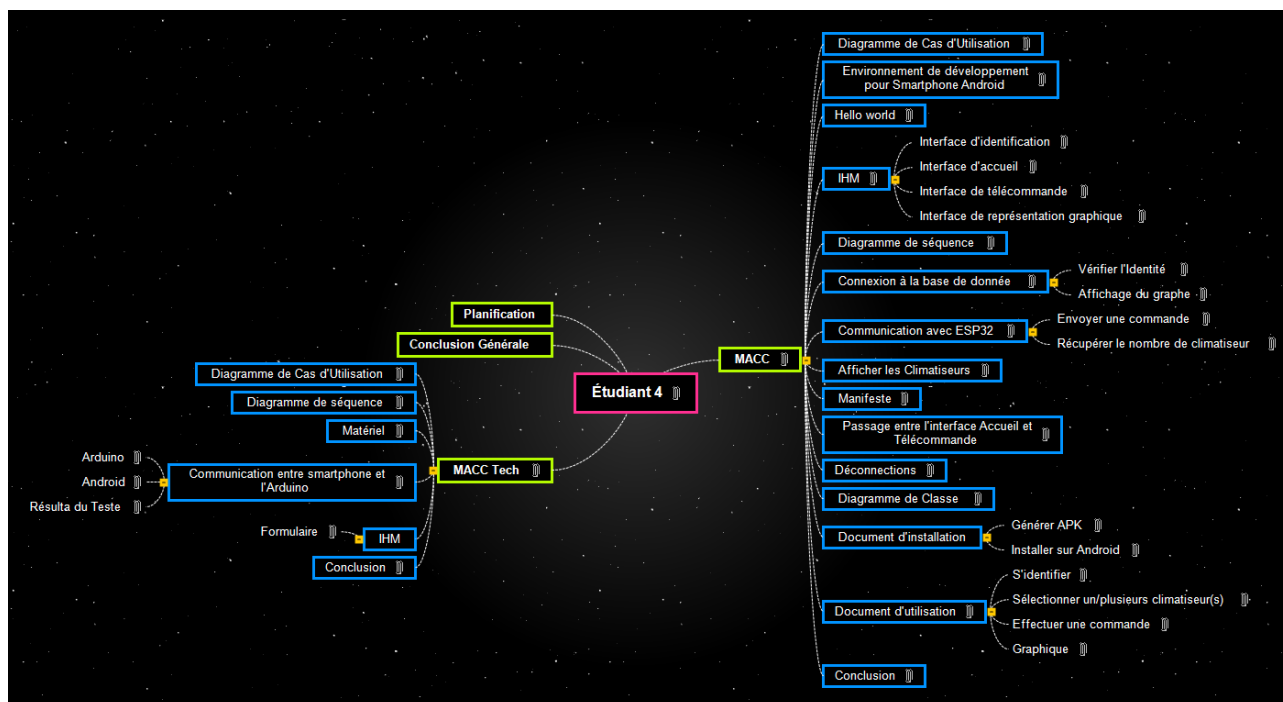


# ÉTUDIANT 4



## MACC 3

Diagramme de Cas d'Utilisation .....	3
Environnement de développement pour Smartphone Android .....	3
Hello world .....	5
IHM .....	7
Interface d'identification .....	7
Interface d'accueil .....	8
Interface de télécommande .....	8
Interface de représentation graphique .....	9
Diagramme de séquence .....	9
Connexion à la base de données .....	10
Vérifier l'Identité .....	11
Affichage du graphe .....	16
Communication avec ESP32 .....	23
Envoyer une commande .....	24
Récupérer le nombre de climatiseur .....	26
Afficher les Climatiseurs .....	27
Manifeste .....	28

Passage entre l'interface Accueil et Télécommande .....	30
Déconnexions.....	32
Diagramme de Classe .....	32
Document d'installation .....	33
Générer APK.....	33
Installer sur Android .....	34
Document d'utilisation .....	35
S'identifier .....	35
Sélectionner un/plusieurs climatiseur(s) .....	36
Effectuer une commande .....	37
Graphique.....	38
Conclusion .....	38
<b>MACC Tech 39</b>	
Diagramme de Cas d'Utilisation .....	39
Diagramme de séquence .....	39
Matériel .....	40
Communication entre smartphone et l'Arduino .....	41
Arduino .....	41
Android .....	45
Résultat du Test.....	50
IHM.....	53
Formulaire.....	53
Conclusion .....	53
<b>Conclusion Générale .....</b>	<b>54</b>
<b>Planification 54</b>	

Réalisation d'une Interface Homme Machine (IHM) pour appareil mobile qui permet de rejouer les trames IR pour marche/arrêt, up/down d'un climatiseur. Dans l'avancement du projet l'utilisateur a l'option de visualiser la température avec son smartphone. Avant de procédé d'allumer, éteindre et changer la température, il y a différent tache a maître en oeuvre sont :

Choisir l'environnement de développement,

Représenter l'application,

Obtenir une connexion avec la base de donnée,

Communiquer avec ESP,

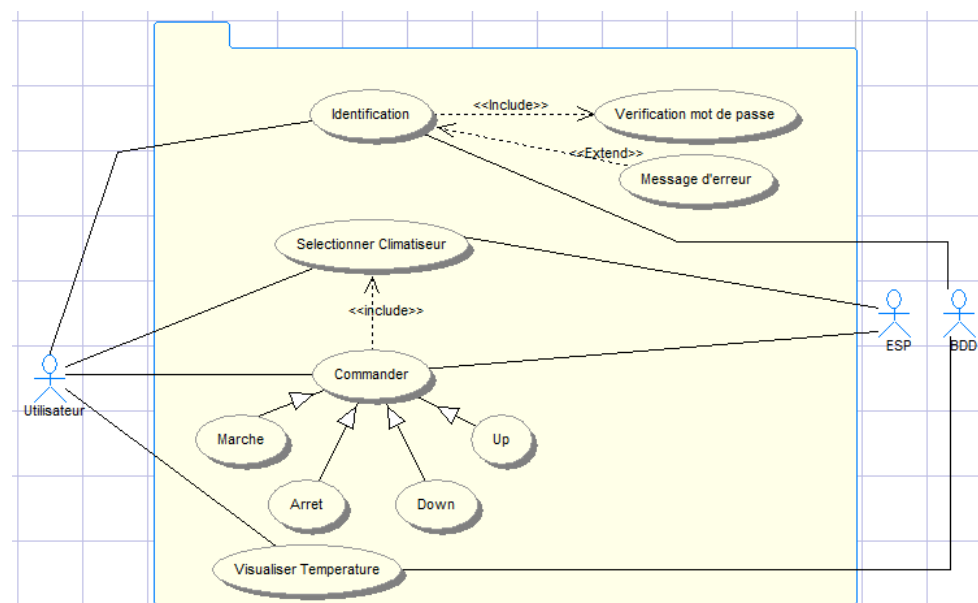
Représenter un graphique avec les quatre dernières température enregistre dans la basse de donnée.

## 1 MACC

MACC est une application pour les utilisateurs. MACC permet d'envoyer une commande pour allumer, éteindre, augmenter ou diminuer la température. L'utilisateur peut aussi visualiser la température de la salle si il souhaite.

### 1.1 DIAGRAMME DE CAS D'UTILISATION

Voici une vision globale du diagramme de cas d'utilisation MACC.



### 1.2 ENVIRONNEMENT DE DEVELOPPEMENT POUR SMARTPHONE ANDROID

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 créé le 7 Novembre 2001 qui contient un « workspace » et des extensions de plug-in. Les programmes d'Eclipse sont plus souvent écrits en Java pour développer des applications de Java, mais Eclipse peut développer des applications sur d'autres langages de programmation par des plug-ins qui incluent : C, C++, C#, Java, JavaScript, Perl, Php, Python et d'autres 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 JDK est 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:

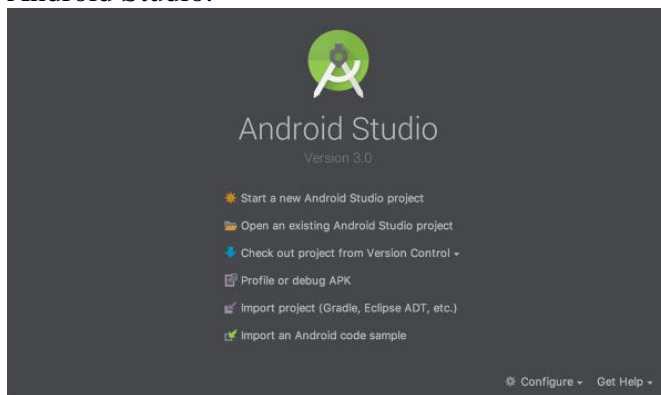
- \*La simplicité et sa gratuité
- \*Système flexible de Gradle-based build
- \*Générer plusieurs fichiers apk
- \*Modèles de code pour vous aider à créer des fonctionnalités d'application courantes
- \*Éditeur de mise en page riche avec prise en charge de l'édition de thèmes par glisser-déposer
- \*Prise en charge intégrée de Google Cloud Platform, facilitant l'intégration de Google Cloud Messaging et App Engine

## 1.3 HELLO WORLD

---

Après l'installation Android Studio pour la première fois, je vais créer un nouveau projet Android avec Android Studio et décrire certains des fichiers du projet.

1 - Dans la fenêtre Bienvenue dans Android Studio, cliquez sur Démarrer un nouveau projet Android Studio.



2 - Dans la fenêtre Créer un nouveau projet, j'entre:

Nom de l'application: "Ma première application"

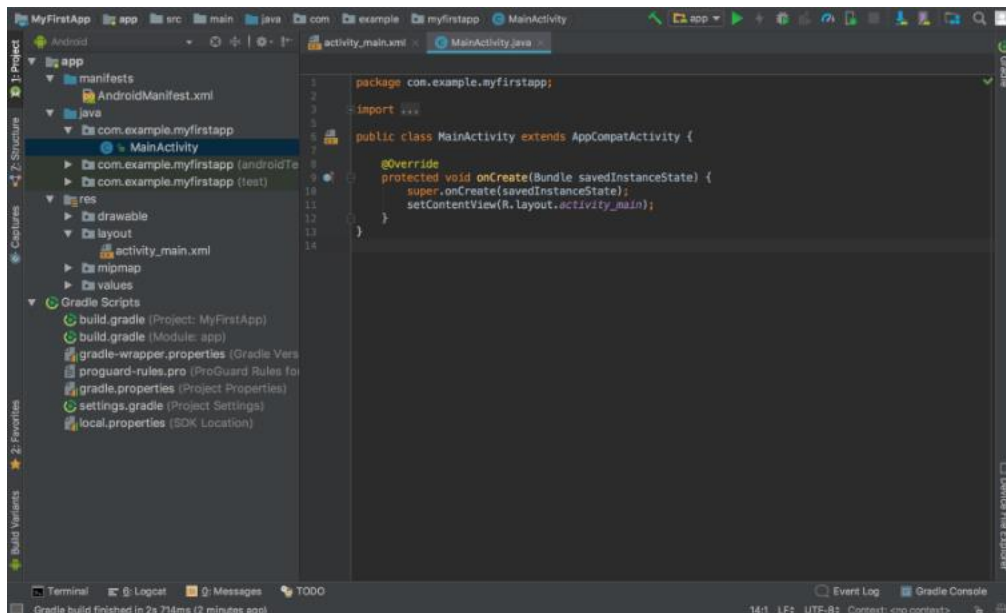
Domaine de la société: "example.com"

3 - Cliquez sur Suivant.

