusieurs corrections dans le code.

Pour effectuer une réservation, l'utilisateur devait être connecté, et la communication sécurisée devait se faire via un token stocké dans un cookie.

Cependant, le token contenu dans le cookie n'était pas toujours reconnu ou envoyé correctement avec les requêtes, ce qui provoque des erreurs d'autorisation (erreurs 401).

Cette situation a bloqué la réservation et fait perdre beaucoup de temps à diagnostiquer et corriger.

Pour résoudre cela :

Un travail a été fait pour s'assurer que le cookie du token soit bien présent et transmis automatiquement lors des requêtes fetch.

Des vérifications ont été effectuées pour gérer la sécurité du cookie (HttpOnly, SameSite).

Parallèlement, des modifications ont été demandées côté API pour corriger la gestion des tokens dans les cookies et garantir une bonne reconnaissance côté serveur.

Conséquences :

Perte de temps importante à cause de problèmes de synchronisation entre site et API.

Blocage temporaire de la fonction réservation, essentielle au projet.

L'IoT (Adel Aichi)

Introduction

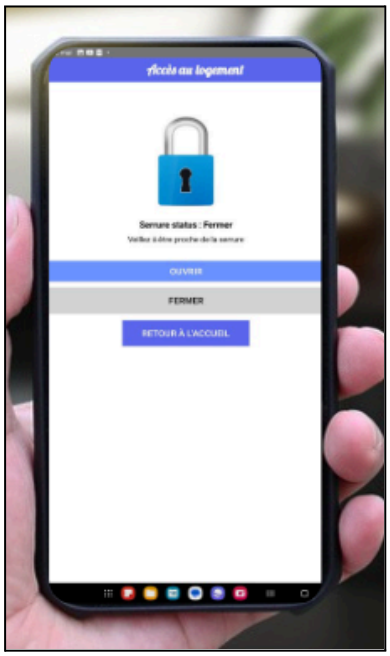
Lors de ce projet, j'ai été responsable de la partie IoT du projet, dont l'objectif était de permettre **le contrôle d'une serrure connectée via un microcontrôleur ESP32**. Cette serrure devait pouvoir être ouverte ou fermée à distance à partir de l'application mobile, en fonction des réservations des utilisateurs. Mon rôle principal consistait à programmer l'ESP32 pour qu'il reçoive des commandes HTTP, sécurisées, et les traduise en actions mécaniques sur la serrure via un servo-moteur.

Procédure

La serrure connectée doit permettre l'ouverture et la fermeture à distance d'un logement, de manière sécurisée, automatique et sans intervention physique. Intégrée à un système de réservation en ligne, cette serrure s'active uniquement pendant les créneaux horaires validés, ce qui permet :

- Un accès autonome pour les utilisateurs, sans remise de clé physique.
- Un gain de temps pour les gestionnaires de logements.
- Un niveau de sécurité renforcé, grâce à un contrôle centralisé via une API et une authentification par token.

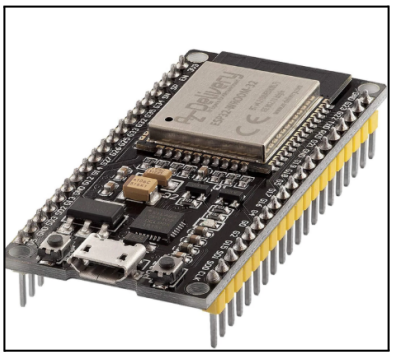
Ce système s'inscrit dans une logique de domotique intelligente et de gestion automatisée des accès, adaptée aux locations.



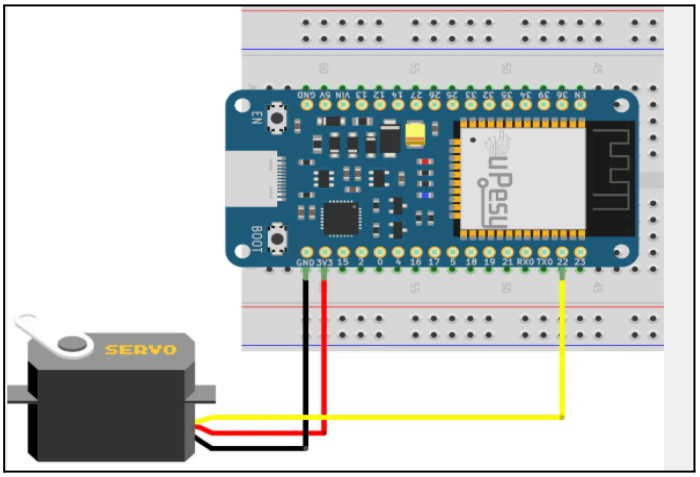
Mise en place de la serrure connectée

La serrure connectée est composée de deux éléments principaux : un **ESP 32** et un **servomoteur**.

L'ESP 32 est une carte électronique qui sert de cerveau au système. Elle permet de se connecter au réseau Wi-Fi et de recevoir les commandes envoyées depuis l'application. C'est un microcontrôleur puissant, équipé de capacités sans fil (Wi-Fi, Bluetooth), idéal pour les projets connectés.



