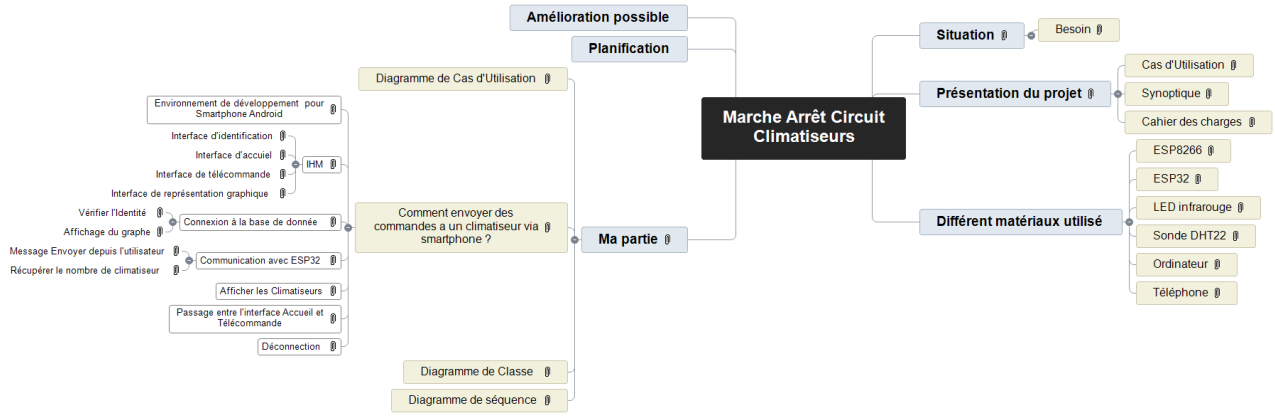
MARCHE ARRET CIRCUIT CLIMATISEURS



1 SITUATION

Le système présenter est utilisé dans un établissement (hôtels, bureaux, écoles) disposant de nombreuses climatisations individuelles disséminée dans des pièces éparses. L'objectif est de réduire les couts énergétiques attribué à l'oublie de l'arrêt par le personnel des systèmes de climatisations. Une marche ou un arrêt distant sous contrainte, horaire journalier, température ambiante, ou par opérateur direct permettra une diminution sensible du cout des factures énergétiques de l'établissement concerné. Une commande par smartphone connecté permettra en étant dans la salle concernée de s'affranchir des télécommandes des constructeurs.

1.1 BESOIN

L'objectif est de réduire les coûts énergétiques attribué à l'oublie de l'arrêt par le personnel des systèmes de climatisation,
permettra une diminution sensible du coût des factures énergétiques de l'établissement concerné.

Prenons exemple une entreprise a cinq pièces avec le système Marche Arrêt Circuit Climatiseur qui constitue deux climatiseur dans chaque pièces. L'entreprise a un compteur 13 kVA. Ces climatiseurs a une puissance en état de fonctionnement moyenne vaut 500 watts qui peut arriver jusqu'à une puissance de 1200 watts. Le système a fixer une consigne qui permet au climatiseur de fonctionner pendant une durée de 8 heures. D'après sur le site www.fournisseurs-electricite.com le prix du kwh a l'EDF option de base depuis le 9 mai 2018 vaut 0.1483 €. Un tableau présente l'énergie consommé avec son prix.

	Utilisation de la climatiseur	Kwh en puissance moyenne	Kwh en puissance ma
1 journée	8h	(500*8)*10 = 40 kwh	(1200*8))*10 = 96
1 semaine	8h	(500*(8*7))*10 = 280 kwh	(1200*(8*7))*10 = 67
1 mois (30 jrs)	8h	(500*(8*30))*10 = 1200 kwh	(1200*(8*30))*10 = 28

2 PRESENTATION DU PROJET

Objectif : Présenter les objectifs principaux

Nom du projet

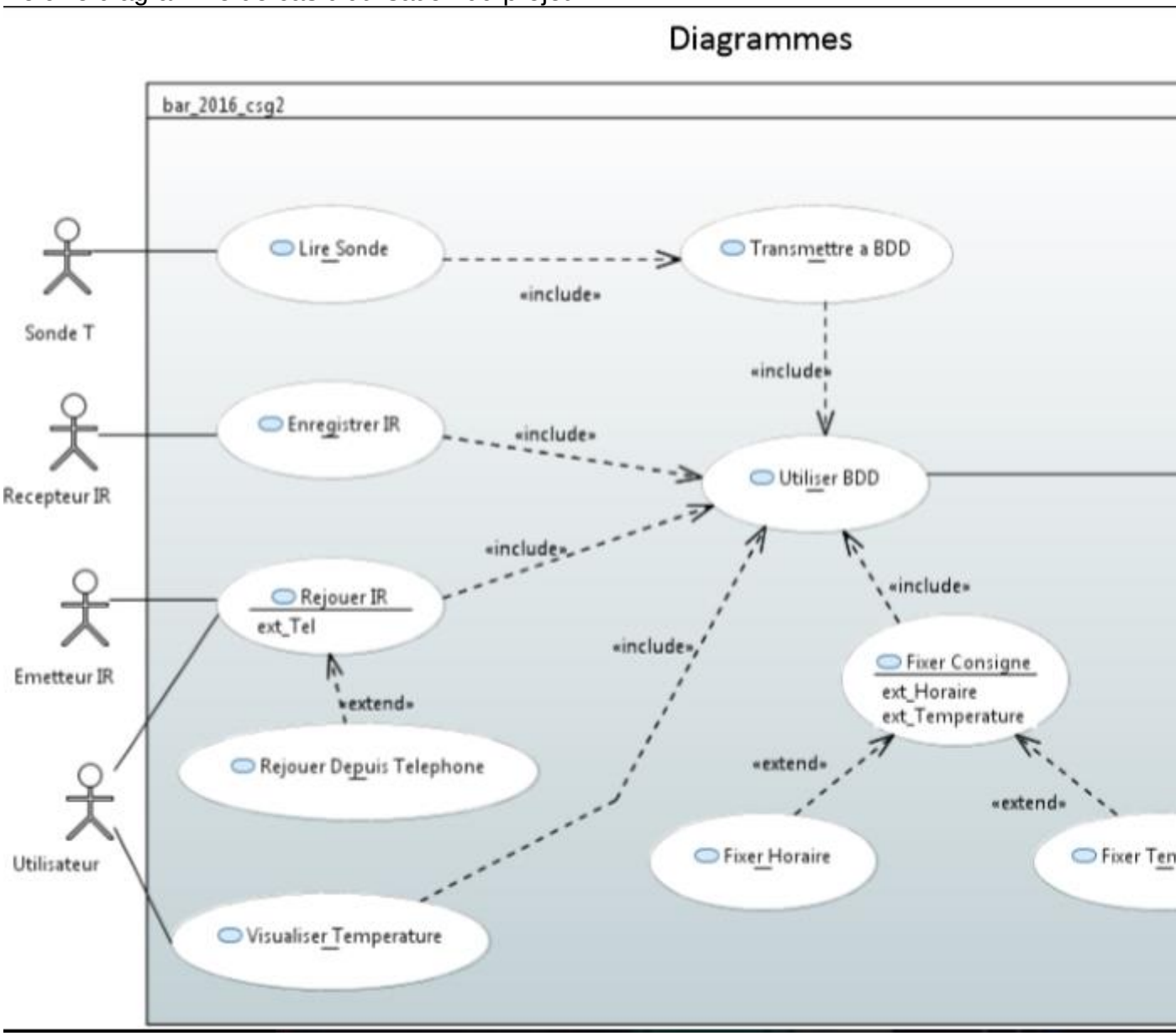
Le système peut être utilisé dans un établissement (hôtels, bureaux, écoles) disposant de nombreuses climatisations individuelles disséminée dans des pièces éparses. L'objectif est de réduire les coûts énergétiques attribué à l'oublie de l'arrêt par le personnel des systèmes de climatisations. Une marche ou un arrêt distant sous contrainte, horaire journalier, température ambiante, ou par opérateur direct permettra une diminution sensible du coût des factures énergétiques de l'établissement concerné. Une commande par smartphone connecté permettra en

étant dans la salle concernée de s’affranchir des télécommandes des constructeurs.

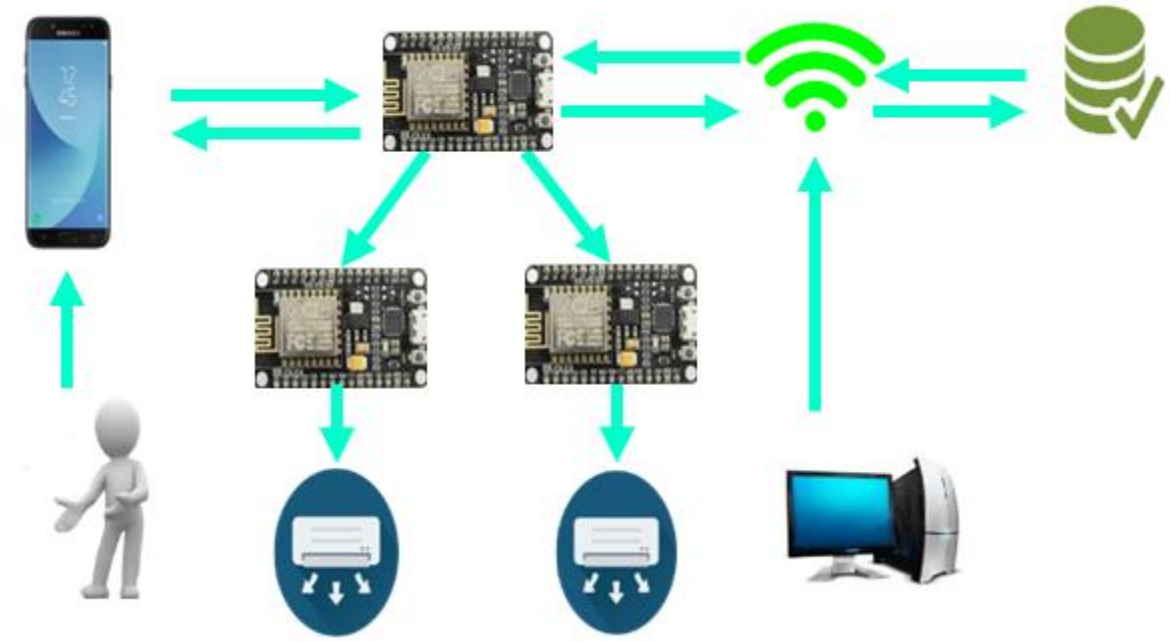
Ajouter une image du projet.