4 - Dans l'écran Target Android Devices, conservez les valeurs par défaut et cliquez sur Next.

5 - Dans l'écran Ajouter une activité à un mobile, sélectionnez Activité vide et cliquez sur Suivant.

6 - Dans l'écran Configurer l'activité, conservez les valeurs par défaut et cliquez sur Terminer.



app> java> com.example.myfirstapp> MainActivity

C'est l'activité principale (le point d'entrée de votre application). Lorsque vous créez et exécutez l'application, le système lance une instance de cette activité et charge sa mise en page.

application> res> mise en page> activity\_main.xml

Ce fichier XML définit la disposition de l'interface utilisateur de l'activité. Il contient un élément TextView avec le texte "Hello world!".

app> manifestes> AndroidManifest.xml

Le fichier manifeste décrit les caractéristiques fondamentales de l'application et définit chacun de ses composants.

Scripts de Gradle> build.gradle

Vous verrez deux fichiers avec ce nom: un pour le projet et un pour le module "app". Chaque module a son propre fichier build.gradle, mais ce projet n'a actuellement qu'un seul module. Vous travaillerez principalement avec le fichier build.gradle du module pour configurer la façon dont les outils Gradle compilent et construisent votre application.

Connectez votre appareil à votre machine de développement à l'aide d'un câble USB. Si vous développez sous Windows, vous devrez peut-être installer le pilote USB approprié pour votre périphérique.

Activez le débogage USB dans les options du développeur comme suit.

D'abord, vous devez activer les options du développeur:

Ouvrez l'application Paramètres.

(Uniquement sur Android 8.0 ou supérieur) Sélectionnez Système.

Faites défiler vers le bas et sélectionnez À propos du téléphone.

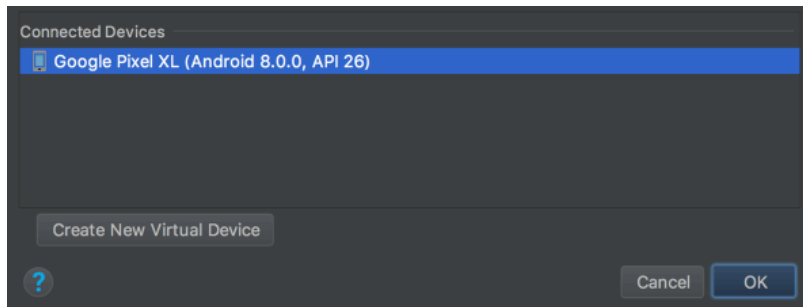
Faites défiler vers le bas et appuyez sur Construire le numéro 7 fois.

Revenez à l'écran précédent pour trouver les options de développeur près du bas.

Ouvrez les options du développeur, puis faites défiler vers le bas pour rechercher et activer le débogage USB.

Exécutez l'application sur votre appareil comme suit:

Dans Android Studio, cliquez sur le module d'application dans la fenêtre Projet, puis sélectionnez Exécuter> Exécuter (ou cliquez sur Exécuter dans la barre d'outils). Dans la fenêtre Sélectionner une cible de déploiement, sélectionnez votre périphérique, puis cliquez sur OK.



Android Studio installe l'application sur votre appareil connecté et la démarre. Vous devriez maintenant voir "Hello World!" s'affiche dans l'application qui s'exécute sur votre appareil.

## 1.4 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.

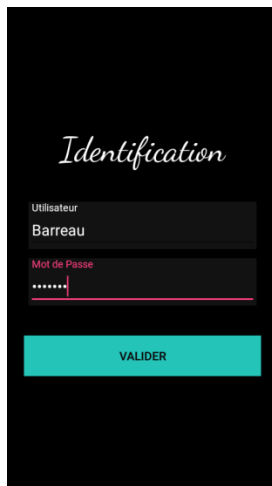
L'interface homme-machine (IHM) est l'interface utilisateur qui relie l'opérateur au dispositif de commande d'un système industriel.

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.

### 1.4.1 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 employé) qui utilise l'application, alors j'ai rajouter un espace d'identification qui sera la premier page avant d'utiliser l'application.



## 1.4.2 INTERFACE D'ACCUEIL

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

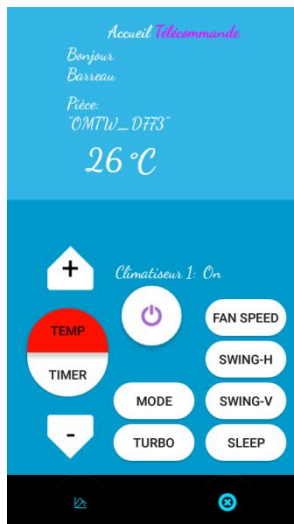


## 1.4.3 INTERFACE DE TELECOMMANDE


Arriver à cette interface il faut que l'utilisateur sélectionne un ou tous les climatiseur(s) disponible dans l'Accueil.

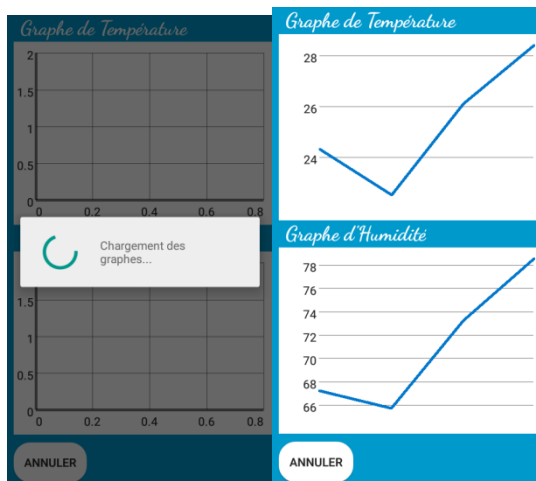
Dans l'interface de Télécommande se situent les actions d'allumer, d'arrêter le climatiseur et d'augmenter ou diminuer la température. L'utilisateur peut observer la température envoyée au climatiseur.





## 1.4.4 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). Dans l'avancement du projet il est intéressant d'afficher un graphe d'humidité pour l'utilisateur.



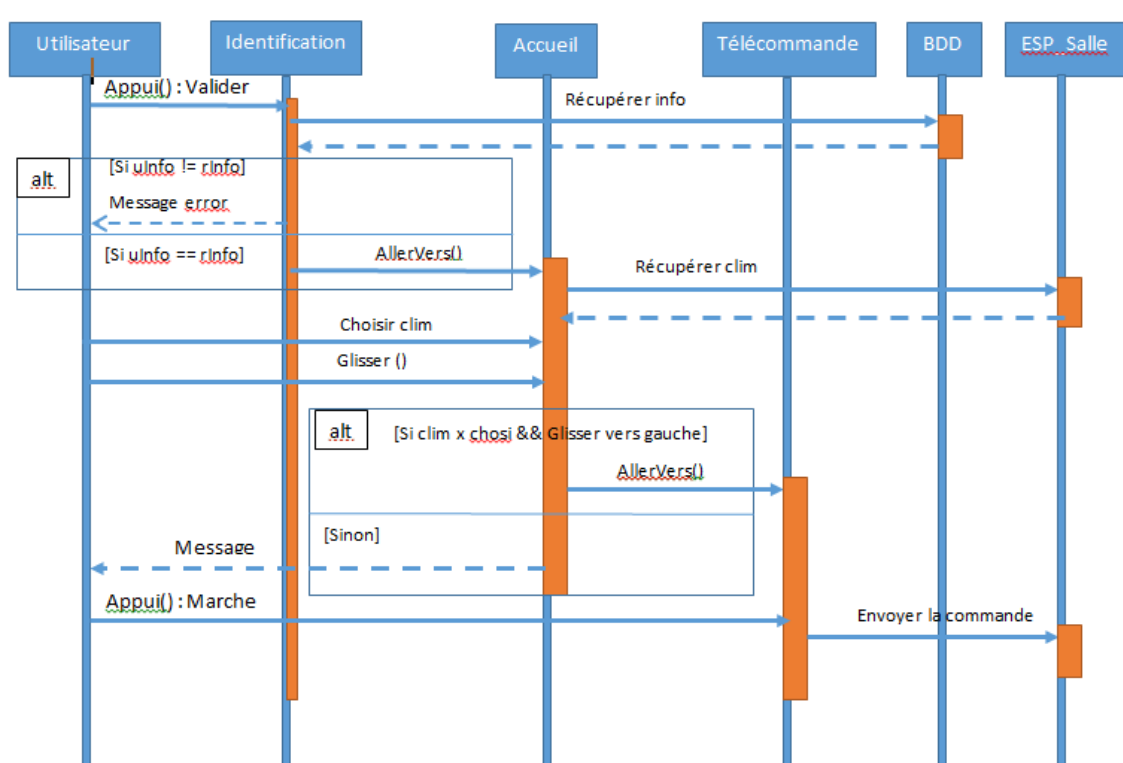
## 1.5 DIAGRAMME DE SEQUENCE

Ce diagramme de Séquence illustre le fonctionnement principal d'allumer le climatiseur.

\*uInfo : Information de l'utilisateur

\*rInfo : Information récupéré

Bien évidemment avant d'appuyer sur Valider, l'utilisateur devra entrer son identifiant et son mot de passe.



## 1.6 CONNEXION A LA BASE DE DONNEE

La connexion à la base de données est importante car elle permet de récupérer des informations critiques du bon fonctionnement de l'application.

\*Comparer l'identifiant que l'utilisateur a entré avec celle dans la base de données.

\*Récupérer les quatre dernières températures et humidité enregistré dans la base de données.

Pour obtenir une connexion à la base de données. Je dois analyser pas à pas:

- Qu'est ce que j'utilise qui m'aide à connecter à la base ?,
- Besoin de la configuration de la base de données (l'adresse IP, nom de la base de données, identifiant de la base de données),
- Appeler une connexion dans un ou plusieurs fichiers.

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 développée sur android studio en java il est recommandé d'utiliser un pilote MySQL Connector/J qui fournit une connectivité pour les applications client. MySQL Connector/J implémente l'API Java Database Connectivity (JDBC).