Le **servomoteur** est un moteur capable de tourner précisément à un angle défini. Contrairement à un moteur classique qui tourne en continu, un servomoteur peut se positionner exactement à 0° et 180°. Ici, il simule le mouvement d'une serrure mécanique en tournant la clé ou le mécanisme d'ouverture.



L'ESP 32 alimente et contrôle le servomoteur en envoyant des signaux électriques adaptés, avec une tension généralement autour de 5 volts.

La partie IoT repose sur l'utilisation d'un ESP32 connecté au réseau Wi-Fi, qui contrôle un servo-moteur simulant l'ouverture et la fermeture d'une serrure. Ce système communique avec le reste de la plateforme via des requêtes HTTP sécurisées, envoyées depuis l'application mobile.

1 - Connexion au Wi-Fi

L'ESP 32 est configuré pour se connecter automatiquement au réseau Wi-Fi défini (SSID et mot de passe). Lors de l'initialisation (setup()), le module établit une connexion au réseau et affiche l'adresse IP locale sur le moniteur série.

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("Wi-Fi connecté !");
Serial.println(WiFi.localIP());
```

```
Connexion au Wi-Fi.
Wi-Fi connecté !
Adresse IP : 192.168.137.167
```

2 - Sécurisation des commandes HTTP

Chaque commande envoyée à l'ESP 32 passe par un point d'entrée HTTP : /on, /off. Ces routes ne sont accessibles que si le client fournit un token d'authentification valide, dans l'en-tête HTTP Authorization.

```
if (!server.hasHeader("Authorization")) return false;
String authHeader = server.header("Authorization");
return authHeader == "Bearer " + String(TOKEN);
```

Cela empêche toute personne non autorisée d'envoyer des commandes à la serrure.

3 - Commandes disponibles

Trois commandes principales sont disponibles pour contrôler l'ouverture de la serrure via le servo :

- /on tourne le servo à 180° → ouverture complète.
- /off : retour à 0° → fermeture.

Chaque commande attache temporairement le servo, effectue le mouvement, puis le détache.

```
monServo.attach(servoPin, 500, 2500);
monServo.write(positionOn);
delay(500);
monServo.detach();
```

```
WebServer server(80);
Servo monServo;
const int servoPin = 13;
const int positionOn = 180;
const int positionOff = 0;
const int positionMid = 90;
```

```
Commande reçue : ON (180°)
```

```
Commande reçue : OFF (0°)
```

Voici dans le moniteur séries les commandes envoyées via l'application, ce qui montre le bon fonctionnement entre les deux.

4 - Serveur Web intégré à l'ESP32

L'ESP32 lance un serveur HTTP local écoutant sur le port 80. Lorsqu'une requête valide est reçue, une réponse 200 OK est envoyée, et l'action est exécutée.

```
server.on("/on", HTTP_GET, []() {
    if (!isAuthorized()) {
        server.send(401, "text/plain", "Unauthorized");
        return;
    }
    monServo.write(positionOn);
    server.send(200, "text/plain", "Servo tourné à 180°");
});
```

Moniteur serie réponse :

