Programmation de mobiles

Professeur Encadrant : Katia Jaffrès-Runser

Réalisé par :
GHALLABI Sidi Mohamed Raid
NACERE Mohamed

Sciences du Numérique
Deuxième année - Parcours Réseaux
27 mai 2025

Table des Matières

1 Introduction

L'objectif du projet est de concevoir une application Android permettant le contrôle d'une maison connectée via une interface graphique dynamique (ScrollView). Le projet consiste à faire communiquer deux téléphones via Bluetooth : l'un agit comme client (télécommande), l'autre comme serveur (passerelle vers une API REST). Le client peut ainsi envoyer des commandes (ON/OFF d'un appareil) et recevoir les mises à jour du serveur.

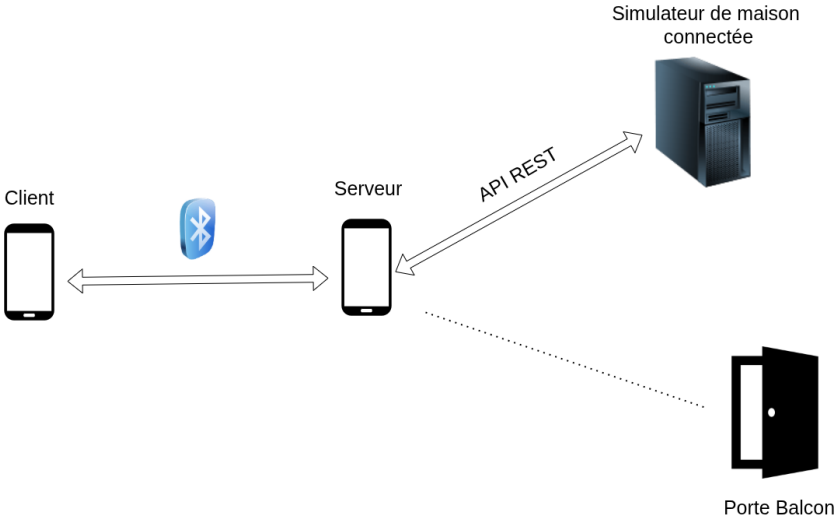


FIGURE 1 – Architecture

2 Architecture de notre Application

L'application Android se compose de deux activités :

— **MainActivity**
— **MainActivity2**

2.1 MainActivity

Elle permet à l'utilisateur de choisir le rôle du téléphone : **client** (télécommande) ou **serveur** (passerelle), et d'initialiser la connexion Bluetooth.

— Affichage de deux boutons permettant de sélectionner le rôle souhaité.

FIGURE 2 –

— Initialise une connexion Bluetooth entre les deux téléphones :
  — Le **serveur** ouvre un `BluetoothServerSocket` et attend une connexion.
  — Le **client** recherche les appareils appairés et se connecte via un `BluetoothSocket`.

Au début, nous avons choisi de définir manuellement l'adresse MAC du serveur pour établir directement la connexion Bluetooth. Cependant, cette méthode impose de toujours utiliser le même téléphone comme serveur, ce qui limite la flexibilité de l'application. Cette approche a donc été abandonnée, et le code correspondant a été conservé mais commenté dans le projet.

Désormais, lorsqu'un utilisateur choisit d'agir en tant que client, l'application affiche la liste des appareils Bluetooth déjà appairés. Cette liste est présentée sous forme de boîte de dialogue, permettant à l'utilisateur de sélectionner manuellement l'appareil serveur.

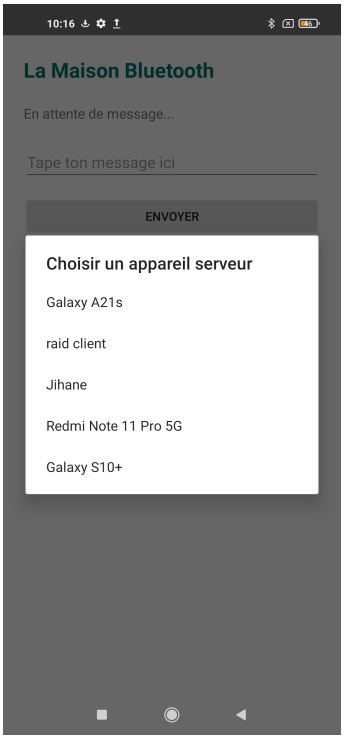
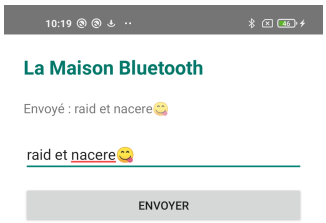