2.1 CAS D'UTILISATION

Voici le diagramme de cas d'utilisation du projet.



2.2 SYNOPTIQUE



2.3 CAHIER DES CHARGES

Le produit rend service au responsable du parc de climatiseurs, en permettant une commande pouvant être groupée de gestion de chaque unité. Il permet ainsi de participer la baisse des émissions CO2 en assurant la maîtrise d'énergie nécessaire aux divers climatiseurs.

Etudiant 1

CU :Enregistrer IR, Fixer Consigne(ext_H)

- Mise en place d'une librairie d'analyse de protocole IR.
- Création d'une BDD associant Message IR aux fonctionnalités d'une télécommande. Marche/Arret, Up/down, Timer et climatiseurs associé

Etudiant 2

CU Rejouer

- Réalisati
  l'envoie
  d'un ou
- Prise en
  réceptio

- Visualisati
courbes de

Etudiant 3

CU Lire_Sonde, Fixer Consigne (ext_T)

- Adaptation d'une librairie de lecture de température.
- Récupération de valeur de température pour insertion dans une BDD

Etudiant 4

CU Rejouer

- Réalisation
mobile. Per
pour march
climatiseur

3 DIFFERENT MATERIAUX UTILISE

3.1 ESP8266

L'ESP8266 est une puce Wi-Fi à faible coût avec une pile TCP / IP complète et une capacité de microcontrôleur développé par le fabricant chinois Espressif.
L'ESP8266 peut se programmer de plusieurs façons :
En C++, avec l'IDE Arduino
En JavaScript, avec le firmware Espruino
En MicroPython, avec le firmware MicroPython
En C, avec le SDK d'Espressif

Voici les caractéristiques de l'ESP32 :

Manufacturer Espressif Systems

Type 32-bit microcontroller
CPU @ 80 MHz (default) or 160 MHz
Memory 32 KiB instruction, 80 KiB user data
Input 16 GPIO pins
Power 3.3 V DC

3.2 ESP32

L'ESP32 est un microcontrôleur avec une connectivité WiFi, Bluetooth Classique et faible consommation d'énergie en un seul puce. L'ESP32 va permettre d'intégrer les fonctionnalités IoT dans une taille macro-métrique. L'ESP32 peut se programmer de plusieurs façons, en C++, avec l'IDE Arduino IDE avec **ESP32 Arduino Core**.

Voici les caractéristiques de l'ESP32 :

-Architecture dual-core (un cœur pour les applications et un cœur en charge du Wifi)
-160 à 240 MHz (tensilica Xtensa LX6 microprocesseur)
-Bluetooth dual mode (Classique et BLE)
-Antenne intégré et Connecteur IPEX pour antenne externe.
-La mémoire RAM passe à 520 kib SRAM
-16 Mb mémoire flash
-Mode deep sleep amélioré
-Presque 40 GPIOs, Plusieurs entrées ADC
-Niveaux de sécurité WEP, WPA/WPA2 PSK/Entreprise, Cryptage Hardware : AES / SH / ECC / RSA-4096
-Plage de fonctionnement de 2,2 à 3,6 V
-Température de fonctionnement de -40°C à +125°C

3.3 LED INFRAROUGE

Un capteur infra-rouge IR est l'œil électronique qui se trouve sur de nombreux appareils qui viennent avec une télécommande. La commande à distance transmet un faisceau infra-rouge, invisible à l'œil humain, sur une distance fixe de l'appareil, qui suit les instructions de transmission codées de chaque bouton de la télécommande. Par exemple, en appuyant sur "On" sur la télécommande du climatiseur provoque la mise en marche du moteur qui allume le climatiseur.

J'attends Mathias pour les caracteristiques.....

3.4 SONDE DHT22

Il s'agit d'un module de température et d'humidité numérique calibré avec capteur intégré DHT22 (AM2302), qui offre une précision supérieure et une plage de mesure plus large que le DHT11.
Il peut être utilisé pour détecter la température ambiante et l'humidité, à travers l'interface standard à un fil. Le DHT22 a ces caractéristiques suivante.

- Température

Résolution: 0.1 ° C
Précision: ± 0,5 ℃
Plage de mesure: -40 ° C ~ 80 ° C

- Humidité

Résolution: 0,1% HR
Précision: ± 2% HR (25 ° C)
Plage de mesure: 0% RH ~ 99,9% RH

- Tension de fonctionnement: 3.3V ~ 5.5 V
- Condition de stockage recommandée

Température: 10 ° C ~ 40 ° C
Humidité: 60% HR ou moins

Applications:

- Station météo
- Contrôleur d'humidité
- Dispositif de test et de détection

Comment utiliser:
Dans le cas de travailler avec un MCU:

- VCC ↔ 3.3V ~ 5.5V
- GND ↔ GND
- DOUT ↔ MCU.IO

3.5 ORDINATEUR

L'utilité d'un ordinateur permet sur une page web d'entrer les différents information d'une nouvelle climatiseur, d'enregistre les informations et les messages IR (trame IR) dans la base de donnée

3.6 TELEPHONE

L'utilisation du Téléphone dans ce projet permet d'effectuer les différents action a un ou plusieurs climatiseur(s) dans une pièces après avoir installer l'application sur le téléphone.

4 MA PARTIE

Réalisation d’une Interface Homme Machine (IHM) pour appareil mobile qui permet

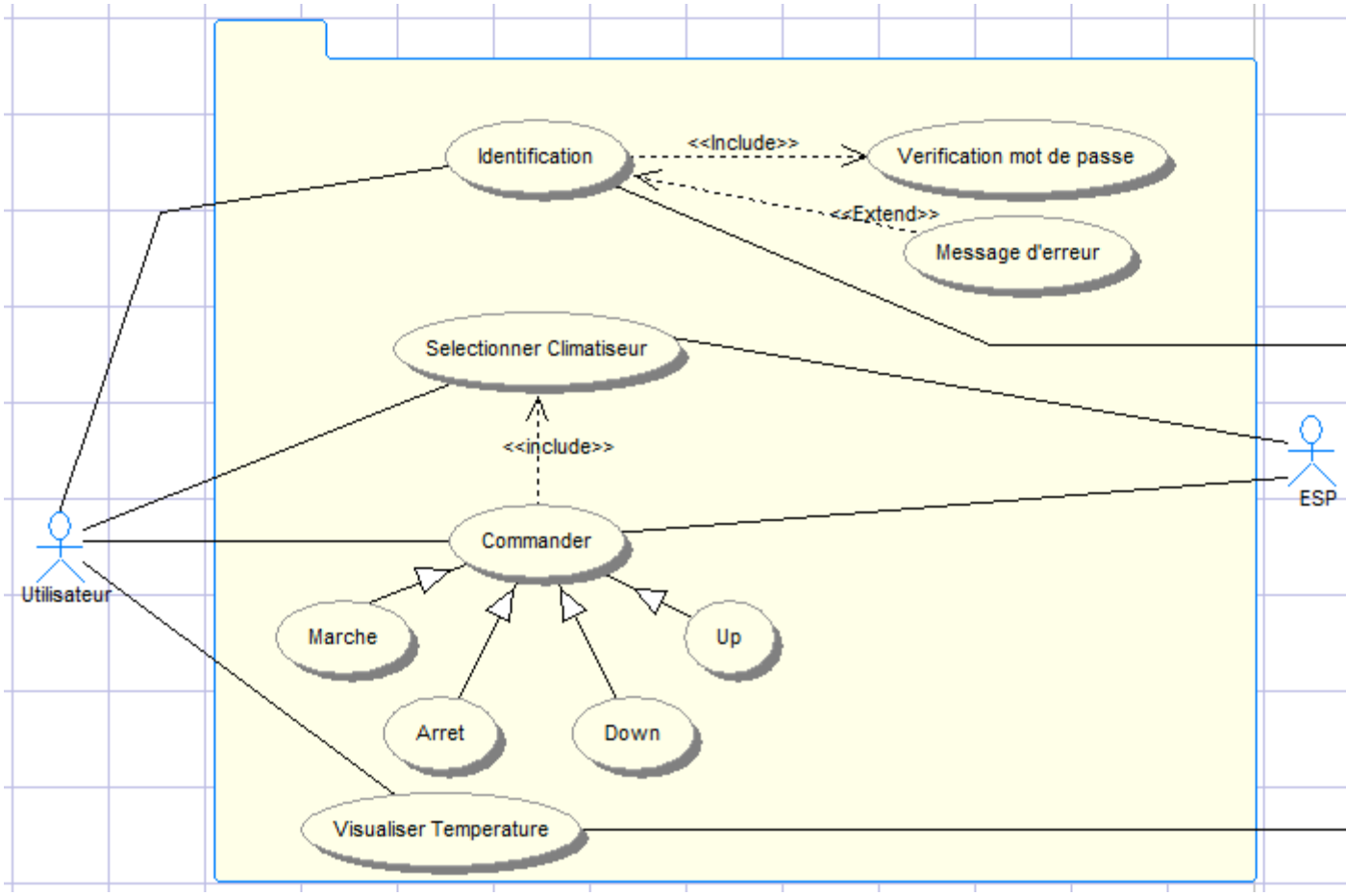
d’envoyer de rejouer les trames IR pour marche/arrêt, up/down d’un climatiseur
Voici comment je découpe ma partie.....

-Diagramme de Cas d'Utilisation
-Diagramme de Classe
-Diagramme de Séquence
    -LoginHome
    -ClimHome
    -GraphTemperature

Comment envoyer des commandes a un climatiseur via smartphone ?

4.1 DIAGRAMME DE CAS D'UTILISATION

Voici une vision globale du comportement fonctionnel de l'application.



4.2 COMMENT ENVOYER DES COMMANDES A UN CLIMATISEUR VIA SMARTPHONE ?

Avant de procédé d'allumer, éteindre et changer la température, il y a différent tache a effectue qui sont :
Choisir l'environnement de développement,

Représenter l'application,
Obtenir une connexion avec la base de donnée,
Communiquer avec ESP,
Représenter un graphique avec les quatre dernières température enregistre dans la basse de donnée.