Ensuite je cherche à savoir de la part de mon collègue le nom de la base de données et l'adresse IP qu' il a configuré:

- L'adresse IP de la base de donnée ("**192.168.137.127**")
- le nom de la base de données ("**MACC1**")
- Identification base de données ("**pi**")
- Mot de passe pour accéder à la base de données ("**Simconolat**")

Avec ces informations obtenu de mon collègue, je crée un fichier ConnectionClass.java avec une classe ConnectionClass, j'utilise un prototype **public** Connection e4Csg1MACC\_CONN() qui permettra à d'autres classes d'appeler cette méthode en cas besoin.

Dans la méthode il aura un objet de Connection conn qui sera initialisé par l'adresse IP de la base de données suivie du nom de la base de données, puis les identifications (username, mot de passe) pour sécuriser la connexion.

Avoir une connexion de la base de données permet à plusieurs méthodes dans plusieurs fichiers java qu'ils puissent avoir une connexion pour exécuter la tâche X.

Chaque fichier qui a besoin de récupérer et/ou comparer des valeurs dans la base de données, créer un objet de Connection avec un accès à la méthode e4Csg1MACC\_CONN().

### 1.6.1 VERIFIER L'IDENTITE

---

J'ai une application qui nécessite de comparer l'information saisie par l'utilisateur avec les informations pré-enregistrées par l'administration dans la base de données, et qui donnerait l'accès à l'utilisateur d'utiliser l'application dans son intégralité.

Tout d'abord, lorsque l'utilisateur ouvrira l'application il y aura une fenêtre d'identification. Il devra ensuite saisir les informations requises par l'administration (nom d'utilisateur, mot de passe) et appuyer sur le bouton « Valider ». Les informations saisies vont être comparées avec celles stockées dans la base de données. Si les informations comparées sont identiques,

l'utilisateur aura accès à la fenêtre suivante, sinon un message d'erreur s'affichera, informant l'utilisateur.

Voici les différentes étapes de la validation d'identité :

- Récupérer l'information saisie par l'utilisateur
- Connexion à la base de données
- Chercher les informations saisies si elles existent
- Valider ou non l'accès

L'utilisateur écrit ces informations son identifiant et son Mot de Passe dans deux « EditText » user et pass que je déclare dans xml et j'initialise a l'aide de la méthode **findViewById**. « user » récupère l'identifiant de l'utilisateur et « pass » qui récupère son mot de passe. La variable usernameDB récupère le nom de l'utilisateur. Voici le code ci-dessous qui permet de faire cela.

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

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'initialise 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 identifiants 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'exécution du SQL Statement et enfin un objet ResultSet qui donne le résultat de la requête SQL. Avec le résultat de la requête je cherche si l'utilisateur a bien saisi les informations ou sinon les informations sont incorrectes 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 abstraite qui hérite de la classe AsyncTask :

```
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 exist) 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 LOGIN= '"+userStr+"' and MDP =
            '"+passStr+"'";
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);

            while (rs.next()) {
                usernameDB = getString(2)
                userDB = rs.getString(4);
                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'est-ce 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 3 étapes, appelées onPreExecute, doInBackground et onPostExecute. L'étape de la méthode onProgressUpdate permet de mettre à jour la progression de la tâche en cours d'exécution. Elle est appelée à l'aide de la fonction publishProgress. Ces étapes sont des méthodes d'AsyncTask.

Quand la **class** E4doLogin **extends** AsyncTask<String,String,String> s'exécute, il y a 3 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 signale 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 final un message s'affiche tout dépend si l'échec de la connexion, l'état d'identification. Si les informations trouvées sont 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);
```

Le bouton dans "activity\_login\_home.xml":

```
<Button
    android:id="@+id/valider"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="30dp"
    android:background="#24c4b9"
    android:text="Valider"
    android:textColor="#0b0b0b"
    tools:ignore="HardcodedText" />
```

Pour représenter l'action qui donne l'accès a l'accueil, j'utilise la méthode "OnClickListener()" 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
            "e4CsglMACC_get_db_Data()".....
            e4CsglMACC_get_db_Data();
        }
        return message;
    }

    @Override
    protected void onPostExecute(String s) {
        // show the message stored in a
```

```

variable.....
    Toast.makeText(getApplicationContext(),""+ 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", usernameDB);
        startActivity(intent);
    }
    //hide my loading message and
symbol.....
    progressDialog.hide();
}
}

```

## 1.6.2 AFFICHAGE DU GRAPHE

---

J'ai une base 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, la date 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 résultat de la requête SQL. Ce résultat de la requête me permet de récupérer mes différentes données nécessaires (ex: Température et heure) que je stocke dans une variable et j'utilise dans la création du graphe.

Au plus simple je veux une connexion à la base de données 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'heure:

```
private ArrayList<Double> tableTemp = new ArrayList<Double>();  
private ArrayList<Time> tableTime = new ArrayList<Time>();
```

Voici deux méthodes 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 stocke mes valeurs récupérées dans une ArrayList et j'affecte la position de l'ArrayList à ma variable concernée. Exemple: j'ajoute dans "arrayListName" le résultat de ma requête SQL qui se situe à la première position dans la base de données `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 Graphe Température. 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 signale 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 crée ma courbe avec mes valeurs récupérées 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 choisis `LineGraphSeries` la sous-classe de `Série`.

La méthode `e4Csg1MACC_sqlQuery()` permet de vérifier si une connexion à la base de données est réussie, d'exécuter une requête SQL, enregistrer les résultats

(Temperature et temps) dans deux tableaux.

Verifier si il y a une connexion 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,DATE_JOUR from CAPTEUR where NUM_SALLE =  
'"+salleName+"' ORDER BY DATE_JOUR";  
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 CAPTEUR where NUM_SALLE =  
'"+salleName+"' order by DATE_JOUR DESC";  
        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 onPostExecute() 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 fichier xml, qui est affecté a "graph" un objet de GraphView. J'ajoute les valeurs reçu a l'aide d'un objet "series" est initialise avant 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>(e4CsglMACC_getDataPoint());
graph.addSeries(series);

```

Quand je récupère l'heure 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 {
            e4CsglMACC_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>(e4CsglMACC_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);

            e4CsglMACC_getDataPoint();

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

```

La méthode `e4CsglMACC_getDataPoint()` retourne les points du graphe à afficher, le code permet de faire cela:

```
public DataPoint[] e4CsglMACC_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 CAPTEUR where NUM_SALLE = '"+salleName+"' order by DATE_JOUR DESC";
```

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 située 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>(e4CsglMACC_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'exécute dans un "setOnClickListener()".

Le code qui représente l'action retourner a l'accueil :

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

## 1.7 COMMUNICATION AVEC ESP32

Une fois que l'utilisateur sélectionne un climatiseur dans l'Accueil et glisse son doigt de droite à gauche sur l'écran, il se dirige vers une fenêtre où figurent les boutons de contrôle d'un climatiseur classique et demander dans le cahier des charges (Marche, Arrêt, Up, Down) avec d'autres boutons comme pour accéder au graphe de température, d'humidité et un bouton de sortie (déconnecter). Quand l'utilisateur rentre dans une salle, il se connecte à la borne Wi-Fi de l'ESP32 (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 accède

à la Télécommande et enfin il effectue une commande Marche, Arrêt...

D'après mes connaissances et pendant une discussion avec mon collègue qui gère l'ESP32, nous avons opté **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 à l'ESP avec son adresse IP **"192.168.4.1"** à l'écoute sur le port 1060.

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

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

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

```
private class E4sendSocket extends AsyncTask<Void, Void, Void>
{
    @Override
    protected Void doInBackground(Void... voids) {
        Character b = (char) commandMessage;
        try {
            String host = "192.168.4.1"; //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;
    }
}
```

## 1.7.1 ENVOYER UNE COMMANDE

---

Avant l'envoi d'une commande à l'ESP, nous nous sommes mis d'accord mon collègue et moi sur le type de message à envoyer ainsi que son contenu.



L'application va envoyer un message de type char qui contient par exemple 26. 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 integer en char avant d'envoyer la socket, le code ci-dessous permet de casté le message envoyer:

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

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

### 1.7.2 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 à chaque fois sur l'interface d'Accueil. Un message est automatiquement envoyé à l'ESP salle pour connaître le nombre d'équipement actuellement connecté, puis par la suite de soustraire de un pour avoir le nombre de climatiseurs connectés.

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();
}
}

```

## 1.8 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 résultat 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 "arrayList" 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 fichier 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.getSelectedItemId();

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

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

## 1.9 MANIFESTE

---

Chaque projet d'application doit avoir un AndroidManifest.xmlfichier (avec

précisément ce nom) à la racine du jeu de sources du projet . Le fichier manifeste décrit des informations essentielles sur votre application pour les outils de construction Android, le système d'exploitation Android et Google Play.

Parmi beaucoup d'autres choses, le fichier manifeste est nécessaire pour déclarer ce qui suit:

Le nom du package de l'application, qui correspond généralement à l'espace de nom de votre code. Les outils de construction Android l'utilisent pour déterminer l'emplacement des entités de code lors de la construction de votre projet. Lors du conditionnement de l'application, les outils de construction remplacent cette valeur par l'ID de l'application à partir des fichiers de construction Gradle, qui est utilisé comme identifiant d'application unique sur le système et sur Google Play.

Les composants de l'application, qui incluent toutes les activités, services, récepteurs de diffusion et fournisseurs de contenu. Chaque composant doit définir des propriétés de base telles que le nom de sa classe Kotlin ou Java. Il peut également déclarer des fonctionnalités telles que les configurations de périphérique qu'il peut gérer et les filtres d'intention qui décrivent comment le composant peut être démarré.

Les autorisations dont l'application a besoin pour accéder aux parties protégées du système ou d'autres applications. Il déclare également toutes les autorisations que d'autres applications doivent avoir s'ils veulent accéder au contenu de cette application.

Les fonctionnalités matérielles et logicielles requises par l'application, ce qui affecte les appareils pouvant installer l'application depuis Google Play.