```
Connexion au Wi-Fi.
Wi-Fi connecté !
Adresse IP : 192.168.137.212
Serveur HTTP lancé.
Commande reçue : ON (180°)
```

L'application Android (Bilal Taoufik)

Introduction

L'application android permet aux utilisateurs de **se connecter** à leur compte, de **consulter** leurs réservations en cours ou à venir, et **d'interagir** avec une serrure connectée, **uniquement pendant la période de réservation.**

Son objectif principal est de rendre l'accès au logement simple, sécurisé et autonome.

Procédure

L'application se lance via un écran de **connexion sécurisé**. Une fois authentifié, l'utilisateur accède à sa page d'accueil qui affiche ses **réservations**.

Si une réservation est **en cours**, il peut la sélectionner et accéder à une page instructions qui lui permet de :

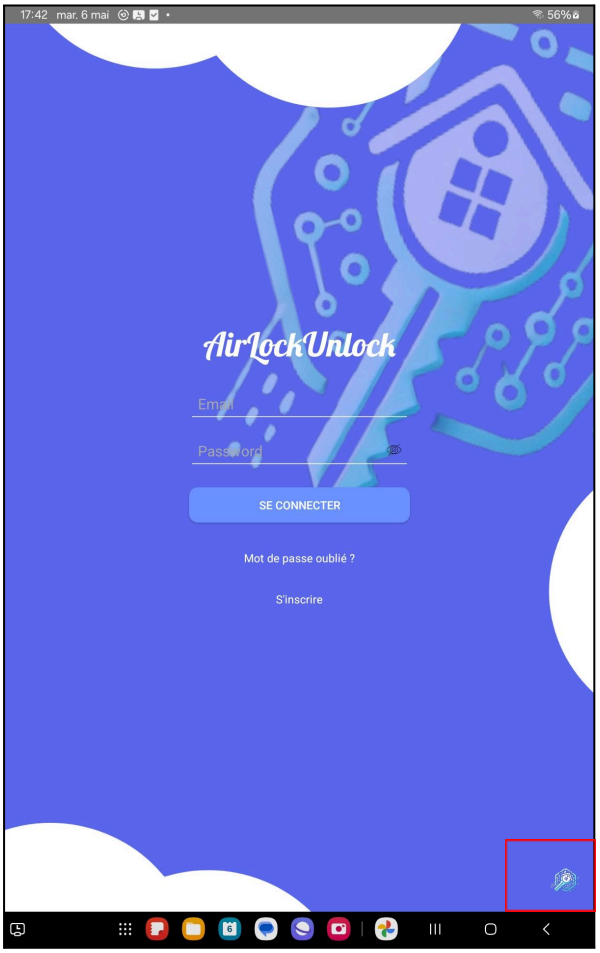
- Ouvrir ou fermer la **serrure connectée**.
- Ces actions ne sont disponibles que pendant la période de réservation.
- Les commandes sont envoyées via HTTP au microcontrôleur (ESP32)

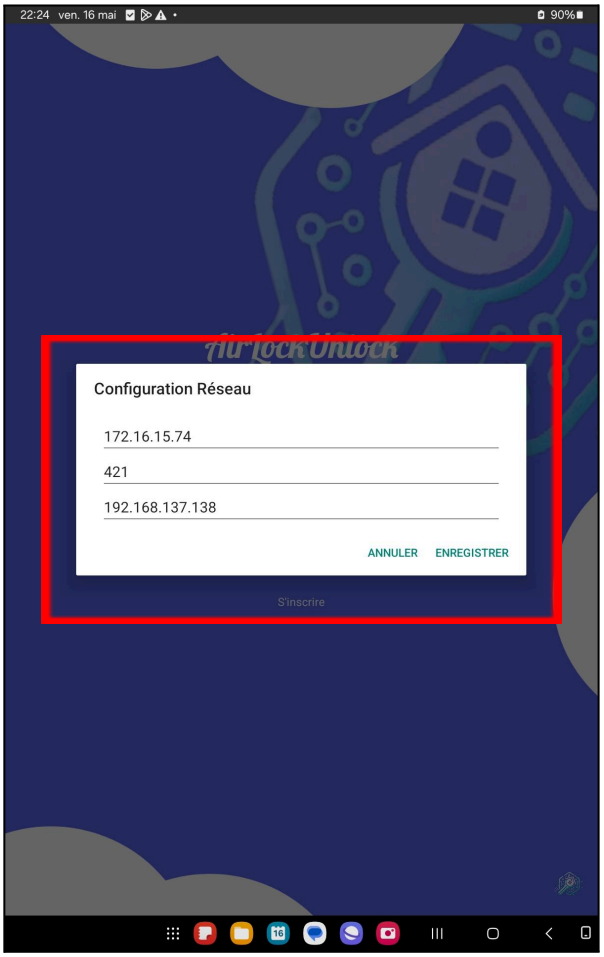
L'interface est conçue pour être **simple, intuitive et efficace**, avec une gestion du temps automatisée pour bloquer l'accès en dehors des créneaux valide.

Mise en place de l'application

Une section **cachée** dans la page de connexion de l'application permet de configurer :

- l'adresse IP et le port de L'API,
- l'adresse IP de l'ESP32





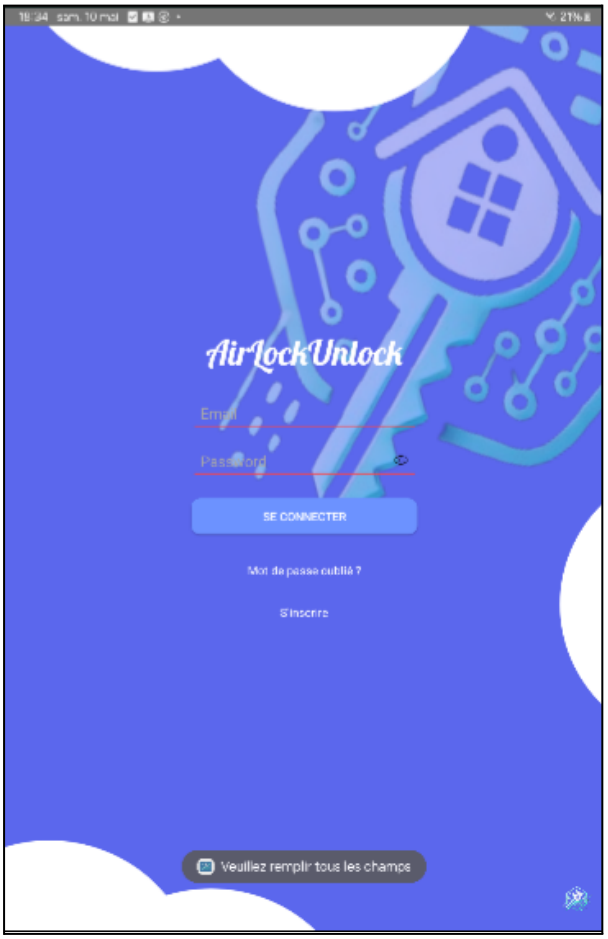
Pour accéder à cette configuration, il suffit de **cliquer trois fois** sur le logo situé en bas à droite de l'écran. Un pop-up s'affiche alors avec trois champs à compléter : l'adresse IP de l'API, son port et l'adresse IP de l'ESP32.

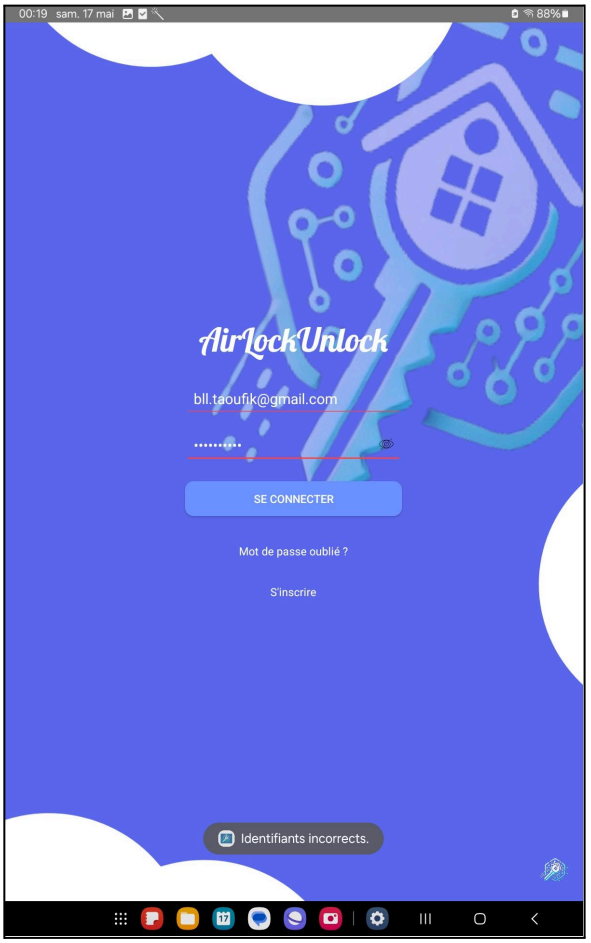
Cette configuration manuelle permet de rendre l'application adaptable à différents environnements réseau et assure une bonne communication entre les différents composants. Une fois la configuration faite, place à la connexion.