4.2.1 ENVIRONNEMENT DE DEVELOPPEMENT POUR SMARTPHONE ANDROID

Recherche de différent environnement de développement pour mobile.

Dans mes recherches il y a plusieurs environnements de développement pour mobile, je vous présente « integrated development environment » (IDE) suivant :

Android Studio sorti en 2013, dont l'installation est réalisable en même temps que le SDK sur tout type de systèmes d'exploitation, représente l'IDE privilégié par Google pour la création d'applications Android.
Grâce à sa puissance, sa simplicité et sa gratuité, il a pu détrôner facilement tous les autres environnements utilisés jusqu'à lors.
En se basant sur IntelliJ IDEA, cet utilitaire ne permet pas juste de créer des applications compatibles avec votre smartphone, mais elles pourront fonctionner aussi sur vos montres connectées, téléviseurs connectés et tablettes. Les développeurs pourront aussi visualiser leur travail grâce à un émulateur intégré.

Eclipse est un IDE créer le 7 Novembre 2001 qui contient un « workspace » et des extensions de plug-in. Les programmes d'Eclipse sont plus souvent écrite en Java pour développer des applications de Java, mais Eclipse peut développer des applications sur d'autre langage de programmation par des plug-ins qui inclue : C, C++ C#, Java, JavaScript, Perl, Php, Python et d'autre encore.

NetBeans est un environnement de développement intégré (EDI), placé en *open source* par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java, NetBeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy, le PHP et le HTML par l'ajout de *greffons*. Il offre toutes les facilités d'un IDE moderne (éditeur en couleurs, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Compilé en Java, NetBeans est disponible
sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDKest requis pour les développements en Java

Qt Creator est un environnement de développement intégré multiplate-forme faisant partie du framework Qt. Il est donc orienté pour la programmation en C++, développé par « Qt Project ».Il intègre directement dans l'interface un débogueur, un outil de création d'interfaces graphiques, des outils pour la publication de code sur Git et Mercurial ainsi que la documentation Qt. L'éditeur de texte intégré permet l'autocomplétion ainsi que la coloration syntaxique. Qt Creator utilise sous Linux le compilateur gcc. Il peut utiliser MinGW ou le compilateur de Visual Studio sous Windows.

Qt Creator a été traduit en français par l'équipe Qt de « Developpez.com ».

Conclusion

Android Studio est l'IDE officiel pour le développement d'applications Android, basé sur IntelliJ IDEA. En plus des fonctionnalités que vous attendez d'IntelliJ, Android Studio propose:
*La simplicité et sa gratuité
*Système flexible de Gradle-based build
*Générer plusieurs fichiers apk
*Modèles de code pour vous aider à créer des fonctionnalités d'application courantes
*Éditeur de mise en page riche avec prise en charge de l'édition de thèmes par glisser-déposer
*Prise en charge intégrée de Google Cloud Platform, facilitant l'intégration de Google Cloud Messaging et App Engine

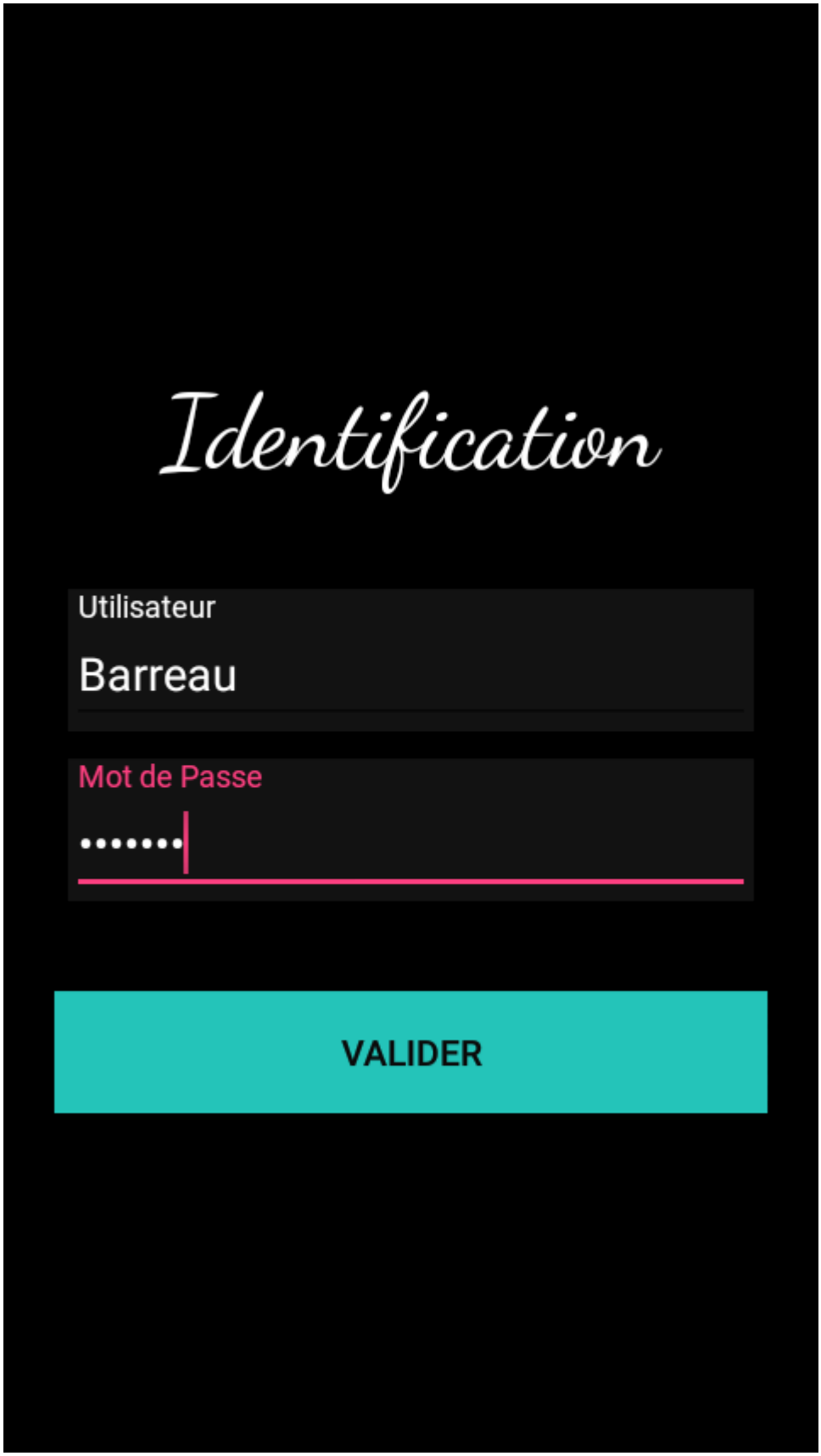
4.2.2 IHM

Les interactions Homme-machines (IHM) définissent les moyens et outils mis en œuvre afin qu'un humain puisse contrôler et communiquer avec une machine.

L'interface homme-machine (IHM) est l'interface utilisateur qui relie l'opérateur au dispositif de commande d'un système industriel.
Les systèmes de contrôle industriel intègrent des équipements et des logiciels conçus pour surveiller et contrôler le fonctionnement des machines-outils et des appareils associés dans les environnements industriels, notamment ceux désignés sous le terme de système essentiel.

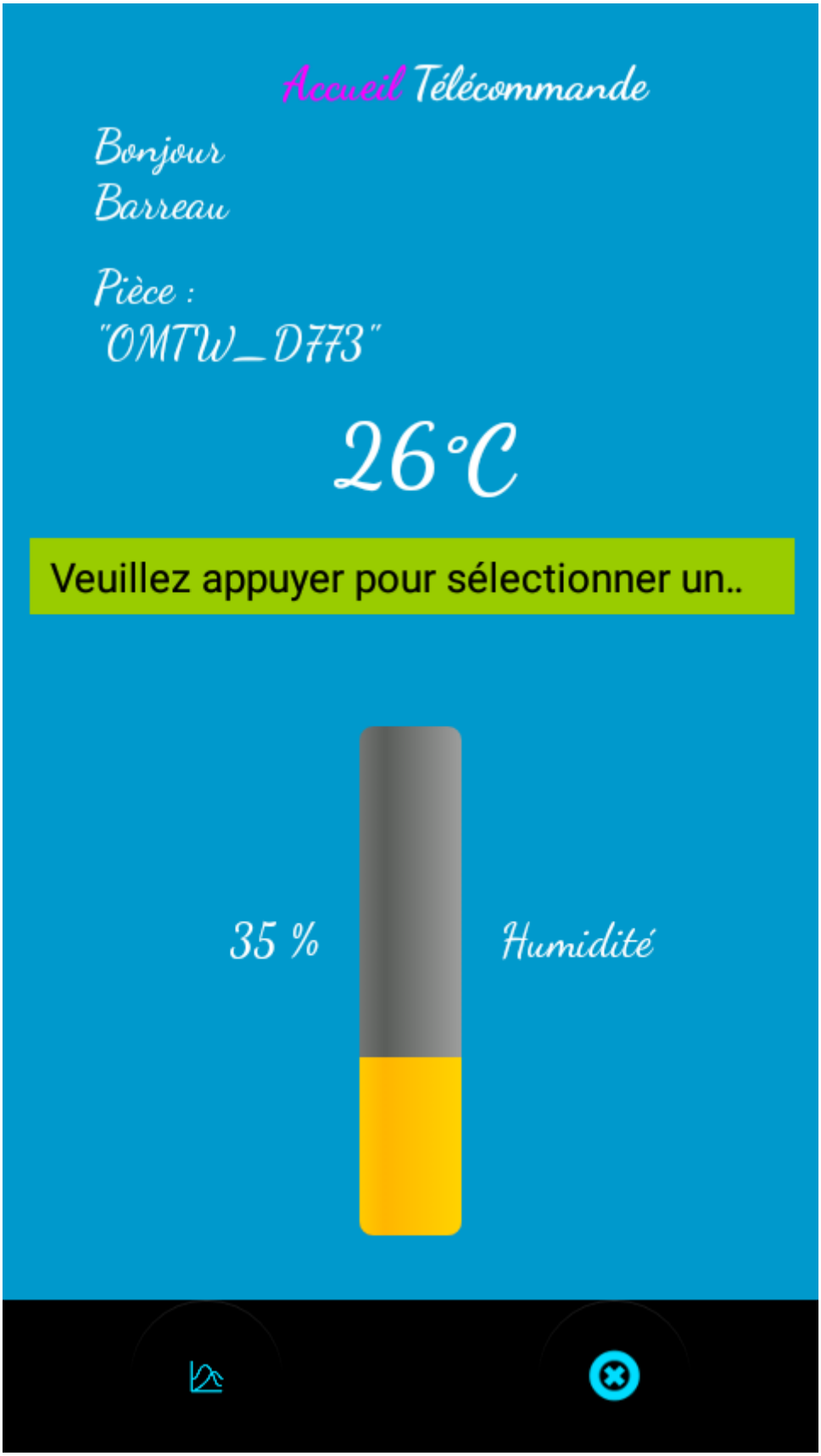
4.2.2.1 INTERFACE D'IDENTIFICATION

Au fil du temps l'équipe et moi réalise que il faut savoir si c'est un utilisateur qui travail pour une entreprise (ex: un professeur) qui utilise l'application, alors j'ai rajouter un espace d'identification qui sera la premier page avant d'utiliser l'application.