FIGURE 3 – Sélection d'un appareil Bluetooth appairé

— Une fois la connexion établie, les deux smartphones passent à la seconde activité, avec un socket Bluetooth actif pour l'échange de données.

Pour tester la connexion Bluetooth, nous avons ajouté une activitépermettant au client d'envoyer un message que le serveur affiche de son côté. Le test :



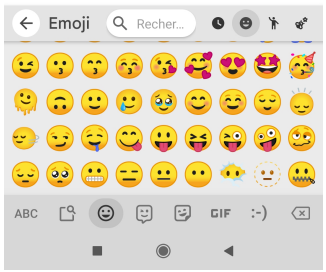


FIGURE 4 – Message envoyé depuis le client

FIGURE 5 – Message recu au serveur

2.2 MainActivity2

Cette activité permet l'affichage et le contrôle des appareils connectés de la maison.

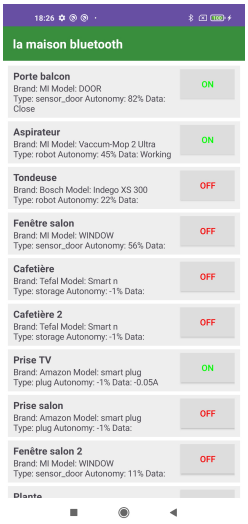


FIGURE 6 – Sélection d'un appareil Bluetooth appairé

— **Affichage dynamique des appareils connectés**
La méthode `createDeviceView(String name, String info, boolean turnOnOff, int id)` génère dynamiquement une vue pour chaque appareil, composée de deux `TextView` (nom + informations) et d'un bouton `ON/OFF`.

— **Requête GET pour récupérer la liste des appareils**
Une requête `GET` est envoyée à l'API REST via la bibliothèque Volley à l'URL :
http ://happyresto.enseeiht.fr/smartHouse/api/v1 /devices/idMaison. idMaison=28
Le résultat, au format JSON, est analysé pour extraire les informations des appareils et créer les vues correspondantes.

— **Envoi de commandes ON/OFF**
Lorsque le client clique sur un bouton associé à un appareil, la méthode sendMessageFromClient(int id, String onOff) envoie un message encodé (format "id :onOff") via Bluetooth grâce à une instance de TransferData. Cette requête est envoyée via un objet `StringRequest` de Volley.

— **Requête POST vers l'API REST pour modification d'état**
La méthode `update(int deviceId, boolean turnOnOff)` envoie une requête POST avec les paramètres `device_id` et `turn_on_off` à l'URL

— **Actualisation périodique de l'interface**
L'état des appareils est mis à jour automatiquement toutes les 10 secondes à l'aide d'un `Handler` et d'un `Runnable` :

```
handler.postDelayed(runnable, 10000);
```

Cela permet de garder l'interface synchronisée avec l'état réel des appareils.

FIGURE 7 – Actualisation toute les 10s

— **Gestion du cycle de vie**
Lorsque l'activité est interrompue (mise en pause), les mises à jour périodiques sont suspendues. Lorsqu'elle reprend, elles sont relancées. Cela permet de préserver les ressources du téléphone et d'éviter les fuites mémoire.

3 Conclusion

Ce projet nous a permis de développer une application Android complète, combinant une interface graphique dynamique, une communication HTTP avec une API REST, ainsi qu'un échange de données en Bluetooth entre deux téléphones.

Ce projet a vraiment renforcé notre maîtrise des outils Android (Activities, Handlers, Volley, Bluetooth). Il fait vraiment partie des projets les plus inspirants de l'année, car il aboutit à une application complète et concrète en fin de développement.