```
D  Configuration enregistrée : IP=192.168.1.160, Port=421, ESP=192.168.138.1
I  show: caller = com.example.projetairlocunlock.LoginActivity.lambda$showCon
```

1 - Connexion

Le client accède à une page de connexion avec un champ pour l'e-mail et un autre pour le mot de passe. La connexion est réservée aux clients ayant un compte, l'inscription se fait via le site web. Si l'un des champs est vide ou mal rempli, les bordures deviennent rouges pour signaler une erreur.





Quand l'utilisateur saisit correctement ses identifiants et valide, l'application envoie une requête à l'API. En cas de succès, le serveur renvoie un token d'authentification. Ce token est alors stocké localement dans l'application, précisément dans les SharedPreferences, afin d'être conservé de façon sécurisée. Le code utilisé par exemple :

```
SharedPreferences sharedPreferences = getSharedPreferences( name: "MyAppPrefs", MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putString("token", token);
editor.apply();
```

Ce token est ensuite utilisé dans les requêtes suivantes pour authentifier l'utilisateur et lui permettre d'accéder aux fonctionnalités sécurisées de l'application.

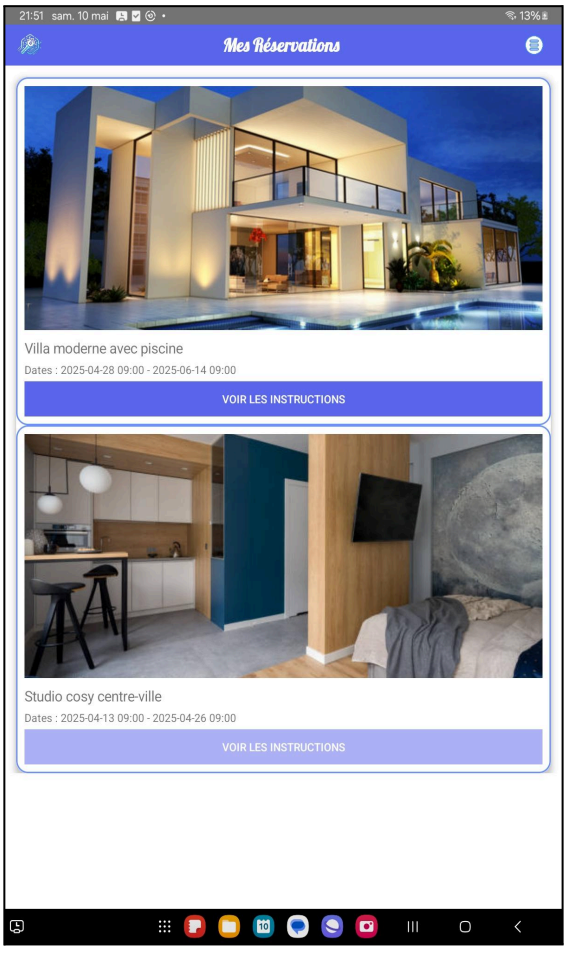
Réponse JSON :