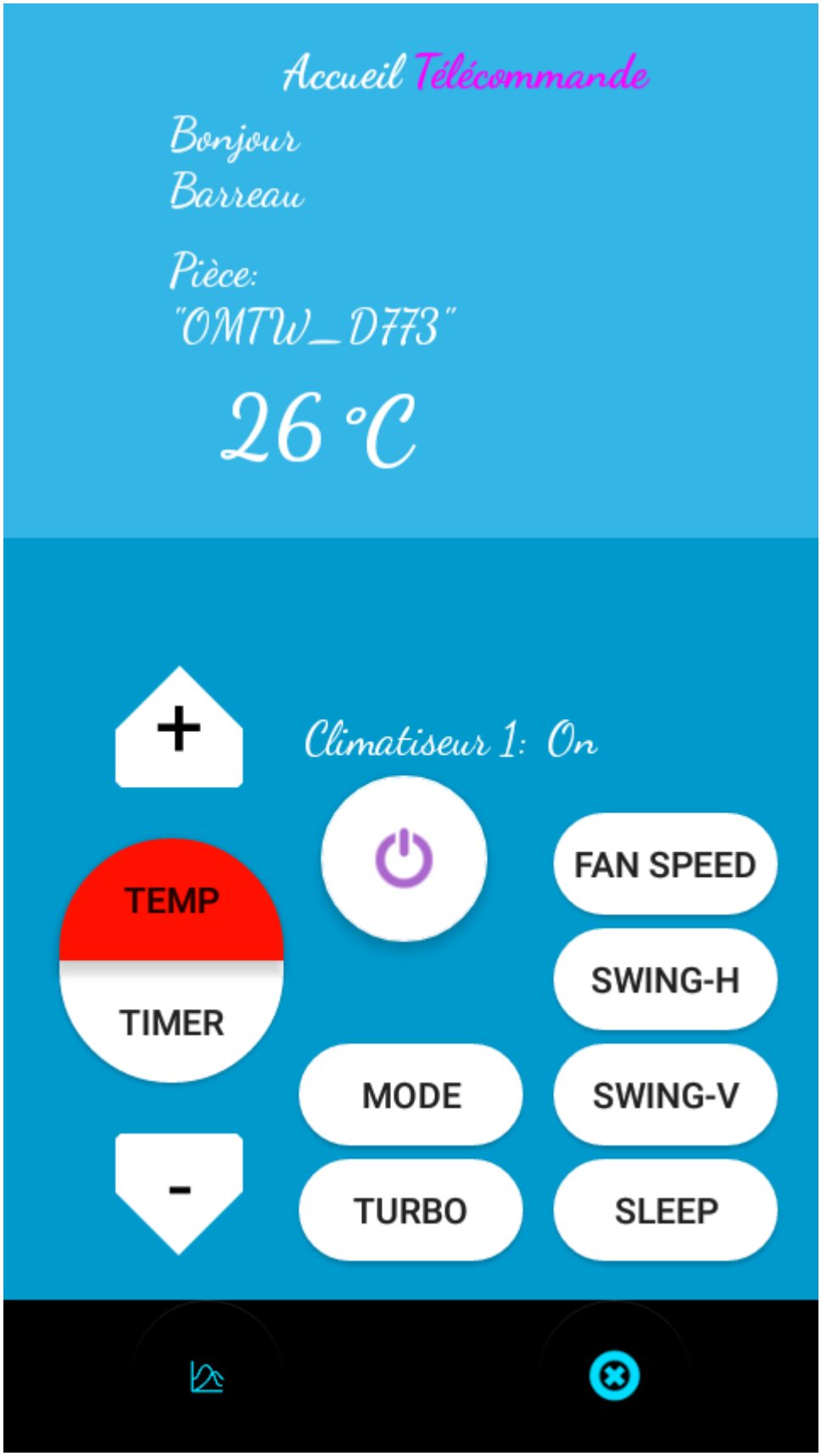
4.2.2.2 INTERFACE D'ACCUIEL

Suivi de l'interface d'identification après avoir identifie l'utilisateur, il sera dans un interface d'accueil. Ici l'utilisateur a des informations dans quels pièce il est connecté, la température, l'humidité de la pièce et une sélection de climatiseur disponible dans la pièce.


Accueil Télécommande
Bonjour
Barreau
Pièce :
"OMTW_D773"
26°C
Veuillez appuyer pour sélectionner un..
35 %
Humidité

4.2.2.3 INTERFACE DE TELECOMMANDE

Arriver a cette interface il faut sélectionner un ou tous les climatiseur disponible dans l'Accueil.
Dans l'interface de Télécommande se situe les actions d'allumer, d'arrêter le climatiseur et d'augmenter ou diminuer la température. L'utilisateur peur observer la température envoyer au climatiseur.



4.2.2.4 INTERFACE DE REPRESENTATION GRAPHIQUE

Visualiser quatre dernière température de la pièce connecte sur un graphique qui est

accessible soit à l'interface d'Accueil ou à l'interface de Télécommande. L'interface

graphique est accessible par un bouton avec un icône
(un icône de représentation graphique)

4.2.3 CONNEXION A LA BASE DE DONNEE

La connexion a la base de donne est important car elle permet de récupérer des informations critique du bon fonctionnement de l'application.
*Comparer l'identifiant que l'utilisateur a entre avec celle dans la base de donnée.
*Récupérer les quatre dernière température et humidité enregistre dans la base de donnée.

Pour obtenir une connexion a la base de donnée. Je dois analyser pas par pas, j'ai :
-Qu-es ce que j'utilise qui m'aide a connecter a la base ?,
-Besoin la configuration de la base de donnée (l'adresse IP, nom de la base de donnée, identifiant de la base de donnée),
-Appeler une connexion dans un ou plusieurs ficher.

la bibliothèque JDBC (**Java Database Connectivity**) est une bibliothèque standard de l'industrie pour la connectivité indépendante de la base de données entre le langage de programmation Java et un large éventail de bases de données. Bases de données SQL et autres sources de données tabulaires, telles que des feuilles de calcul ou des fichiers plats. L'API JDBC fournit une API de niveau appel pour l'accès à la base de données SQL.
La technologie JDBC vous permet d'utiliser le langage de programmation Java pour exploiter les fonctionnalités «Écrire une fois, exécuter n'importe où» pour les applications qui requièrent un accès aux données de l'entreprise.
Tandis que l'application est développé sur android studio en java il est recommender d'utiliser un pilot MySQL Connector/J qui fournit une

connectivité pour les applications client. MySQL Connector/J implémente l'API Java Database Connectivity (JDBC).

Ensuite je cherche a savoir de la part de mon collègue le nom de la basse de donnée et l'adresse IP qu'il a configuré:
-L'adresse IP de la base de donnée (**"93.121.229.118"**)
-le nom de la base de donnée (**"MACC"**)
-Identification base de donnée (**"pi"**)
-Mot de passe pour accéder la base de donnée (**"Simconolat"**)

Avec ces informations obtenu de mon collègue, je crée un ficher ConnectionClass.java avec une classe ConnectionClass, j'utilise un prototype **public** Connection e4Csg1MACC_CONN() qui permettra a d'autre classe d'appeler cette méthode en cas besoin.
Dans la méthode il aura un objet de Connection conn qui sera initialise par l'adresse IP de la basse de donnée suivie du nom de le base de donnée, puis les identifications (username, mot de passe) pour sécurise la connexion.

Avoir une connexion de la base de donnée permet a plusieurs méthode dans plusieurs ficher java puise avoir une connexion pour exécute la tache X. Chaque ficher qui a besoin de récupérer et/ou comparer des valeurs dans la basse de donnée, créer un objet de Connection avec un accès a la méthode e4Csg1MACC_CONN().

4.2.3.1 VERIFIER L'IDENTITE

J'ai une application qui nécessite de comparer l'information saisie par l'utilisateur avec les informations pré-enregistré par l'administration dans la basse de donnée, ce qui donnerait l'accès a l'utilisateur d'utiliser l'application dans son intégralité.

Tout d'abord Quand l'utilisateur ouvrira l'application il aura une fenêtre d'identification. Il devra saisir ces informations donnée par l'administration (nom d'utilisateur, mot de passe) et appuie sur le bouton « Valider ». Les informations entré vont être compare avec celle trouve dans la base de donnée, si les informations compare sont exact l'utilisateur accès à la fenêtre suivante sinon un message d'erreur informe l'utilisateur.