Ce qui concerne le "Manifest" de MACC, j'ai ajouté quatre permissions qui autorise l'application d'ouvrir des sockets réseau, d'accéder à des informations sur les réseaux Wi-Fi, d'accéder à des informations sur les réseaux et permet d'accéder à l'emplacement approximatif.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.j_lds.macc">

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

    <application
        android:allowBackup="true"
```

```

        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".LoginHome">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".ConnectionClass" />
        <activity
            android:name=".Home"
            android:noHistory="true" />
        <activity
            android:name=".FullRemote"
            android:noHistory="true" />
        <activity android:name=".Graph" />
    </application>

</manifest>

```

## 1.10 PASSAGE ENTRE L'INTERFACE ACCUEIL ET TELECOMMANDE

---

La classe GestureDetectorCompat détecte divers gestes et événements en utilisant les MotionEvent fournis. Le rappel GestureDetector.OnGestureListener avertira les utilisateurs lorsqu'un événement de mouvement particulier s'est produit. Cette classe ne doit être utilisée qu'avec MotionEvent signalé par toucher.

Une fois utilisateur sélectionne un climatiseur dans l'Accueil, il glisse son doigt de droite à gauche sur l'écran pour démarrer l'activité de la télécommande. Il peut aussi retourner en arrière pour utiliser un autre climatiseur.

Dans les deux activités Accueil et Télécommande je crée un instant gestureObjet de la classe GestureDetectorCompat avec deux argument que GestureDetectorCompat demande, la classe et la méthode que j'exécute dans une classe fille LearnGesture.

```

private GestureDetectorCompat gestureObjet;

gestureObjet = new GestureDetectorCompat(this, new FullRemote.LearnGesture());

```

La méthode **public boolean** onTouchEvent (MotionEvent event) permet de d'avoir le mouvement de l'utilisateur effectuer sur l'écran. Si une valeur de retour true de l'individu sur la méthode <onTouchEvent> indique que utilisateur a géré un événement tactile. Une valeur de retour false transfère les événements dans la pile de vues jusqu'à ce que le contact ait été géré avec succès. Ce mouvement est transmit à la méthode onFling qui prend quatre argument où j'utilise deux qui sont MotionEvent1 et MotionEvent2.

```

public boolean onTouchEvent (MotionEvent event){
    this.gestureObjet.onTouchEvent(event);
    return super.onTouchEvent(event);
}

```

La classe LearnGesture a une extension

GestureDetector.SimpleOnGestureListener et dans la class a une méthode onFling(MotionEvent event1, MotionEvent event2, **float** veloccityX, **float** veloccityY) est inclus depuis Gesture mouvement est transmit a la méthode onFling qui prend quatre argument où j'utilise deux qui sont MotionEvent1 et MontionEvent2.

MotionEvent1 est d'où l'utilisateur a commencer son mouvement sur l'écran et MontionEvent2 est ou il est terminé. Si MontionEvent2 est plus grand que MontionEvent1 sur l'écran signifie que le doigt de l'utilisateur est passé de droite a gauche (aller vers l'interface Télécommande) ou si MontionEvent1 est plus grand que MontionEvent2 sur l'écran signifie que le doigt de l'utilisateur est passé de gauche a droite (aller vers l'interface d'Accueil).

Dans Accueil code ci-dessous:

```

//create the gesture Objet class
private class LearnGesture extends GestureDetector.SimpleOnGestureListener {
    //SimpleOnGestureListener is a lister for what we want to do

    @Override
    public boolean onFling(MotionEvent event1, MotionEvent event2, float
    veloccityX, float veloccityY) {
        if (event2.getX() > event1.getX()) {
            //left
        } else if (event2.getX() < event1.getX()) {
            //right
            if (goodClimInfo){
                Intent intent = new Intent(Home.this, FullRemote.class);
                intent.putExtra("user", nameTextPassStr);
                intent.putExtra("ACName", selectedAC);
                intent.putExtra("climIdInfo", nubSelectedAC);
                intent.putExtra("tempSetInstruction", lastTempInstruction);
                startActivity(intent);
                finish();
                System.exit(0);
            }else{
                Toast.makeText(getBaseContext(), "Veuillez sélectionner un
                AC\navant de procéder...", Toast.LENGTH_SHORT).show();
            }
        }
        return true;
    }
}

```

L'Intent objet permet de passer mes variables (nom de l'utilisateur, nom de la climatiser, la valeur du climatiseur et fixer consigne de la température) a Télécommande java "FullRemote".

L'Intent passe l'utilisateur a l'interface de Télécommande depuis l'Accueil "Home".

Dans Télécommande code ci-dessous:

```

//create the gesture Objet class

```

```

class LearnGesture extends GestureDetector.SimpleOnGestureListener {
    //SimpleOnGestureListener is a listener for what we want to do

    @Override
    public boolean onFling(MotionEvent event1, MotionEvent event2, float
    veloccityX, float veloccityY) {
        if (event2.getX() > event1.getX()) {
            //left
            Intent intent = new Intent(FullRemote.this, Home.class);
            intent.putExtra("user", nameTextPassStr);
            startActivity(intent);
            finish();
            System.exit(0);

        } else if (event2.getX() < event1.getX()) {
            //right
        }
        return true;
    }
}

```

Quand l'utilisateur retourne a l'Accueil il retourne avec son nom et commence l'activité Home "Home".

## 1.11 DECONNECTIONS

---

Un bouton de déconnexion situé en bas de la page de l'application dans la section noire.

Cette déconnexion quitte l'activité actuelle où se situe l'utilisateur ,puis l'autorise à revenir à l'identification où l'utilisateur devra entrer de nouveau son identifiant et son mot de passe.

Avant de procéder une déconnexion avec succès, il faut attribuer un objet de bouton avec celle dans le fichier 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);
    }
});

```

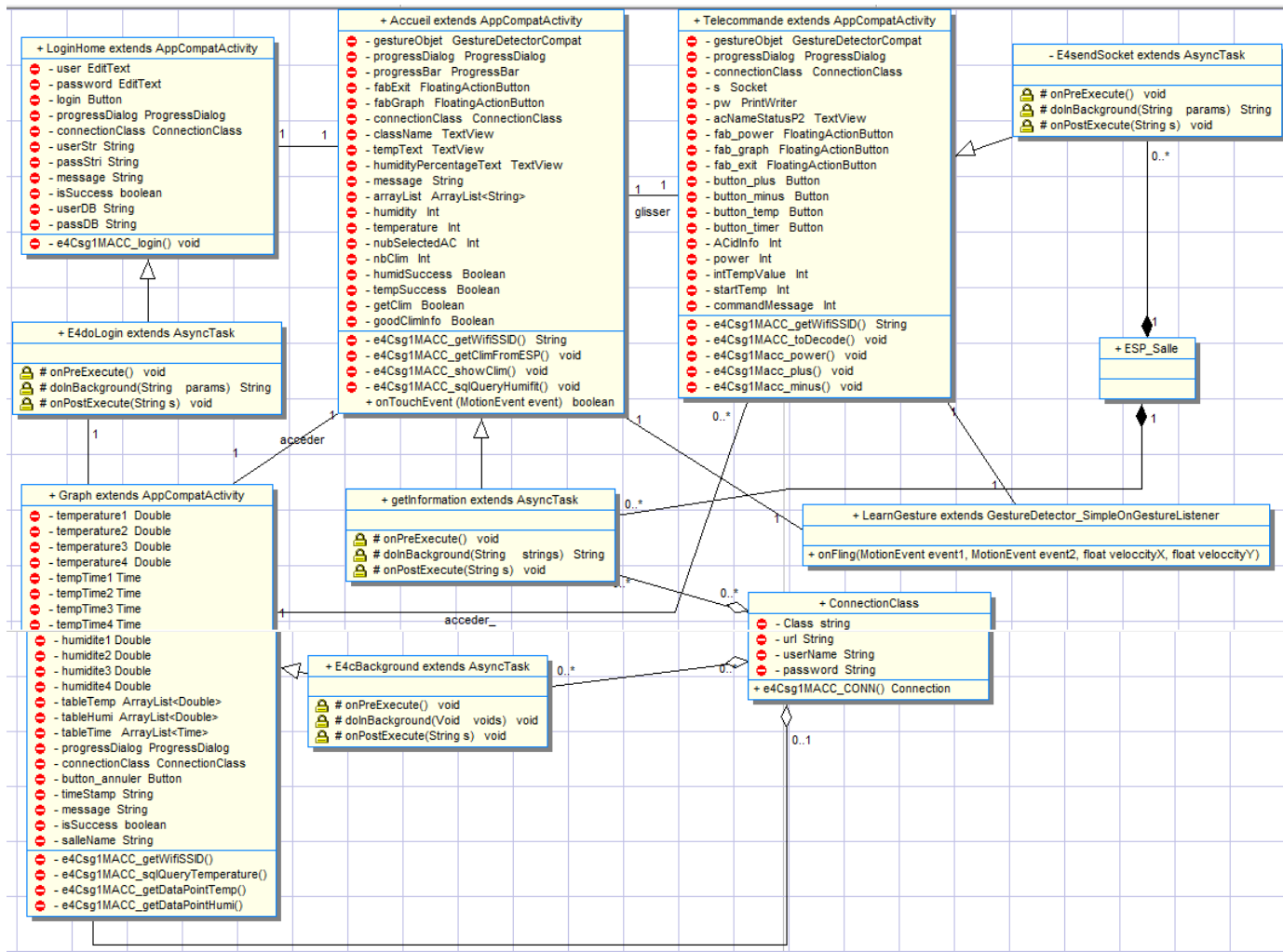
La méthode "finish()" suivi "System.exit(0)" est appelé pour terminer (détruire) l'activité actuelle quand l'utilisateur est sur l'interface d'identification.

## 1.12 DIAGRAMME DE CLASSE

---



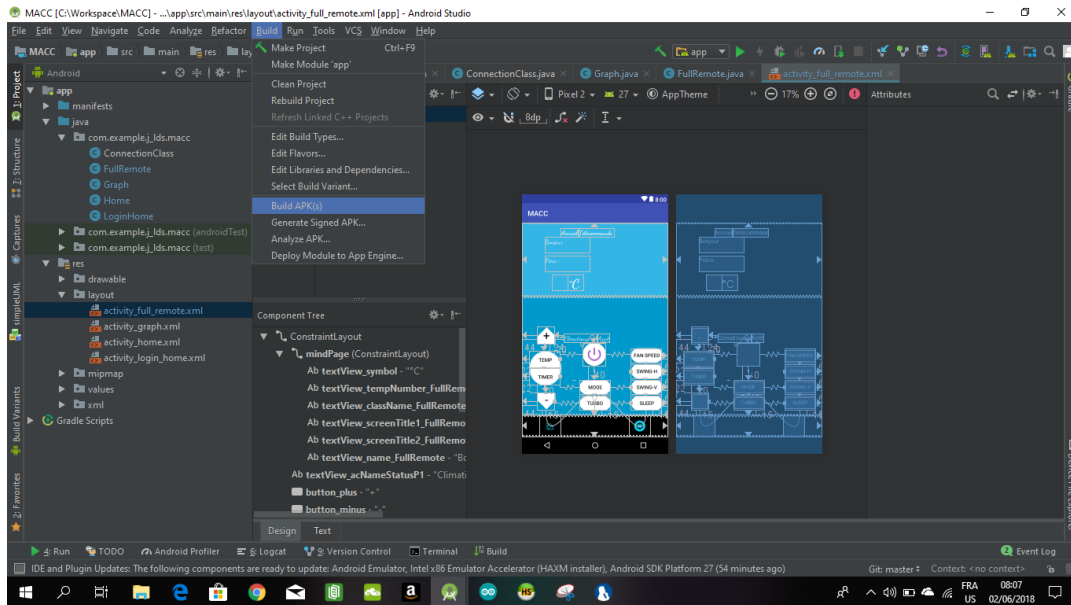
Voici le diagramme de classe de l'application