```
{
    "status": "success",
    "message": "Connexion réussie.",
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
        eyJpZF9jbGllbnQiOjIsIm5vbSI6IkJpbGFsIFRhb3VmaWsiLCJyb2xlIjoiY2xpZW50IiwiaWF0IjoxNzQ3NTgwNTQwLCJ
        leHAiOjE3NDc1ODQxNDB9.lzXkis9AtMqq7zDBjxLKJNGMTbUDWNavtesb6xlCrDU",
    "client_id": 2,
    "nom": "Bilal Taoufik",
    "email": "bilal@gmail.com"
}
```


2 - Affichage des réservations

Une fois les identifiants valides, l'utilisateur est redirigé automatiquement vers la page d'accueil. Elle leur permet ensuite d'accéder à leurs réservations. Cette page affiche en temps réel toutes ses réservations, récupérées depuis l'API AirlockUnlock, avec la date d'arrivée, date de départ et le titre, ainsi que la photo correspondante. Bien sûr, il est obligatoire que l**e statut soit confirmé**.

En arrivant sur la page d'accueil, on peut voir les réservations liées au compte connecté. Deux exemples sont affichés : **une réservation active**, avec la possibilité d'ouvrir la serrure, et une autre dont la date est dépassée, donc **non accessible**.

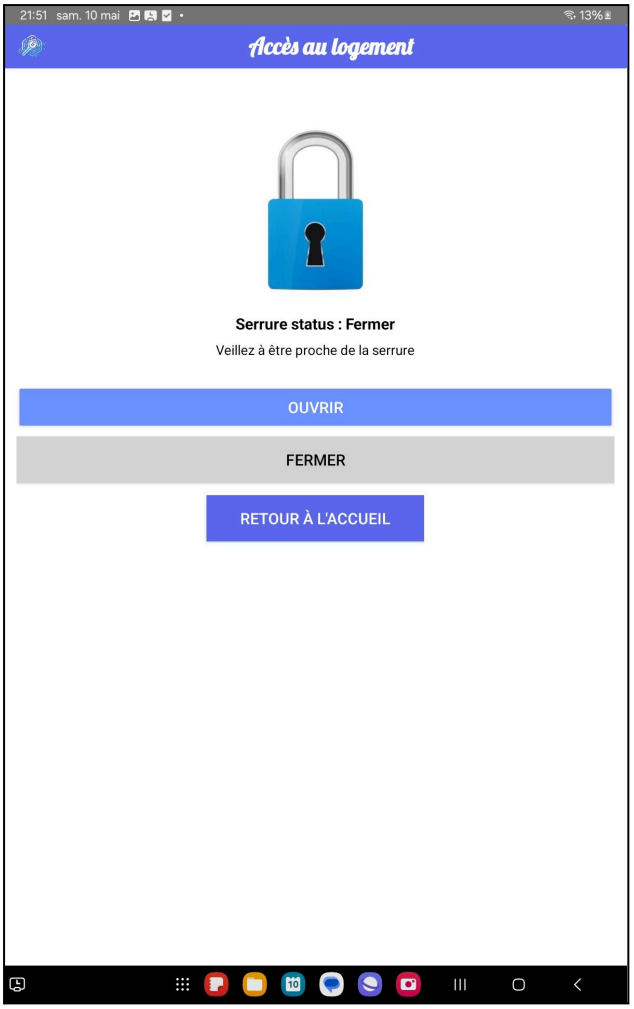


Réponse JSON :

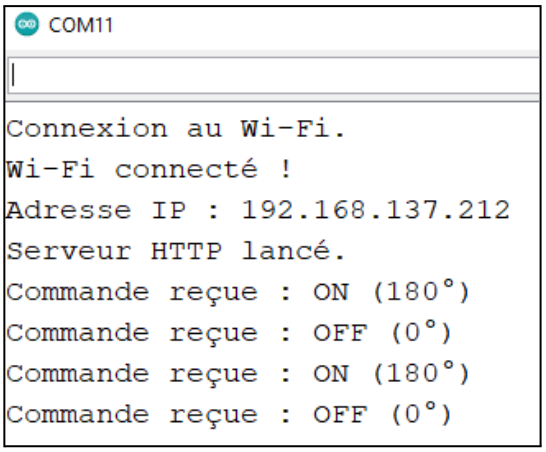