Voici les différents étapes comment je procède à la validation d'identité :
-Récupérer l'information saisie par l'utilisateur

-Connexion à la base de donnée
-Chercher les information saisie si elle existe
-Valider ou non l'accès

L'utilisateur écrit ces informations dans deux objets « EditText » Utilisateur et Mot de Passe que j'affecte mes objets EditText dans xml à mes objets de constructeurs « user » qui récupère le nom de l'utilisateur et « pwd » qui récupère son mot de passe. Voici le code ci-dessous qui permet de faire cela.

```
private EditText user,pass;
private String userStr;
private String passStr;


user = (EditText) findViewById(R.id.User);
pass = (EditText) findViewById(R.id.pass);

userStr = user.getText().toString();
passStr = pass.getText().toString();
```

La création d'un objet "con" de "Connection" que j'affecte la méthode "e4Csg1MACC_CONN()" dans la classe ConnectionClass.
L'appelle de con permet lancer la connexion entre moi et la base de donnée. La représentation du code :
Connection con = **connectionClass**.e4Csg1MACC_CONN();

Avec les informations saisie par l'utilisateur, j'écris une requête SQL qui me permet d'avoir tout les noms d'utilisateurs et leurs mot de passe, mais avant il faut établir une connexion avec la base de donnée puis crée un objet pour l'execution du SQL Statement et enfin un objet ResultSet qui donne le résulta de la requête SQL. Avec le résulta de la requête je cherche si l'utilisateur a bien saisie les information ou sinon les informations sont incorrect ou il n'existe pas dans la base de donnée.

J'ai ainsi formulé un prototype serait: **private void** e4Csg1MACC_get_db_Data() et une classe abstrait : **private class** E4doLogin **extends** AsyncTask<String,String,String>.

Dans la méthode e4Csg1MACC_get_db_Data(), j'établie une connexion avec la base de donnée. Si ma connexion est nulle (n'existe pas) j'affiche un message "Veuillez vérifier votre connexion internet". Si la connexion existe alors:
-préparer une requête SQL
-exécuter la requête
- trouver et faire correspondre les informations d'identification entrées par l'utilisateur et stockées dans le base de donnée.
-Si trouvé, affiche un message disant "Bienvenu"

-Sinon, affiche un message "Erreur d'identité… Utilisateur et ou mot de passe incorrect!!!"

Voici le code de la méthode :

```
 private void e4Csg1MACC_get_db_Data(){
    try {
        //call my e4Csg1MACC_CONN() method situated in my ConnectionClass file
        // to establish a connexion with the
db.....................................................
        //if my connexion is null (does not exit) show a
message.................................
        Connection con = connectionClass.e4Csg1MACC_CONN();
        if (con == null) {
            message = "Veuillez vérifier votre connexion internet";
        } else {
            //if the connexion exist then :
            //prepare a query
            //execute the query
            //find and match the credentials entered and stored in the db
            //      -if found, show a message saying "valider successfully"
            //      -if not, show a message "Error credential...not
match!!!"....................

            String query = " select * from PROFESSEUR where NOM = '"+userStr+"' and
MDP = '"+passStr+"'";
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);


            while (rs.next()) {
                userDB = rs.getString(2);
                passDB = rs.getString(5);

                if (userDB.equals(userStr) && passDB.equals(passStr)) {
                    isSuccess = true;
                    message = "Bienvenu";
                }
            }

            if(!userDB.equals(userStr)|| !passDB.equals(passStr)){
                message = "Erreur d'identité… Utilisateur et ou mot de passe
incorrect!!!";
            }
        }
    }catch (Exception ex) {
        //if the db does not exist
        //if the table does not exist
        //show a message explaining it to the
user.........................................
        isSuccess = false;
        message = "Exceptions....." +ex;
        Log.e("Exceptions.....",ex.getMessage());
    }
}
```

Qu'es qu'un "AsyncTask" ?
AsyncTask permet une utilisation correcte et facile du thread UI. Cette classe me permet d'effectuer des opérations en arrière-plan et de publier des résultats sur le thread d'interface utilisateur sans avoir à manipuler les threads et / ou les

gestionnaires.
AsyncTask est conçu pour être une classe d'assistance autour de Thread et Handler et ne constitue pas un cadre de threading générique. AsyncTasks devrait idéalement être utilisé pour des opérations courtes.
Une tâche asynchrone est définie par un calcul qui s'exécute sur un thread d'arrière-plan et dont le résultat est publié sur le thread de l'interface utilisateur.
Une tâche asynchrone est définie par 3 types génériques, appelés Params, Progress et Result, et 4 étapes, appelées onPreExecute, doInBackground, onProgressUpdate et onPostExecute.

Quand la **class** E4doLogin **extends** AsyncTask<String,String,String> s'exécute, il y a 4 étapes:
onPreExecute (), invoqué sur le thread de l'interface utilisateur avant l'exécution de la tâche. Cette étape est normalement utilisée pour configurer la tâche.
J'ai affiché une barre de progression dans l'interface utilisateur qui signal le chargement du graphique.
doInBackground (String... params), invoqué sur le thread d'arrière-plan immédiatement après l'exécution de onPreExecute (). Cette étape est utilisée pour effectuer un calcul en arrière-plan qui peut prendre beaucoup de temps. Les paramètres de la tâche asynchrone sont transmis à cette étape. Le résultat du calcul doit être retourné par cette étape et sera renvoyé à la dernière étape.
Ici où j'exécute ma méthode e4Csg1MACC_get_db_Data().
onPostExecute (Résultat), invoqué sur le thread d'interface utilisateur après la fin du calcul de l'arrière-plan. Le résultat du calcul de l'arrière-plan est passé à cette étape en tant que paramètre.
Au finale un message s'affiche tout dépend si l'échec de la connexion, l'état d'identification. Si les informations trouvé est un succès alors je passe sur l'autre écran avec le nom de l'utilisateur.

Sur l'écran de "Identification" il y a un bouton "Valider" qui permet de lancer l'identification, que je déclare ici :
```
private Button valider;
```

Puis j'affecte ma variable "Valider" a mon bouton objet "Valider" dans "activity_login_home.xml" qui a un id "Valider". Voici la représentation coder :
```
valider = (Button) findViewById(R.id.valider);
```

Pour représenter l'action qui donne l'accès a l'accueil, j'utilise la méthode "OnClickListener()" de la librairie "View" d'Android Studio, quand j'appuie mon bouton "Valider" il exécute la classe "E4doLogin" dans "setOnClickListener()".
Le code qui représenter l'action :
```
valider.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
    public void onClick(View v) {
        E4doLogin login=new E4doLogin();
        login.execute();
    }
});
```

Alors ci-dessous voici le code effectuer pour ma classe E4doLogin:

```
private class E4doLogin extends AsyncTask<String,String,String>
{
    @Override
    protected void onPreExecute() {
        /*while the doInBackground(String... params) : protected is executed
        * show a search symbol and mark
"Loading..."..............................................*/
        progressDialog.setMessage("Chargement...");
        progressDialog.show();

        super.onPreExecute();
    }

    @Override
    protected String doInBackground(String... params) {
        //if one or two fields are empty, advice the
user.....................................
        if (userStr.trim().equals("") || passStr.trim().equals(""))
            message = "Merci de compléter tous les champs....";
        else {
            //if not execute my
"e4Csg1MACC_get_db_Data()".....................................
            e4Csg1MACC_get_db_Data();
        }
        return message;
    }

    @Override
    protected void onPostExecute(String s) {
        // show the message stored in a
variable.................................................
        Toast.makeText(getBaseContext(),""+ message,Toast.LENGTH_SHORT).show();

        //if successfully the input data and saved data is the same,
        // give access to the next screen and passe the
username.............................
        if(isSuccess) {
            Intent intent=new Intent(LoginHome.this,Home.class);    //Home.class
            intent.putExtra("user", userStr);
            startActivity(intent);
        }
        //hide my loading message and
symbol.....................................................
        progressDialog.hide();
    }
}
```

4.2.3.2 AFFICHAGE DU GRAPHE

J'ai une basse de donnée distant qui contient différents températures et les heures attribuent a chaque température de plusieurs pièces. Le principe est de récupérer les quatre dernières températures associer a leur heures qui constituera la température du graphique avec l'aidée de la bibliothèque GraphView.

GraphView est une bibliothèques pour Android pour créer des diagrammes flexibles et beaux.
C'est facile à utiliser, à intégrer et à personnaliser.
GraphView aide à créer des graphiques linéaires, des graphiques à barres, des graphiques à points
ou implémenter vos propres types personnalisés.

Voici les différents étapes comment je procède à réaliser mon graphique:
-Connexion à la base de donnée
-Récupération des valeurs de température et d'heure
-Représentation du graphe avec les valeurs reçu

La création d'un objet "con" de "Connection" que j'affecte la méthode "e4Csg1MACC_CONN()" dans la classe ConnectionClass.
L'appelle de con permet lancer la connexion entre moi et la base de donnée. La représentation du code :
Connection con = **connectionClass**.e4Csg1MACC_CONN();

D'après ma collègue, elle écrit dans la base de donne MACC le numéro de salle qui représente l'ID unique, le nom de salle, la température, l'humidité, le jour et l'heure. Avec cette information j'écris une requête SQL qui me permet d'avoir les valeur de température et celle des heures, mais avant il faut établir une connexion avec la base de donnée puis crée un objet pour l'execution du SQL Statement et enfin un objet ResultSet qui donne le résulta de la requête SQL. Ce résulta de la requête me permet de récupérer mes différent donnée nécessaire (ex: Température et heure) que je stock dans une variable et j'utilise dans la création du graphe.

Au plus simple je veut une connexion à la base de donnée que je récupère des valeurs et je l'utilise pour mon graphique. Ainsi j'utilise deux tableaux, un pour les températures et l'autre pour l'heur:
private ArrayList<Double> **tableTemp** = **new** ArrayList<Double>();
private ArrayList<Time> **tableTime** = **new** ArrayList<Time>();

Voici deux méthode que j'utilise:
private void e4Csg1MACC_sqlQuery();
public DataPoint[] e4Csg1MACC_getDataPoint();

Dans ma méthode e4Csg1MACC_sqlQuery() effectue la connexion puis la requête SQL que je stock mes valeurs récupéré dans une ArrayList et j'affecte la

position du ArrayList a ma variable concerné. Exemple: j'ajoute dans "arrayListName" le résulta de ma requête SQL qui se situe a la première position dans la basse de donnée arrayListName.add(resultSet.getInt(1)).

Avant tout j'ai une **class** E4cBackground **extends** AsyncTask<Void, Void, String> s'exécute en arrière plan dès l'affichage de l'écran Temperature Graph. Quand la **class** E4cBackground **extends** AsyncTask<Void, Void, String> s'exécute il y a 3 étapes:
onPreExecute (), J'affiche une barre de progression dans l'interface utilisateur qui signal le chargement du graphique.
doInBackground (Params ...), ici où j'exécute ma méthode e4Csg1MACC_sqlQuery().
onPostExecute (Résultat), Ici, je prépare mon graphe avec l'aide de la bibliothèque qui créer ma courbe avec mes valeurs récupéré en utilisant ma méthode e4Csg1MACC_getDataPoint(). J'ai besoin d'identifier le layout du graphe puis j'utilise la classe Series pour remplir le graphique avec mes données. Une série contient les points de données d'une "ligne", qui sera représentée sous la forme d'une ligne, de points ou de barres. Ma série sera représentée sous forme de ligne, alors je choisi LineGraphSeries la sous-classe de Série.