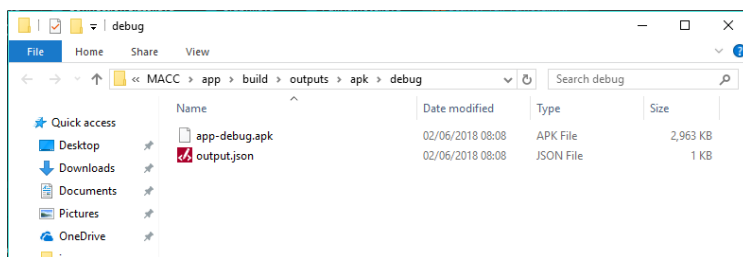
## 1.13 DOCUMENT D'INSTALLATION

### 1.13.1 GENERER APK

Pour générer l'apk depuis l'android studio : Build/ Build apk(s)



Après une fenêtre de notification s'affiche qui informe où se trouve l'apk généré, par défaut elle se situe `nom_du_projet\app\build\outputs\apk\debug` :



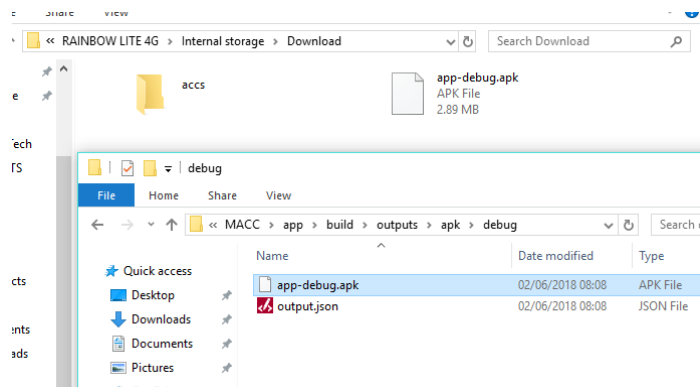
App-debug.apk est l'apk de MACC.

### 1.13.2 INSTALLER SUR ANDROID

Maintenant que l'apk est généré il faut le transmettre dans l'Android.

Connecter l'Android sur l'ordinateur avec le câble usb.

Prenez l'apk et transmettez-le dans le dossier de téléchargement de l'Android.



## 1.14 DOCUMENT D'UTILISATION

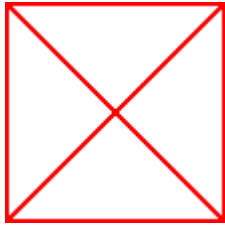
---

Bonjour,

Merci d'avoir installer l'application MACC, nous allons procéder les différents étapes pour d'allumer un climatiseur avec votre smartphone.

Veillez vous connecter a la pièce où vous voulez allumer votre climatiseur.

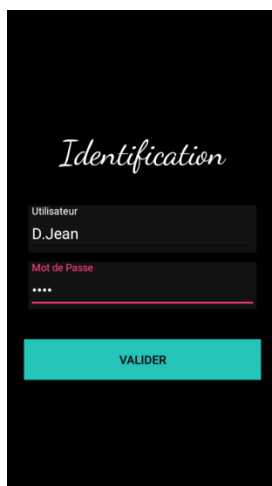
Veillez ouvrir l'application



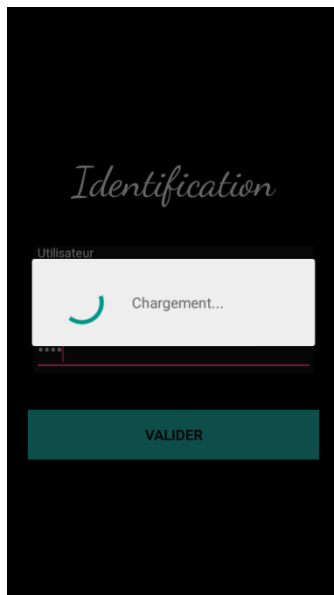
### 1.14.1 S'IDENTIFIER

---

Une fois l'application ouvert entrer votre identifiant et mot de passe.



Puis appuie sur le bouton "Valider".

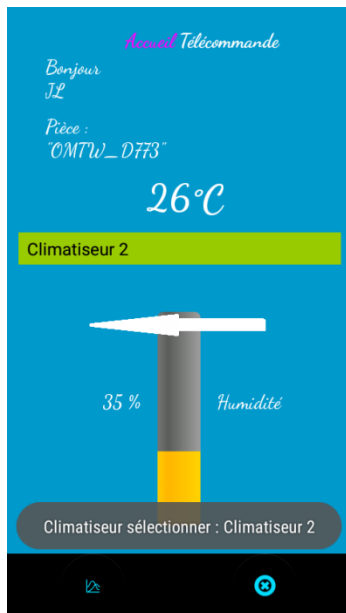


### 1.14.2 SELECTIONNER UN/PLUSIEURS CLIMATISEUR(S)

Après avoir validé votre accès, appuyez sur la liste déroulante et sélectionnez un climatiseur X.

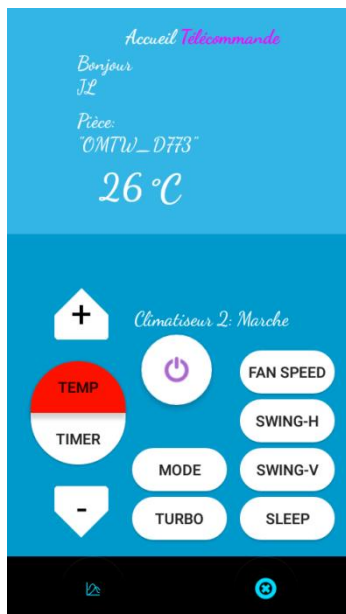


Ensuite glissez votre doigt de droite à gauche sur l'écran.

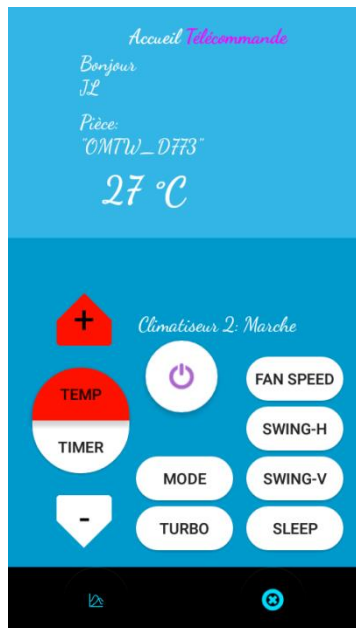


### 1.14.3 EFFECTUER UNE COMMANDE


Vous êtes à l'interface de la télécommande. Le climatiseur sélectionné est éteint, veuillez appuyer sur **TEMP** ce qui vous donne la possibilité de commander le climatiseur. Puis appuyer sur le bouton puissance subitement l'état du climatiseur change de **Arrêt** à **Marche**.

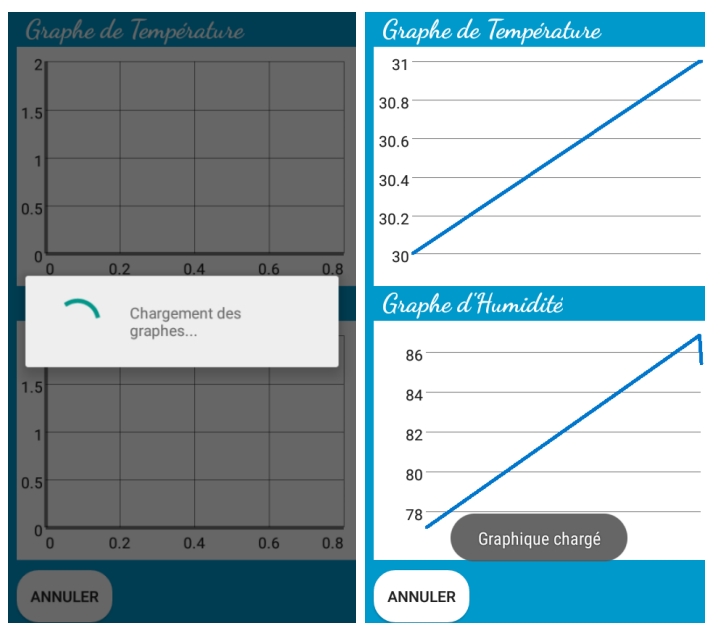


Avec le bouton **TEMP** appuie avec la couleur rouge, vous pouvez changer la température demande a le climatiseur. Remarque le climatiseur doit être allumer pour changer la température.



#### 1.14.4 GRAPHIQUE

Accéder aux graphes à l'Accueil ou à l'interface de télécommande, vers le bas gauche au pied de page noir appuyer sur l'icône  pour avoir le graphe de température et de l'humidité.



#### 1.15 CONCLUSION

L'application MACC répond au cahier des charges initial est réaliser une IHM pour appareil mobile qui permet de rejouer les trames IR pour marche/arrêt, up/down d'un climatiseur.

MACC répond aussi les changements durant la période du projet. L'application peut être améliorée au niveau de l'interface d'Accueil où l'utilisateur peut voir chaque climatiseur dans la salle connectée au lieu d'avoir une liste déroulante. MACC peut aussi améliorer le type de message envoyé vers l'ESP salle (d'après MACC Tech / Communication entre smartphone et l'Arduino).