```
"status": "success",
"reservations": [
    {
        "id_reservation": 4,
        "date_arrivee": "2025-04-28",
        "date_depart": "2025-06-14",
        "nombre_personnes": 10,
        "statut": "confirmée",
        "titre": "Villa moderne avec piscine",
        "photo_url": "https://192.168.1.160:421/A
    },
    {
        "id_reservation": 3,
        "date_arrivee": "2025-04-13",
        "date_depart": "2025-04-26",
        "nombre_personnes": 2,
        "statut": "confirmée",
        "titre": "Studio cosy centre-ville",
        "photo_url": "https://192.168.1.160:421/A
    }
]
```

3 – Sélection d’une réservation active

Lorsque l’une des réservations est active (selon la date et l’heure), l’utilisateur peut la sélectionner pour accéder à l’interface d'instruction. Cette page propose deux boutons : **ouvrir** et **fermer** la serrure. Chaque action déclenche l’envoi d’une requête HTTP vers l'ESP 32, accompagnée d’un token d’authentification placé dans les en-têtes, et utilisant l’adresse IP configurée dans les paramètres cachés de l’application, vue au début. Ce mécanisme permet un contrôle sécurisé et efficace de la serrure pendant la période de réservation.



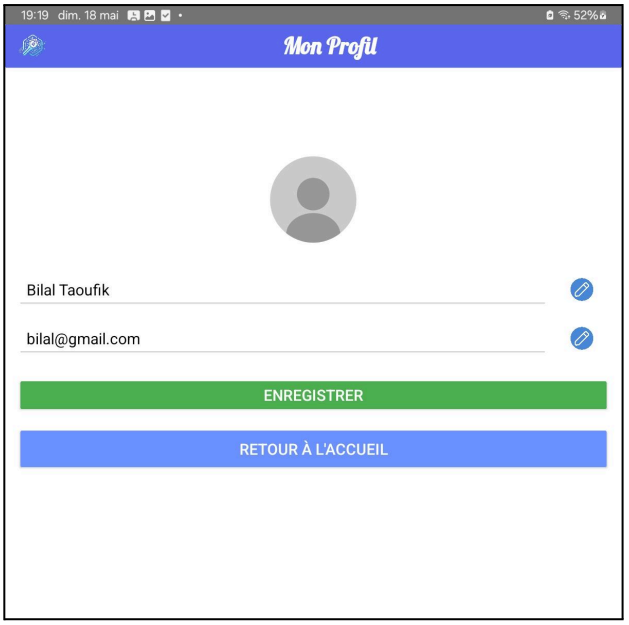
Commande reçus dans le moniteur série, lors du clique sur Ouvrir/Fermer :



4 – Profils

Sur la page d'accueil, un bouton hamburger est placé en haut. En cliquant dessus, un menu déroulant s'ouvre, proposant des options comme accéder au profil de l'utilisateur ou se déconnecter.

Cette page correspond au profil de l'utilisateur dans l'application. Elle récupère et affiche les informations personnelles comme le nom et l'e-mail en les chargeant depuis l'API via une requête sécurisée avec le token d'authentification stocké dans l'application.



5 – Sécurité : Token

Toutes **les actions sensibles** de l'application (comme accéder au profil, consulter des données de réservations) sont sécurisées grâce à un token d'authentification. Ce token est obtenu lors de la connexion et **stocké localement dans l'application (dans les SharedPreferences)** comme vu précédemment. Il est **indispensable** pour récupérer les informations d'un utilisateur depuis l'API. Sans lui, l'accès est refusé. Ce token a une **durée de validité d'environ 1 heure**, après quoi l'utilisateur devra se reconnecter pour en obtenir un nouveau.

6 – Eléments indispensables pour le bon fonctionnement de l'application

L'application nécessite certaines autorisations pour fonctionner correctement. Elles doivent être déclarées dans le fichier **AndroidManifest.xml**.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Rôle des permissions :

- **INTERNET** : permet d'**effectuer des requêtes HTTP/HTTPS** vers le serveur.
- **ACCESS_NETWORK_STATE** : permet à l'application de **vérifier l'état de la connexion réseau**.

Étant donné que le serveur utilise un **certificat SSL autosigné**, Android bloque par défaut les connexions HTTPS non vérifiées. Il faut donc désactiver temporairement la validation SSL pour les besoins du développement.

La fonction :

```
private void disableSSLCertificateChecking() {  1 usage
    try {
        TrustManager[] trustAllCerts = new TrustManager[] {
                new X509TrustManager() {
                    public void checkClientTrusted(X509Certificate[] chain, String authType) {}
                    public void checkServerTrusted(X509Certificate[] chain, String authType) {}
                    public X509Certificate[] getAcceptedIssuers() { return new X509Certificate[0]; }
                }
        };