La méthode e4Csg1MACC_sqlQuery() permet de vérifier si une connexion a la base de donnée est réussite, d'exécuter une requête SQL, enregistrer les résultas (Temperature et temps) dans deux tableaux.
Verifer si il y a une connecxion avec un teste si "con" est null sinon etape suivante, voici le code :

```
if (con == null) {
    message = "Veuillez vérifier votre connexion internet";
} else {
```

Si il y a une connexion j'exécute une requête SQL qui va récupérer tous les température et l'heure récente de la journée qui sont stocker dans deux tableaux.
Le code qui permet:

```
String query = "select TEMPERATURE,HEURE from SALLE where NOM_SALLE = 'V14' and
DATE_JOUR = '2018-04-10' order by HEURE";
                    Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery(query);

        //
        while(rs.next()){
            tableTemp.add(rs.getDouble(1));
            tableTime.add(rs.getTime(2));
        }
```

Ensuite j"affecte a mes quatre variables de température a leur index du tableau "tableTemp" (température) et mes quatre variable de temps a leur index du

tableau tableTime (heure).

```
temperature1 = tableTemp.get(0);
        temperature2 = tableTemp.get(1);
        temperature3 = tableTemp.get(2);
        temperature4 = tableTemp.get(3);

        tempTime1 = tableTime.get(0);
        tempTime2 = tableTime.get(1);
        tempTime3 = tableTime.get(2);
        tempTime4 = tableTime.get(3);
```

Ci-dessous voici la globalité du code :

-e4Csg1MACC_sqlQuery()

```
private void e4Csg1MACC_sqlQuery() throws SQLException {
    //open a
connection.....................................................................
    Connection con = connectionClass.e4Csg1MACC_CONN();
    //test the
connection.....................................................................
    if (con == null) {
        message = "Veuillez vérifier votre connexion internet";
    } else {
        //prepare a query and a
statement..............................................................
        //i need to get four temperatures and their hours and place them from
recent to dated...
        String query = "select TEMPERATURE,DATE_JOUR from SALLE where NOM_BAT =
'"+salleName+"' and DATE_JOUR = '"+timeStamp+"' order by DATE_JOUR";
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery(query);

        //store the results in the
table.....................................................
        while(rs.next()){
            tableTemp.add(rs.getDouble(1));
            tableTime.add(rs.getTime(2));
        }
        isSuccess = true;
        //affect the four recent values to my temperature variables from the
ArrayList..........
        temperature1 = tableTemp.get(0);
        temperature2 = tableTemp.get(1);
        temperature3 = tableTemp.get(2);
        temperature4 = tableTemp.get(3);

        //affect the four recent values to my temperature variables from the
ArrayList..........
        tempTime1 = tableTime.get(0);
        tempTime2 = tableTime.get(1);
        tempTime3 = tableTime.get(2);
        tempTime4 = tableTime.get(3);
    }
}
```

La classe E4cBackground quand appeler elle s'exécute en arrière plan. Cette classe permet d exécuté la méthode e4Csg1MACC_sqlQuery() puis vérifier si la méthode a récupéré les valeurs de température et d'heure, ensuite représenté le graphique avec les valeurs reçu.
Avant de ce connecter a la base de donnée dans la méthode onPreExecute(), elle

va afficher une fenêtre de chargement avec un message suivant "Chargement des graphes...". Voici le la représentation en code:

```
protected void onPreExecute() {

            progressDialog.setMessage("Chargement des graphes...");
            progressDialog.show();

            super.onPreExecute();
```

Pendant la méthode onPreExecute() est en cour, en arrière plan la méthode doInBackground() exécute ma méthode e4Csg1MACC_sqlQuery(). Ci-dessous le code qui fait cela:

```
protected String doInBackground(Void... voids) {
            try {
                e4Csg1MACC_sqlQuery();
            }catch (Exception ex){
                message = "Exceptions......" + ex;
            }
            return message;
        }
```

Quand doInBackground() a terminer exécuter la méthode onPostExeccute() prend le relais, elle vérifie si les valeurs de température et leurs heures ont été récupérer. Je vérifie si un boolean "isSuccess" retourne vrai car le boolean est affecter a faux. Je présente mon boolean a l'état vrai (true) dans ma methode e4Csg1MACC_sqlQuery():

```
isSuccess = true;
```

Ensuite je teste si mon boolean est vrai, si oui je crée le graphique avec les valeurs reçu depuis la base de donnée. Sinon j'affecte un message "Le graphique n'a pas pu être chargé " dans une variable de String "message":

```
if (isSuccess){
       // create the graph
                ...
}else{
    message = "Le graphique n'a pas pu être chargé";
}
```

Si boolean"isSuccess" retourne vrai alors il faut trouver le nom d'idée du graphique dans le ficher xml, qui est affecté a "graph" un objet de GraphView. J'ajoute les valeurs reçu a l'aide d'un objet "series" est initialise avent l'ajout des valeurs, qui prends un argument le nom de la methode ou je place les valeurs. La partie du code :

```
GraphView graph = (GraphView)findViewById(R.id.Graph);
series = new LineGraphSeries<DataPoint>(e4Csg1MACC_getDataPoint());
graph.addSeries(series);
```

Quand je récupère l'heures des températures elles sont dans un format "timestamp". Le "timestamp" en français l'horodatage représentant la date et l'heure est appelée timestamp (de l'anglais time, « heure » et stamp, marquage par un timbre ou un tampon) ou tout simplement « horodatage ». Il peut s'agir

d'une séquence de caractères (groupe date-heure) représentant la date et l'heure sous une forme intelligible. Le format je reçois de la base de donnée "yyyy-mm-dd hh:mm:ss", quatre "y" signifie l'année suivie du mois en deux "m" et le jour en deux "j" et l'heure en double "hh", minute en double "mm" et second en double "ss".
Pour récupérer seulement le format heure et minute j'utilise un "SimpleDateFormat" permet d'afficher seulement l'heure et les minutes sur l'axe horizontale. Le code ci-dessous permet cela:

```
SimpleDateFormat sdf = new java.text.SimpleDateFormat("hh:mm");

...

graph.getGridLabelRenderer().setLabelFormatter(new DefaultLabelFormatter(){
            @Override
            public String formatLabel(double value, boolean isValueX) {
                if(isValueX){
                    return sdf.format(new Date((long)value));
                }
                return super.formatLabel(value, isValueX);
            }
        });
```

Le graphique limite une plage des valeur de température maximum et minimum, Pour terminer l'utilisateur peut agrandir et dézoomer le graphique:

```
graph.getViewport().setScalable(true);
Viewport viewport = graph.getViewport();
viewport.setYAxisBoundsManual(true);
```

-class E4cBackground extends AsyncTask<Void, Void, String>

```
class E4cBackground extends AsyncTask<Void, Void, String>
    {
        @Override
        protected void onPreExecute() {

            progressDialog.setMessage("Chargement des graphes...");
            progressDialog.show();

            super.onPreExecute();
        }

        @Override
        protected String doInBackground(Void... voids) {
            try {
                e4Csg1MACC_sqlQuery();
            }catch (Exception ex){
                message = "Exceptions......" + ex;
            }
            return message;
        }

        @Override
        protected void onPostExecute(String s){
            //if isSuccess is true than represent the graph and warn the
user.....................
            //or warn the user the graph did not
loaded.........................................
            if (isSuccess) {
```

```
                //find my Graph id from the xml
file.............................................
                //create an object series
.....................................................
                //add the points using series and show on
graph.....................................
                GraphView graph = (GraphView)findViewById(R.id.Graph);
                series = new LineGraphSeries<DataPoint>(e4Csg1MACC_getDataPoint());
                graph.addSeries(series);

                //show the time in a time format hh:mm
        graph.getGridLabelRenderer().setLabelFormatter(new DefaultLabelFormatter(){
            @Override
            public String formatLabel(double value, boolean isValueX) {
                if(isValueX){
                    return sdf.format(new Date((long)value));
                }
                return super.formatLabel(value, isValueX);
            }
        });

                graph.getViewport().setScalable(true);
                Viewport viewport = graph.getViewport();
                viewport.setYAxisBoundsManual(true);

                e4Csg1MACC_getDataPoint();

                message = "Graphique chargé";
            }else{
                message = "Le graphique n'a pas pu être chargé";
            }
            Toast.makeText(getBaseContext(),""+ message,Toast.LENGTH_LONG).show();
            progressDialog.hide();
        }
    }
```

La methode e4Csg1MACC_getDataPoint() retourne les points du graphe a afficher, le code permet de faire cela:

```
public DataPoint[] e4Csg1MACC_getDataPoint() {
    DataPoint[] dp = new DataPoint[]{
            new DataPoint(tempTime1, temperature1),     //new
DataPoint(heur,valeur),
            new DataPoint(tempTime2, temperature2),
            new DataPoint(tempTime3, temperature3),
            new DataPoint(tempTime4, temperature4)
    };
    return (dp);
}
```

Plus tard dans le développement l'équipe et moi avons pensé à montrer les quatre dernières valeurs d'humidité dans un graphique séparé appelé "Graphe d'humidité" qui rend intéressant pour l'utilisateur de savoir le taux d'humidité dans la pièce.
Cette nouvelle graphe modifie la requête SQL pour avoir la température, l'humidité et leur heure. Voici la nouvelle requête SQL:

```
String query = "select TEMPERATURE,HUMIDITE,DATE_JOUR from SALLE_BAT where NOM_BAT
= '"+salleName+"' and DATE_JOUR = '"+timeStamp+"' order by DATE_JOUR";
```