## 2 MACC TECH

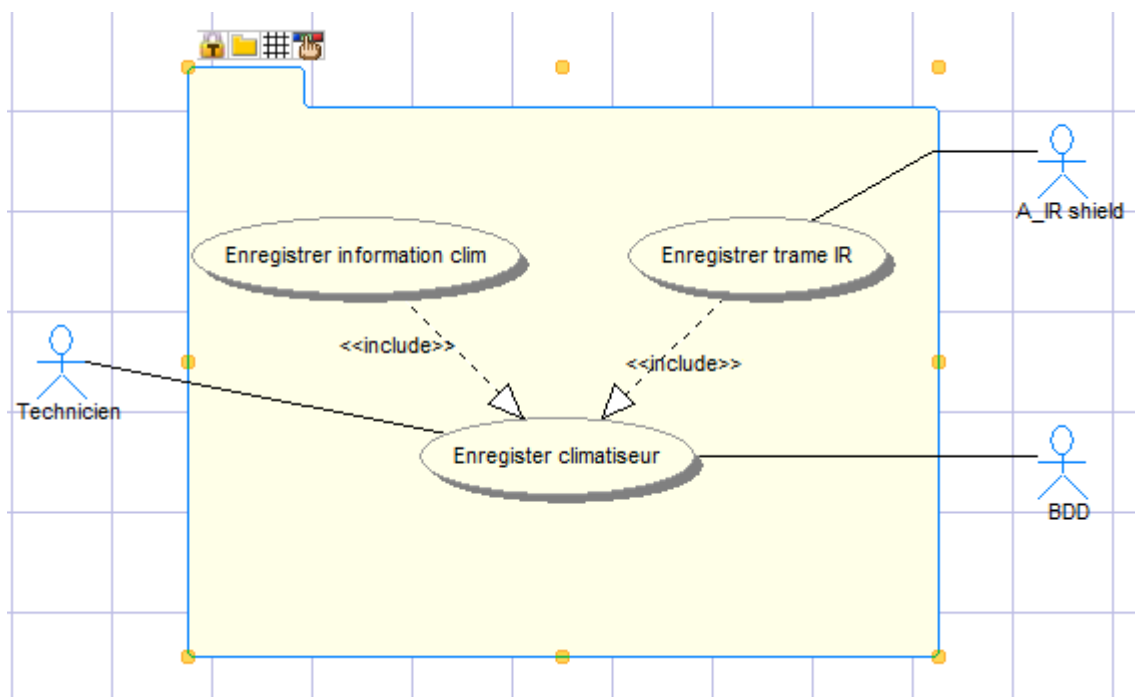
---

MACC Tech est une application pour les techniciens utilisés durant l'installation d'un climatiseur. Avec cette application permet en situation réelle de mieux faire l'apprentissage des trames. Cette partie je suis en collaboration avec étudiant 1 pour l'apprentissage des trames IR.

### 2.1 DIAGRAMME DE CAS D'UTILISATION

---

Voici une vision globale du diagramme de cas d'utilisation MACC Tech.



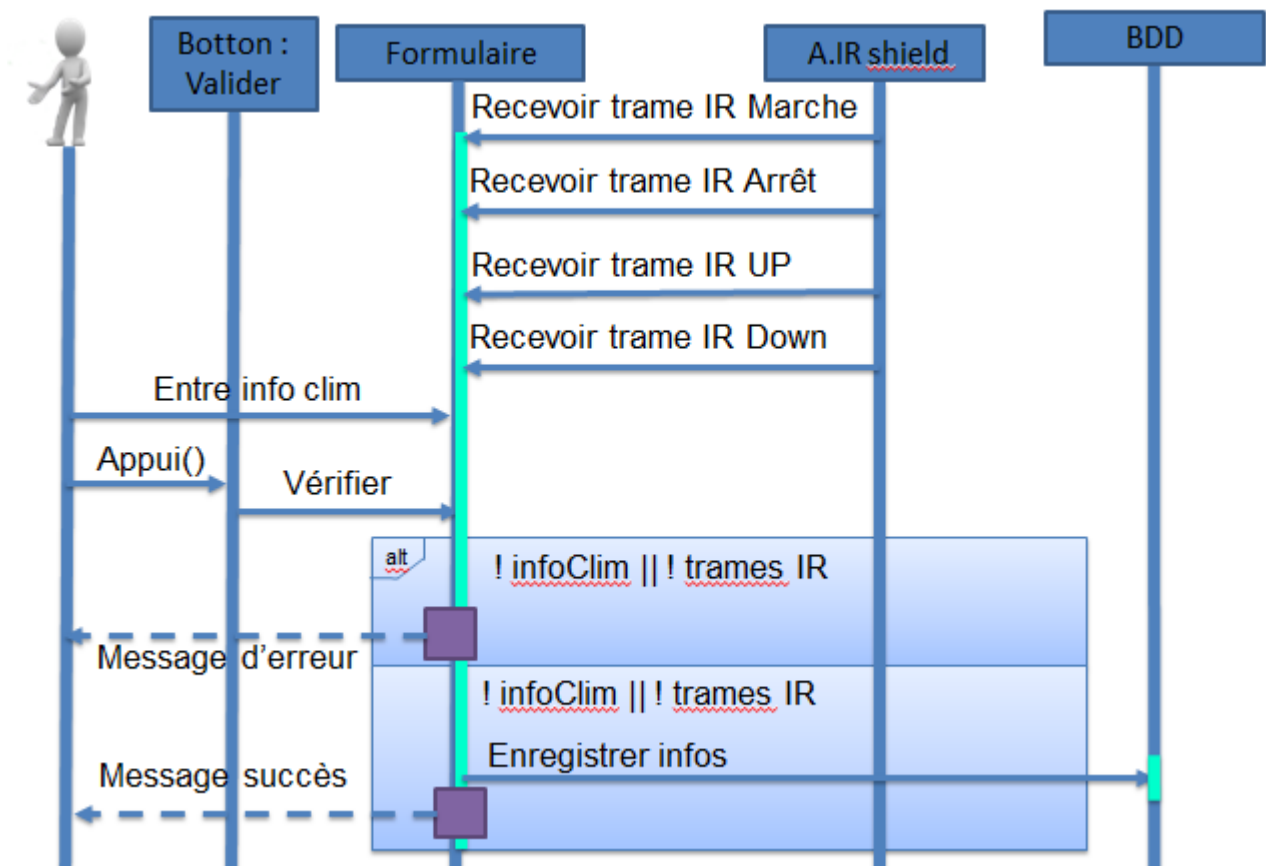
### 2.2 DIAGRAMME DE SEQUENCE

---

Ce diagramme de Séquence illustre le fonctionnement principal enregistrer un climatiseur dans la base de donnée avec l'application MACC Tech.

\*InfoClim représente toutes les informations de la climatiseur (Nom, Marque, localisation).

\*trames IR signifie les trames IR (Marche, Arrêt, Up, Down) de la télécommande du climatiseur.



## 2.3 MATERIEL

Un câble USB OTG (On The Go) est une extension de la norme USB 2.0 qui permet aux périphériques USB d'avoir davantage de flexibilité dans la gestion des connexions USB. En effet, grâce à l'OTG, deux périphériques peuvent échanger des données directement, sans avoir besoin de passer par un ordinateur hôte.

La possibilité d'envoyer des données sans passer par un contrôleur USB maître (typiquement, un ordinateur de bureau) ouvre un vaste champ d'application, par exemple :

- envoyer directement les photos d'un appareil photo à une imprimante ou un disque dur ;
- connecter directement un caméscope à un graveur de DVD ;
- échanger la musique d'un smartphone avec un lecteur MP3.





Les périphériques compatibles OTG disposent d'un connecteur de type mini-AB (ou, pour les plus récents, micro-AB), c'est-à-dire pouvant accepter indifféremment une fiche A (maître) ou B (esclave). Il n'est pas obligatoire que les deux appareils soient compatibles OTG pour communiquer, il suffit que l'un d'eux possède cette capacité pour établir la connexion point à point. Si l'autre périphérique ne prend pas l'OTG en charge, l'appareil OTG sera alors le maître de la communication.

## 2.4 COMMUNICATION ENTRE SMARTPHONE ET L'ARDUINO

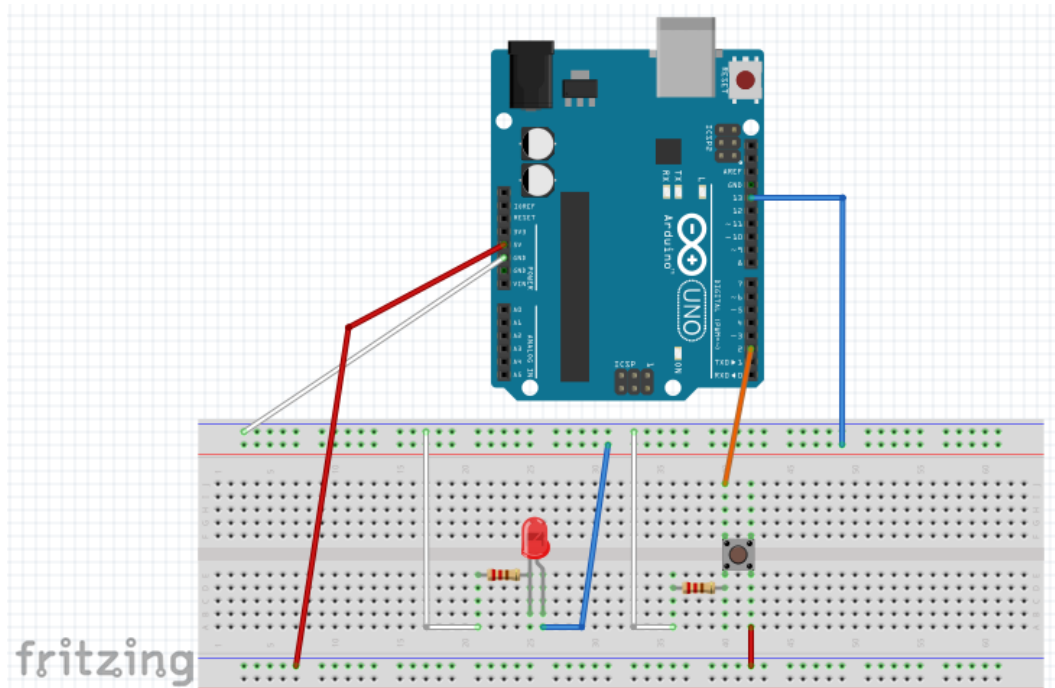
---

Pour que l'application MACC Tech puisse recevoir une trame IR d'un télécommande de climatiseur, je teste la communication entre mon arduino Uno avec l'android qui a une application teste pour recevoir un message.

### 2.4.1 ARDUINO

---

Sur l'arduino Uno va représenter un A.IR Shield Nano, le schéma ci-dessous est le circuit pour faire mes testes avec une LED rouge et un bouton poussoir. Le bouton poussoir représente un bouton X sur une télécommande X d'un climatiseur. Chaque fois le bouton est appuyé elle représente l'action d'une trame enregistre dans une variable, puis le contenu de cette variable est envoyé a l'android. La LED rouge indique que l'action s'est exécuté.



Grâce avec le logiciel "Fritzing" me permet de réaliser des maquettes pour circuit d'arduino.

Ensuite le code v1:

Les constantes ne changeront pas. Ils sont utilisés ici pour définir les numéros de broche.