        SSLContext sc = SSLContext.getInstance( protocol: "SSL");
        sc.init( km: null, trustAllCerts, new SecureRandom());
        HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
        HttpsURLConnection.setDefaultHostnameVerifier(( String hostname,  SSLSession session) -> true);
        Log.d(TAG,  msg: "SSL désactivé avec succès");
```

Problème rencontré

Au cours du développement, le principal problème a été la communication entre l'application Android et l'ESP 32.
Initialement, le but était d'utiliser le Bluetooth pour établir la connexion. Cependant, plusieurs obstacles techniques ont rendu cette solution non fonctionnelle :

- Difficultés à établir une connexion stable entre l'ESP 32 et l'application Android via bluetooth.
- Problèmes de compatibilité entre les versions Android récentes et la gestion des permissions Bluetooth.

Face à ces contraintes, la communication a été basculée vers le Wi-Fi via des requêtes HTTP envoyées directement à l'ESP 32.

Cela a impliqué de modifier l'architecture prévue, notamment en intégrant une configuration dynamique de l'adresse IP de l'ESP 32 dans l'application.

Suite à ces modifications, nous avons de nouveau avantages et inconvénients :

Avantages du Wi-Fi :

- Portée plus large, ce qui permet à l'utilisateur d'interagir avec la serrure sans être collé à l'ESP32.
- Communication plus fiable et plus rapide, surtout pour l'envoi de commandes critiques comme l'ouverture ou la fermeture.

Inconvénients :

- Nécessite une connexion réseau locale ou un point d'accès Wi-Fi entre l'ESP 32 et le
- smartphone.
- Risque de déconnexion si l'environnement réseau est instable.
- Moins sécurisé si le Wi-Fi n'est pas bien protégé.

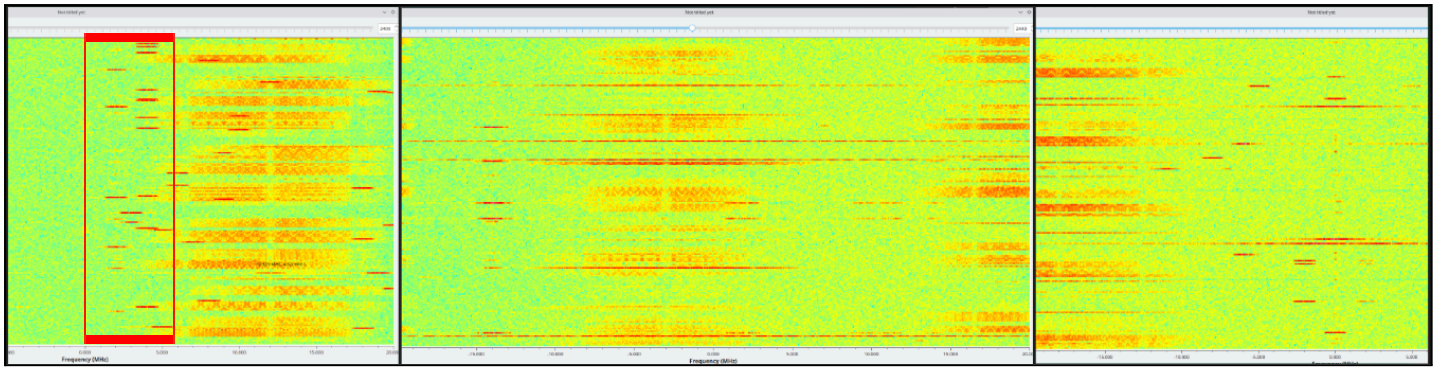
Malgré ce changement, l'application a pu conserver toutes ses fonctionnalités principales, assurant une communication fiable avec la serrure et répondant ainsi aux objectifs du projet.

VI - Physique

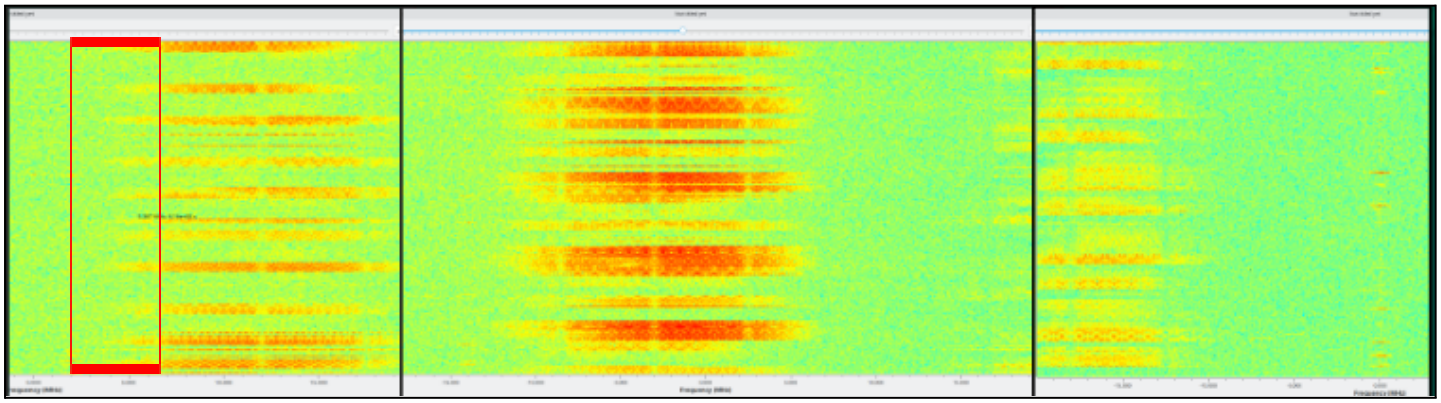
Analyse spectrale de la bande 2.4 GHz – Wi-Fi/Bluetooth

L'analyse que l'on a fait repose sur des mesures réalisées via GNU Radio, un outil open source de traitement du signal, permettant la visualisation du spectre radiofréquence en temps réel. Elle s'intéresse particulièrement à la bande 2.4 GHz, partagée par plusieurs technologies sur lesquelles on a travaillé durant ce projet, notamment le Wi-Fi et le Bluetooth.

Avec Bluetooth :



Sans Bluetooth :



Le spectrogramme capturé ici est un **waterfall**, où le temps évolue de bas en haut. L'axe horizontal représente la fréquence (en MHz), et la couleur indique l'intensité du signal (rouge = forte intensité). La bande observée s'étend de 2400 MHz à 2480 MHz.

Signaux Wi-Fi

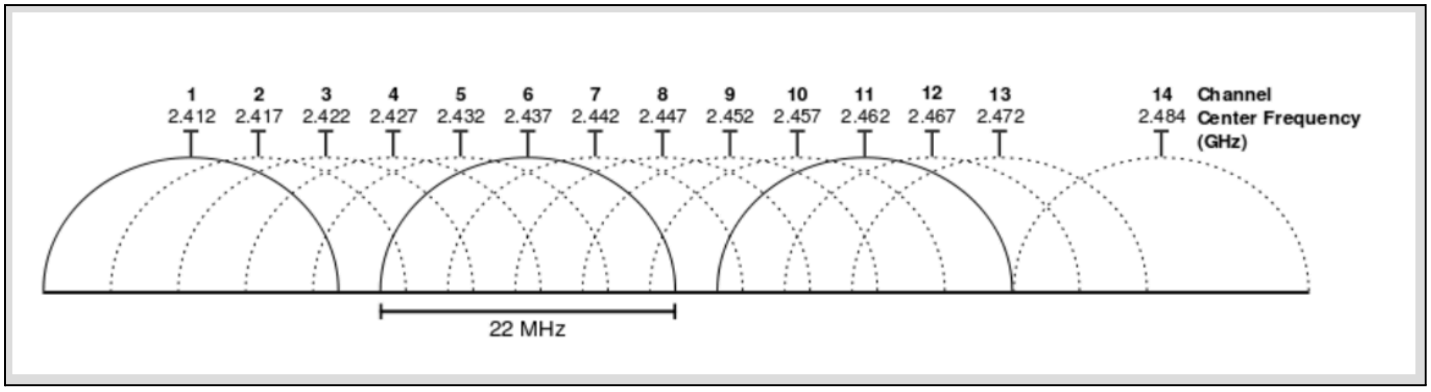