Il est aussi nécessaire de mettre les quatre dernières valeur d'humidité dans un tableau "tableHumi" puis affecter chaque variable humidité au valeur d'humidité situe dans le tableau: voici ci-dessous le nouveau tableau, comment récupéré les quatre valeurs d'humidité :

```
private ArrayList<Double> tableHumi = new ArrayList<Double>();

...
while(rs.next()){
    tableTemp.add(rs.getDouble(1));
    tableHumi.add(rs.getDouble(2));
    tableTime.add(rs.getTime(3));

}

...
humidite1 = tableHumi.get(0);
humidite2 = tableHumi.get(1);
humidite3 = tableHumi.get(2);
humidite4 = tableHumi.get(3);
```

Après avoir affecté mes variables d'humidité, je vais créer un autre graphique avec le même code précédemment avec quelque changement:

```
GraphView graphHumidity = (GraphView)findViewById(R.id.Graph1);
seriesHumi = new LineGraphSeries<DataPoint>(e4Csg1MACC_getDataPointHumi());
graphHumidity.addSeries(seriesHumi);


graphHumidity.getGridLabelRenderer().setLabelFormatter(new DefaultLabelFormatter(){
    @Override
    public String formatLabel(double value, boolean isValueX) {
        if(isValueX){
            return sdf.format(new Date((long)value));
        }
        return super.formatLabel(value, isValueX);
    }
});

graphHumidity.getViewport().setScalable(true);
Viewport viewportHumi = graphHumidity.getViewport();
viewportHumi.setYAxisBoundsManual(true);


public DataPoint[] e4Csg1MACC_getDataPointHumi() {
    DataPoint[] dp = new DataPoint[]{
            new DataPoint(tempTime1,humidite1),   //new DataPoint(heur,valeur),
            new DataPoint(tempTime2,humidite2),
            new DataPoint(tempTime3,humidite3),
            new DataPoint(tempTime4,humidite4)
    };
    return (dp);
}
```

Notamment sur l'écran de "Graphe de Température" et "Graphe d'humidité" il y a un bouton "Annuler" qui permet de revenir sur l'accueil "Accueil", que je déclare ici :

```
private Button button_annuler;
```

Puis j'affecte ma variable "button_annuler" a mon bouton objet "Annuler" dans

"activity_graph.xml" qui a un id "button_annulerGraph". Voici la représentation coder :

```
button_annuler = (Button)findViewById(R.id.button_annulerGraph);
```

Pour représenter l'action qui retourne a l'accueil, j'utilise la méthode "OnClickListener()" dans la librairie "View" d'Android Studio que quand j'appuie mon bouton "Annuler" la méthode e4Csg1MACC_annulerGraph() s'execute dans un "setOnClickListener()".
Le code qui représenter l'action retourner a l'accueil :

```
button_annuler.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        e4Csg1MACC_annulerGraph();
    }
});
```

4.2.4 COMMUNICATION AVEC ESP32

Une fois utilisateur sélectionne un climatiseur dans l'Accueil et glisse son doigt de droite a gauche sur l'écran, il se dirige a une fenêtre qui a les boutons de contrôle d'un climatiseur classique et demander dans le cahier décharge (Marche, Arrêt, Up, Down) avec d'autres boutons comme pour accédé au graphe de température, d'humidité et un boutons de sortie (déconnecter).
L'utilisateur rentre dans une salle, il se connecte au borne Wi-Fi de l'ESP Salle (ESP32). Puis une fois connecté il s'identifie sur l'application. Ensuite il sélectionne un ou tous les climatiseurs de la liste dans l'Accueil. Ensuite accéder a la Télécommande et enfin il effectue une commande Marche, Arrêt...

D'après mes connaissances et une discussion avec mon collège qui gère l'ESP32, nous arrivons a une conclusion d'utiliser les Sockets pour la communication entre application (client) et ESP (serveur).

Une socket est une extrémité d'une liaison de communication lien entre deux programmes s'exécutant sur le réseau. Une socket est liée à un numéro de port afin que la couche TCP puisse identifier l'application à laquelle les données sont destinées.
Un point de terminaison est une combinaison d'une adresse IP et d'un numéro de port. Chaque connexion TCP peut être identifiée de manière unique par ses deux extrémités. De cette façon, vous pouvez avoir plusieurs connexions entre le client et le serveur.

J'ai un message que je veux passer a l'ESP avec son adresse IP **"192.168.4.1** a l'écoute sur le port 1060.

Au plus simple on veut envoyer une socket qui aura un ordre x. Ainsi formulé une class E4sendSocket extands AsyncTask :

```
private class E4sendSocket extends AsyncTask<Void, Void, Void>{}
```

Voici le code socket permet d'envoyé vers l'ESP:

```
private class E4sendSocket extends AsyncTask<Void, Void, Void>
{
    @Override
    protected Void doInBackground(Void... voids) {
        Character b = (char) commandMessage;
            try {
            String host = "93.121.180.47"; //93.121.180.74  192.168.4.1
            int port = 1060;
            s = new Socket(host, port);
            pw = new PrintWriter(s.getOutputStream());
            pw.write(b);
            pw.flush();
            pw.close();
            s.close();
            Log.e("SOCKET int:", String.valueOf(commandMessage));
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

4.2.4.1 MESSAGE ENVOYER DEPUIS L'UTILISATEUR

Avant l'envoie d'une commande a l'ESP salle mon collègue et moi c'est mis d'accord sur quel type de message a envoyer et son contenu. L'application va envoyer un message de type char qui contient par exemple 26, car si l'application envoi un message de type integer 26 l'ESP salle va recevoir soit un symbole carré ou un point d'interrogation. L'envoi d'un message String 26 l'ESP salle reçois le message "2" a la ligne "6", tandis que l'envoie d'un char "26" de l'application l'ESP salle reçois "26". Envoyer un char me limite le contenu du message entre un plage de (-128 +127).En gros je vais casté le message envoyer soit integet en char avant d'envoyer la socket, le code ci-dessous permet de casté le message envoyer:

```
Character b = (char) commandMessage;
```

Maintenant le contenu du message envoie va dépendre sur quel climatiseur l'utilisateur a sélectionné, mais avant de procédé mon collègue m'a averti que le nombre maximum d'équipement autorisé a se connecter sur l'ESP salle est 4. Alors nous avons décidé que un ou deux utilisateur(s) puise contrôler deux climatiseurs max dans une salle.

Trois tableaux suivant démontre les commandes de chaque climatiseur et le message envoyé a l'ESP salle:
Pour Climatiseur 1

Commande/ Température	Message avant l'envoi	Message envoyer
Marche	100	10
Arret	101	11
16	116	12
17	117	13
18	118	14
19	119	15

20	120	16
21	121	17
22	122	18
23	123	19
24	124	20
25	125	21
26	126	22
27	127	23
28	128	24
29	129	25
30	130	26

Pour Climatiseur 2

Commande/ Température	Message avant l'envoi	Message envoyer
Marche	200	50
Arret	201	51
16	216	52
17	217	53
18	218	54
19	219	55
20	220	56
21	221	57
22	222	58
23	223	59
24	224	60
25	225	61
26	226	62
27	227	63
28	228	64
29	229	65
30	230	66

Pour tous les Climatiseurs

Commande/ Température	Message avant l'envoi	Message envoyer
Marche	300	100
Arret	301	101
16	316	102
17	317	103
18	318	104
19	319	105
20	320	106
21	321	107
22	322	108
23	323	109
24	324	110
25	325	111
26	326	112
27	327	113

28	328	114
29	329	115
30	330	116

4.2.4.2 RECUPERER LE NOMBRE DE CLIMATISEUR

L'application envoie automatiquement un message vers l'ESP salle sans l'aide de l'utilisateur. Cette action se fait quand l'utilisateur arrive a chaque fois sur l'interface d'Accueil, automatique un message envoyé a l'ESP salle pour avoir le nombre d'équipement actuellement connecter puis par la suite de soustraire de un pour avoir le nombre de climatiseur connecter.
Tout d'abord formuler une méthode :

```
private void e4Csg1MACC_getClimFromESP()
```

Dans cette méthode une socket va envoyer un message "0" a l'ESP salle, puis l'application va recevoir le nombre d'équipement connecté, ce nombre est affecté dans une variable global nommé "nbClim". Avant de fermer la socket je soustrait nbClim de 1 pour me retirer des équipements connecté.Tant que nbClim est égale a "-1" la socket est de nouveau envoyé a l'ESP salle. Le code suivant permet cela:

```
try {
    int getnbClim =0;
    nbClim = -1;
    while (nbClim == -1) {
        Socket s;
        s = new Socket("192.168.4.1", 1060);
        PrintStream p = new PrintStream(s.getOutputStream());
        Character val = (char) getnbClim;
        p.println(val);

        //accept the resultat
        InputStreamReader isr = new InputStreamReader(s.getInputStream());
        nbClim = isr.read();
        nbClim--;
        p.close();
        isr.close();
        s.close();
        getClim = true;
    }
}catch (IOException  e){
    e.printStackTrace();
}
```

4.2.5 AFFICHER LES CLIMATISEURS

Une fois récupéré le nombre de climatiseur disponible, il faut afficher ces climatiseurs qui donne l'utilisateur le choix de sélectionner et savoir quel message (commandMessage) l'application envoi a l'ESP Salle. Dans l'interface d'Accueil il y a un menu déroulant qui contient le nom de climatiseur, l'image ci-dessous montre le résulta obtenu.

Procédé ainsi une méthode qui gère d'affecter les noms dans la liste déroulant, `private void e4Csg1MACC_showClim().`
J'ajoute dans une liste de tableau "arryList" une phrase qui explique l'utilisateur de

choisir un climatiseur, cette phrase est la suivante "Veuillez appuyer pour sélectionner un climatiseur". Mais avant tout il faut initialiser la liste de tableau, voir code :

```
arrayList = new ArrayList<String>();
arrayList.add("Veuillez appuyer pour sélectionner un climatiseur");
```

Ensuite je récupère le nombre de climatiseur "`nbClim`" que l'ESP Salle ma fourni l'hors du démarrage de l'activité expliqué précédemment. Une boucle est utiliser pour tant que le nombre de climatiseur est plus petit ou égale a une variable défini " i " de type integer initialiser a 1. La boucle ajoute dans la liste de tableau un texte définie avec le numéro " i ".
Le code ci-dessous permet de réaliser ceci:

```
for (int i = 1; i <= nbClim; i++) {
    arrayList.add("Climatiseur " + i);
}
arrayList.add("Tous les climatiseurs");
```

Dans la méthode le"Spinner" dans le ficher xml est affecté a un objet Spinner ce qui me permet par la suite d'ajouter les nom de climatiseur avec l'aide un "ArrayAdapter" et d'une liste de tableau "arrayList". un "ArrayAdapter" permet de réaliser la liste déroulant et d'ajouter les textes situer dans le tableau au "Spinner".