Initialiser la branche du bouton poussoir sur 2 et la LED sur 13.

```
// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin = 13;     // the number of the LED pin
```

Une variable qui va stocker l'état de la valeur du bouton-poussoir :

```
// variables will change:
int buttonState = 0;        // variable for reading the pushbutton status
```

Le void setup() execute un fois, ici permet etablir la vitesse 9600 baud, initialiser la broche LED en sortie et initialiser la broche bouton poussoir en entre.

```
void setup() {
  //initialize the serial speed
  Serial.begin(9600);
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}
```

Ici le void loop() où exécute dans une boucle la partie du code qui envoie le message et allume la LED rouge :

```
void loop() {
```

Lire la valeur de l'état du bouton :

```
// read the state of the pushbutton value:  
buttonState = digitalRead(buttonPin);
```

Si le bouton est appuyé envoie le message puis allume la LED:

```
// check if the pushbutton is pressed. If it is, the buttonState is HIGH:  
if (buttonState == HIGH) {  
  // turn LED on:  
  digitalWrite(ledPin, HIGH);  
  Serial.peek();  
  Serial.print("ABCD"); //j'envoie ABCD  
  delay(5000);  
}
```

Sinon la LED est éteinte et arduino envoie rien :

```
} else {  
  // turn LED off:  
  digitalWrite(ledPin, LOW);  
}  
}
```

Ensuite le code v2:

le message à envoyer à l'écran android :

Ce message est un exemple de trame moitié décoder d'une télécommande de télévision trouvée sur internet :

```
String message = "Decoded NEC(1): Value:FD807F (32 bits) Raw samples(68):  
Gap:40826 Head: m8850 s4450 0:m500 s600 1:m550 s550 2:m500 s600 3:m550  
s600 4:m500 s600 5:m500 s600 6:m500 s600 7:m550 s550 8:m500 s1750 9:m500  
s1700 10:m500 s1700 11:m550 s1650 12:m550 s1700 13:m500 s1700 14:m500  
s600 15:m550 s1700 16:m500 s1700 17:m500 s600 18:m500 s600 19:m500 s600  
20:m550 s600 21:m450 s650 22:m500 s600 23:m500 s600 arduino is ending";
```

Initialiser la branche du bouton poussoir sur 2 et la LED sur 13 :

```
const int buttonPin = 2; // the number of the pushbutton pin  
const int ledPin = 13; // the number of the LED pin
```

Une variable qui va stocker l'état de la valeur du bouton-poussoir :

```
// variables will change:  
int buttonState = 0;
```

Le void setup() execute un fois, ici permet etablir la vitesse 9600 baud, initialiser la broche LED en sortie et initialiser la broche bouton poussoir en entre.

```
void setup() {  
  Serial.begin(9600);  
  // initialize the LED pin as an output:  
  pinMode(ledPin, OUTPUT);  
  // initialize the pushbutton pin as an input:  
  pinMode(buttonPin, INPUT);  
}
```

Ici le void loop() où exécute dans une boucle la partie du code qui envoie le message et allumer la LED rouge :

```
void loop() {
```

lire la valeur de l'état du bouton :

```
// read the state of the pushbutton value:  
buttonState = digitalRead(buttonPin);
```

Si le bouton est appuyé envoie le message caractère par caractère jusqu'à la fin du message puis allume la LED:

```
// check if the pushbutton is pressed. If it is, send message and the buttonState is HIGH.
```

```
if (buttonState == HIGH) {  
  //send the message character by character with a fast delay about 50  
  for(int index = -1; index != message.length(); index++){  
    Serial.print(message[index]);  
    delay(10);  
  }  
}
```

```
// turn LED on:  
digitalWrite(ledPin, HIGH);  
delay(1000);
```

Sinon la LED est éteint et arduino envoie rien :

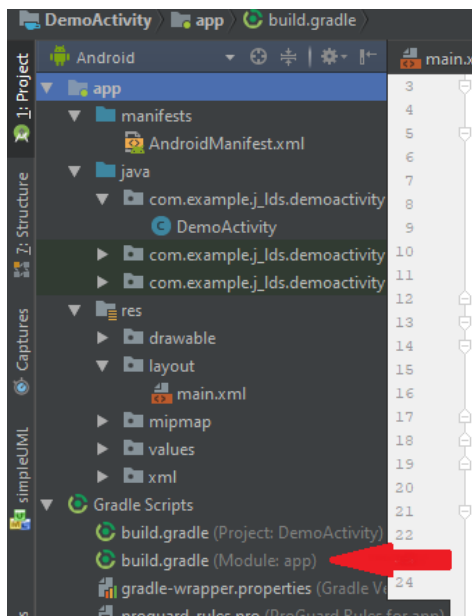
```
} else {  
  //send nothing  
  // turn LED off:  
  digitalWrite(ledPin, LOW);  
}
```

```
}  
}
```

## 2.4.2 ANDROID

Avec Android Studio je prends une partie d'un code d'un projet sur GitHub avec la librairie **usb-serial-for-android-v010.jar** qui vient avec. La partie prise reçoit un message et affiche ce message en byte depuis l'arduino.

La logique de cette application teste qui s'appelle DemoActivity est quand l'application reçoit un message sur l'android que j'affiche. Avant de coder sur l'application il faut implémenter la librairie **usb-serial-for-android-v010.jar** dans "Module: app"



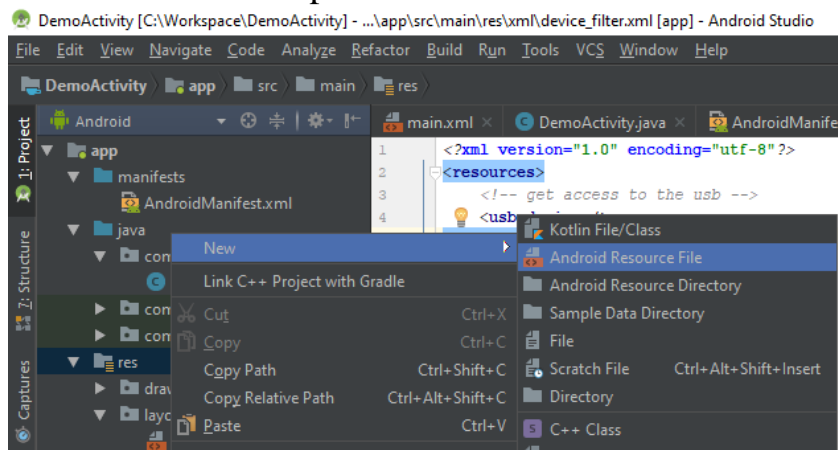
Ajouter la ligne suivante : `implementation files('libs/usb-serial-for-android-v010.jar')` qui va me donner accès au contenu du fichier.

```
androidTestImplementation 'com.android.support.test.runner:1.0.2'  
androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.1.0'  
implementation files('libs/usb-serial-for-android-v010.jar')  
}
```

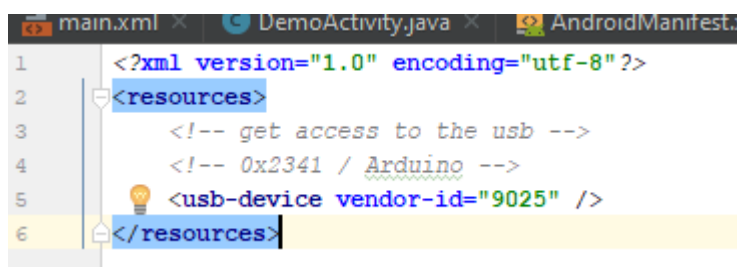
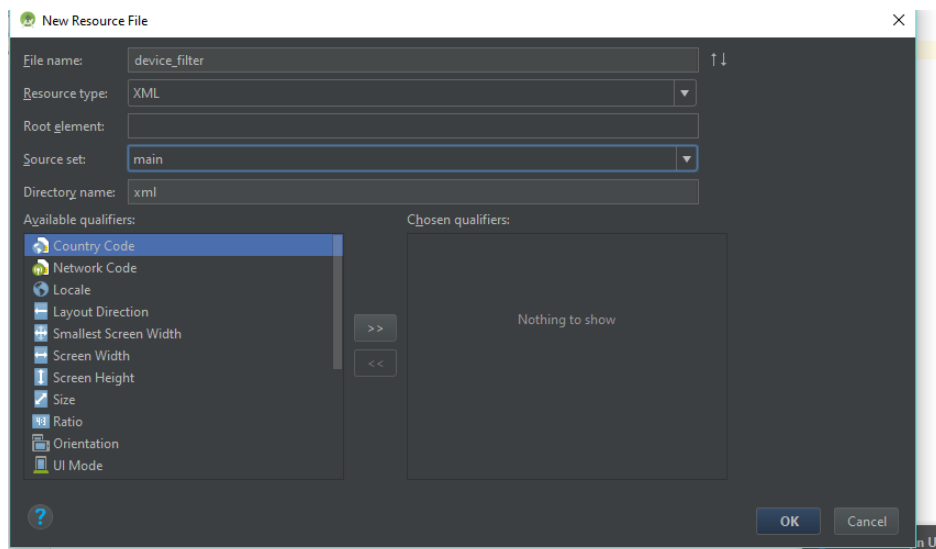
Les dépendances d'implémentation sont utilisées pour que vous ayez vraiment besoin d'accéder à toutes les classes d'un autre module, dans ce cas oui j'ai besoin de toutes les classes du fichier.

Puis je crée un fichier source d'android. Les périphériques esclaves USB ont un identifiant de fournisseur et de produit, utilisé pour identifier les pilotes qui

doivent être utilisés pour cela.



Ce fichier device\_filter.xml contient l'identifiant du fournisseur Arduino est toujours 0x2341 ou 9025.



Ensuite dans la fichier Manifest, il faut dire a l'application d'activer le mode hôte USB sur votre Android.

**<uses-feature android:name="android.hardware.usb.host" />**

Pour que l'application découvre un périphérique USB particulier (mon Arduino

Uno), je spécifie un filtre d'intention à filtrer pour l'intention **android.hardware.usb.action.USB\_DEVICE\_ATTACHED**. Lorsque le technicien connecte A.IR Shield Nano correspondant au filtre de votre périphérique, le système leur présente une boîte de dialogue leur demandant s'ils souhaitent démarrer votre application. Si les utilisateurs acceptent, votre application a automatiquement l'autorisation d'accéder à l'appareil jusqu'à ce que l'appareil soit déconnecté.