Des blocs rouges horizontaux, larges et persistants, sont visibles aux fréquences centrales : 2412 MHz (canal 1) - 2437 MHz (canal 6) - 2462 MHz (canal 11).

Ces canaux sont les plus couramment utilisés car ils ne se chevauchent pas. Le Wi-Fi utilise typiquement des canaux de 20 MHz de large, ce qui donne aux blocs une forme stable et bien définie dans le spectre.

Signaux Bluetooth

Des lignes fines, discontinues et réparties aléatoirement apparaissent à travers toute la bande. Elles correspondent à des transmissions Bluetooth, qui utilisent une technique appelée **saut de fréquence (frequency hopping)**.

Schéma des canaux Wifi 2.4 Ghz :



L'analyse spectrale de la bande 2.4 GHz (2400 à 2480 MHz) met en évidence la coexistence des technologies sur lequel on a travaillé durant ce projet. Le Wi-Fi utilise des canaux fixes et larges, occupant durablement certaines portions du spectre, tandis que le Bluetooth adopte une approche dynamique basée sur le saut de fréquence. Malgré des mécanismes conçus pour limiter les interférences, des perturbations peuvent survenir, en particulier dans les environnements denses ou critiques. Cela souligne l'importance d'une gestion rigoureuse du spectre pour garantir la fiabilité, le débit et la faible latence des communications.

VII - Bilan de Projet

Ce projet nous a permis de répondre efficacement à la problématique initiale en concevant une solution complète et fonctionnelle permettant à un utilisateur de gérer ses réservations et d'interagir avec un système connecté via une application mobile. Mais au-delà de l'objectif technique, ce projet a été **l'occasion d'explorer un large éventail de domaines**.

Nous avons en effet travaillé sur **l'ensemble de la chaîne de développement** :

- **Développement mobile** avec une application Android complète, intégrant une interface utilisateur, la gestion de sessions, les permissions, et la communication réseau.

- **Développement web** aussi bien en **frontend** qu'en **backend**, avec la création d'APIs, l'affichage dynamique de données, la gestion de bases de données et la mise en place de l'authentification.

- **IoT** grâce à l'intégration d'un **ESP32**, qui assure la liaison physique avec le dispositif de verrouillage, et la gestion réseau locale.

- **Cybersécurité**, avec la mise en place d'une authentification sécurisée via token, la gestion des connexions HTTPS,

Ce projet nous a permis de **travailler en équipe de manière efficace**, chacun apportant ses compétences tout en apprenant des autres. Cela nous a donné la possibilité de **toucher à chaque aspect**.

En résumé, ce projet a parfaitement synthétisé les enseignements de nos **deux années de formation**, en mobilisant des compétences techniques variées et complémentaires, dans un cadre concret et intéressant.