```
Spinner spinner = findViewById(R.id.spinner_clim);

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_dropdown_item, arrayList);

spinner.setAdapter(adapter);
```

La liste déroulant des climatiseurs doit interagir avec l'utilisateur maintenant en exécuter la méthode "setOnItemSelectedListener". Il permet a l'utilisateur de passer a la l'interface de la télécommande en glissant le doigt de droite a gauche, sinon un message s'affiche et expliquera de selectionne une clim avant d'utiliser la télécommande ce qui est logique. Ainsi le code :

```
//when spinner selected an item in the list
spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long
id) {
        selectedAC = parent.getItemAtPosition(position).toString();
        nubSelectedAC = parent.getSelectedItemPosition();

        if(selectedAC.equals("Veuillez appuyer pour sélectionner un climatiseur")){
            Toast.makeText(getBaseContext(), "Veuillez sélectionner un
climatiseur", Toast.LENGTH_SHORT).show();
        }
        if(!selectedAC.equals("Veuillez appuyer pour sélectionner un
climatiseur")){
            goodClimInfo = true;
            Toast.makeText(getBaseContext(), "Climatiseur sélectionner : " +
selectedAC, Toast.LENGTH_SHORT).show();

        }
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {
        // TODO Auto-generated method stub
    }
});
```

4.2.6 PASSAGE ENTRE L'INTERFACE ACCUEIL ET TELECOMMANDE

Une fois utilisateur sélectionne un climatiseur dans l'Accueil, il glisse son doigt de droite a gauche sur l'écran pour démarier l'activité de la télécommande.

4.2.7 DECONNECTION

Un bouton de déconnection situer au pied de page de l'application dans la section noir.
Cette déconnection quitte l'activité actuel où se situe l'utilisateur puis revenir a l'identification où l'utilisateur devra entrer de nouveau son identifiant et son mot de passe.
Avant de procéder une déconnection avec succès, il faut attribuer un objet de bouton avec celle dans le ficher xml montrer ci-dessous

```
private FloatingActionButton fabExit;

fabExit = (FloatingActionButton)findViewById(R.id.floatingActionButton_homeExit);
```

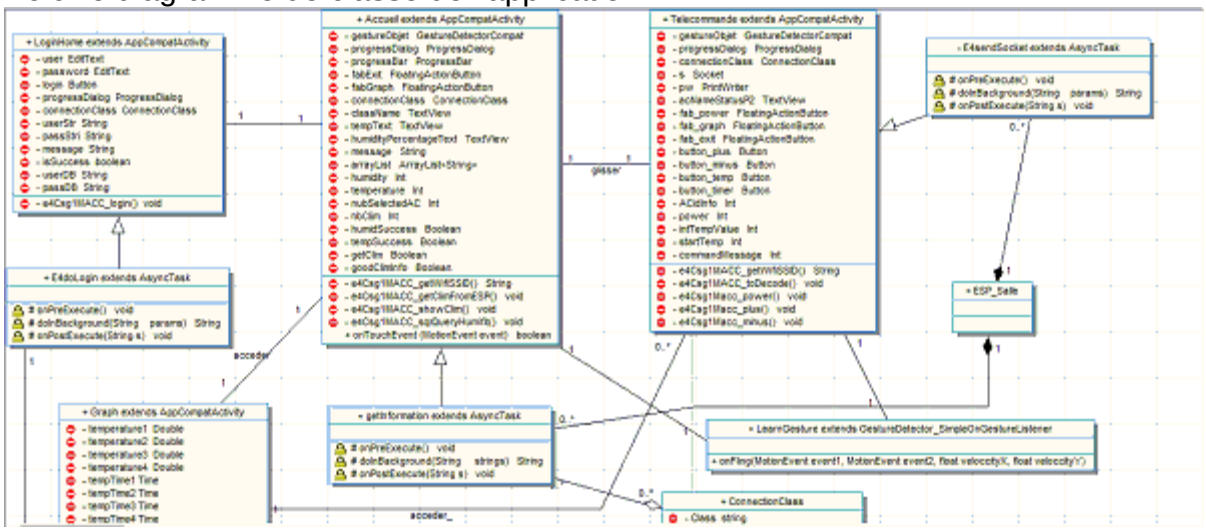
Ensuite mettre un onClickListener sur mon objet du bouton, ce qui permet exécuter les taches a chaque fois ce bouton est appuie

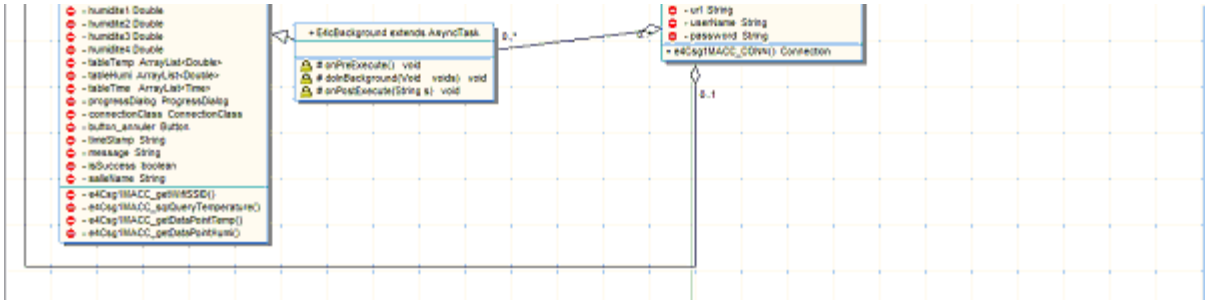
```
fabExit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(Home.this,LoginHome.class);
        startActivity(intent);
        finish();
        System.exit(0);

    }
});
```

La méthode "finish()" suivi "System.exit(0)" est appelé pour terminer (détruire) l'activité actuelle quand l'utilisateur est sur l'interface d'identification.

4.3 DIAGRAMME DE CLASSE

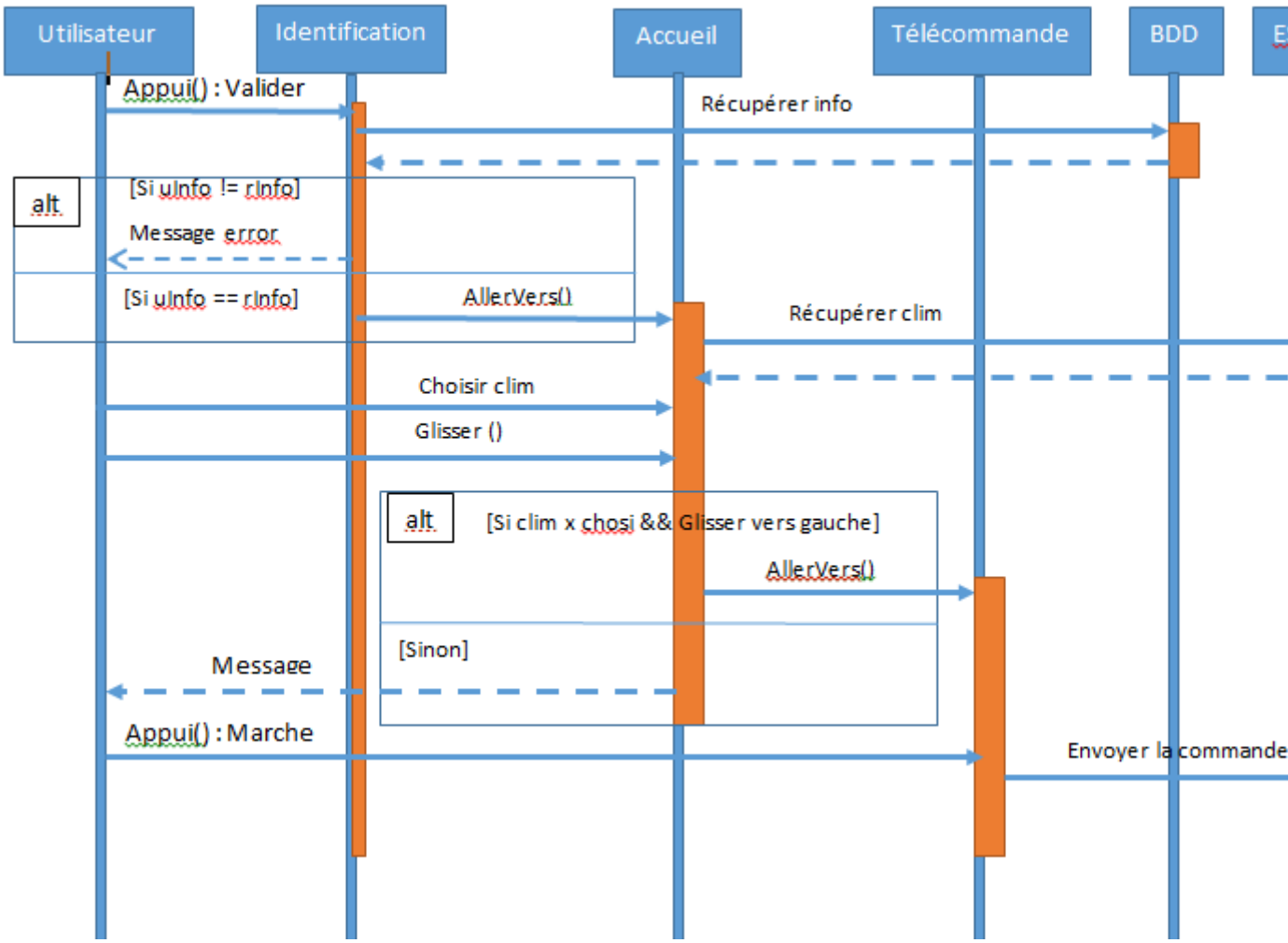
Voici le diagramme de classe de l'application





4.4 DIAGRAMME DE SEQUENCE

Ce diagramme de Séquence illustre le fonctionnement principal d'allumer le climatiseur.
*uInfo : Information de l'utilisateur
*rInfo : Information récupéré



5 PLANIFICATION

6 AMELIORATION POSSIBLE