L'exemple suivant montre comment déclarer le filtre d'intention:

```
<intent-filter>
    <action
        android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />
</intent-filter>
<meta-data
    android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
    android:resource="@xml/device_filter" />
```

Enfin dans DemoActivity.java j' intègre le code copier et la ma modification.

```
public class DemoActivity extends AppCompatActivity {
```

Ici je déclare mes objets:

TAG contient le nom de la classe(c'est DemoActivity).

```
private final String TAG = DemoActivity.class.getSimpleName();
private UsbSerialDriver mSerialDevice;
private UsbManager mUsbManager;

private TextView mTitleTextView;
private TextView mDumpTextView;
private ScrollView mScrollView;

//variable ascii1 contiendra un caractère récupéré depuis l'arduino

private String ascii1;

//arduinoFullMessage contiendra le message total envoye

private String arduinoFullMessage = "";

private final ExecutorService mExecutor = Executors.newSingleThreadExecutor();

private SerialInputOutputManager mSerialIoManager;

//obtenir l'entrée série de Arduino

private final SerialInputOutputManager.Listener mListener =
    new SerialInputOutputManager.Listener() {

        @Override
        public void onRunError(Exception e) {
            Log.d(TAG, "Runner stopped.");
        }

        //un thread qui écoute un message de arduino qui vient en byte
        array //passe le message byte
        array a updateReceivedData et démarrez la méthode

        @Override
        public void onNewData(final byte[] data) {
            DemoActivity.this.runOnUiThread(new Runnable() {
```

```

        @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
        @Override
        public void run() {
            DemoActivity.this.updateReceivedData(data);
        }
    });
};

//onCreate est utilisé pour démarrer une activité
//définir la mise en page

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    //mUsbDevice représente le périphérique attaché

    mUsbManager = (UsbManager) getSystemService(Context.USB_SERVICE);

    //le reste les objets représente dans la mise en page xml

    mTitleTextView = (TextView) findViewById(R.id.demoTitle);
    mDumpTextView = (TextView) findViewById(R.id.demoText);
    mScrollView = (ScrollView) findViewById(R.id.demoScroller);
}

//si le périphérique est détaché de l'android, fermer la connexion

@Override
protected void onPause() {
    super.onPause();
    stopIoManager();
    if (mSerialDevice != null) {
        try {
            mSerialDevice.close();
        } catch (IOException e) {
            // Ignore.
        }
        mSerialDevice = null;
    }
}

//revenir à l'activité puis essayi de commencer serial inputOut Manager

@RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
@Override
protected void onResume() {
    super.onResume();

    //avoir le serial du périphérique

    mSerialDevice = UsbSerialProber.acquire(mUsbManager);
    Log.d(TAG, "Resumed, mSerialDevice=" + mSerialDevice);

    //si le périphérique n'est pas connecté

    if (mSerialDevice == null) {
        mTitleTextView.setText("No serial device.");
    } else {

        //si le périphérique est connecté
        //essai d'ouvrir la connexion
        try {
            mSerialDevice.open();
        } catch (IOException e) {

```



```

        //si la connexion échoue, informez l'utilisateur

        Log.e(TAG, "Error setting up device: " + e.getMessage(), e);
        mTitleTextView.setText("Error opening device: " + e.getMessage());
        try {
            //ferme la connexion

            mSerialDevice.close();
        } catch (IOException e2) {
            // Ignore.
        }
        //initiale serial null

        mSerialDevice = null;
        return;
    }
    //serial est ouvert et connecté

    mTitleTextView.setText("Serial device: " + mSerialDevice);
}
//commencer serial inputOutput Manager

onDeviceStateChange();
}

//arreter serial inputOutput Manager

private void stopIoManager() {
    if (mSerialIoManager != null) {
        Log.i(TAG, "Stopping io manager ..");
        mSerialIoManager.stop();
        mSerialIoManager = null;
    }
}

//commencer serial inputOutput Manager

private void startIoManager() {
    if (mSerialDevice != null) {
        Log.i(TAG, "Starting io manager ..");
        mSerialIoManager = new SerialInputOutputManager(mSerialDevice,
mListener);
        mExecutor.submit(mSerialIoManager);
    }
}

private void onDeviceStateChange() {
    stopIoManager();
    startIoManager();
}

//Réception d'un message de l'arduino byte array
//Ici j'ai modifier cette méthode

@RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
private void updateReceivedData(byte[] data) {
    try {
        //Renvoie vrai si le peripherique est autorise d'accéder à l'android

        mUsbManager.hasPermission(mSerialDevice.getDevice());

        //Définir la vitesse série 9600 bauds bps (bits par seconde)

        mSerialDevice.setBaudRate(9600);

        //obtenir le message

```

//synchronisation est utilisé qu'un seul thread peut accéder à la ressource à un moment donné.

```
synchronized (mSerialIoManager.getListener()) {  
  
    //data est un caractère reçu en byte qui est converti ascii grâce a  
    UTF-8  
    //effectue dans ascii1 le caractère depuis l'arduino  
  
    ascii1 = new String(data, "UTF-8");  
  
    //j'ajoute le caractère ascii dans une variable A LA SUITE  
    arduinoFullMessage += ascii1;  
  
    //je verifie si le message total se termine avec "adrduino is  
    ending"  
    if (arduinoFullMessage.endsWith("adrduino is ending")){  
  
        //si oui je retire  
        "adrduino is ending" puis j'ai le message de l'arduino  
        //"adrduino is ending" me permet de savoir que l'arduino à  
        terminé de transmettre son message  
  
        arduinoFullMessage = arduinoFullMessage.replaceAll("adrduino is  
        ending", "");  
        String msg = "Arduino Data length :" + ascii1.length() + "  
        \nArduino Data : '" + arduinoFullMessage + "'\n\n";  
  
        //j'affiche le message reçu sur l'écran  
  
        mDumpTextView.append(msg);  
        mScrollView.smoothScrollTo(0, mDumpTextView.getBottom());  
  
        //initialise où contient le message total de l'arduino a rien  
        pour le message suivant  
        arduinoFullMessage = "";  
    }  
}  
} catch (Exception e) {  
    e.printStackTrace();  
}  
}
```

Voici la méthode initial avant la modification

```
private void updateReceivedData(byte[] data) {  
    final String message = "Read " + data.length + " bytes: \n"  
        + HexDump.dumpHexString(data) + "\n\n";  
    mDumpTextView.append(message);  
    mScrollView.smoothScrollTo(0, mDumpTextView.getBottom());  
}
```

## 2.4.3 RESULTA DU TESTE

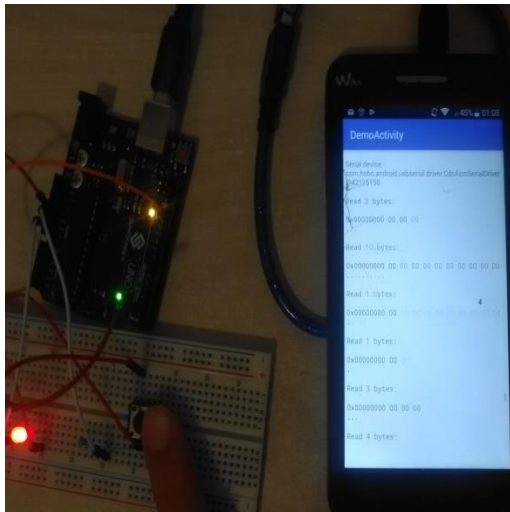
---

Voici les différents essais pour ce test :

Essai 1 avec le premier code de l'arduino (v1) et le code d'android sans modifier la méthode updateReceivedData :

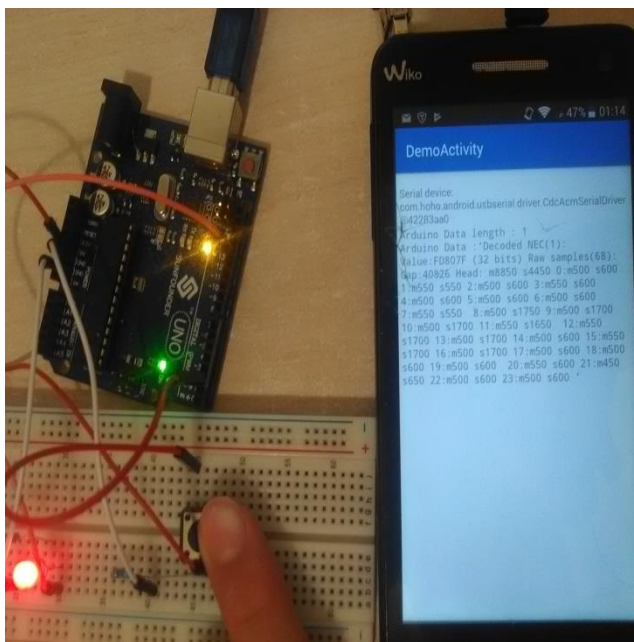
Connecte l'arduino à l'android avec l'usb OTG, une fois connecté l'arduino demande l'autorisation de connexion l'arduino :





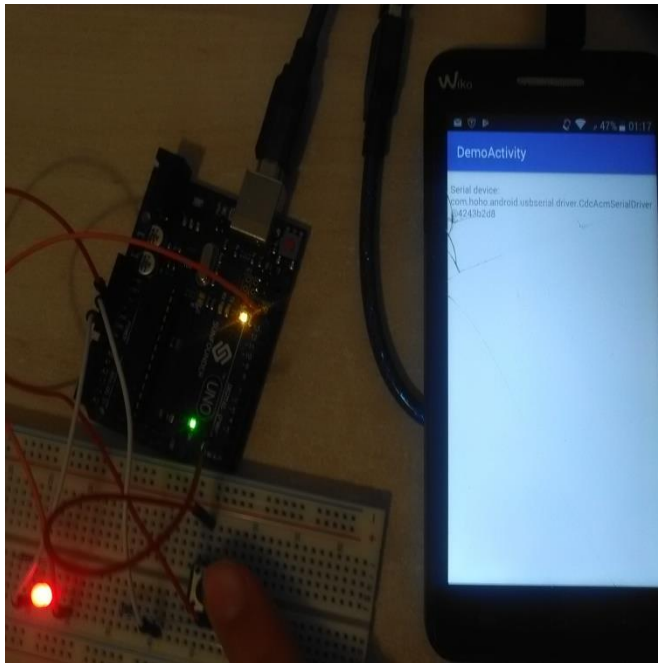
Essai 3 avec le deuxième code de l'arduino (v2) et le code d'android, la méthode `updateReceivedData` modifier :

Appuie le bouton poussoir l'arduino envoie le message vers l'android, l'android reçoit le message et affiche en ascii:



Essai 4 avec le premier code de l'arduino (v1) et le code d'android, la méthode `updateReceivedData` modifier :

Appuie le bouton poussoir l'arduino envoie le message vers l'android, l'android reçoit aucun message ou n'affiche aucun message:



En conclusion l'essai 3 marche avec succès, l'android reçoit et affiche le message.

## 2.5 IHM

### 2.5.1 FORMULAIRE

L'image ci-dessous re présente une interface où le technicien récupère les trames IR avec la télécommande du climatiseur.

MACC Tech

Formulaire de Climatiseur

Nom

Trame IR

☐ On ☐ Off

Temperature

<input type="checkbox"/> 16	<input type="checkbox"/> 17	<input type="checkbox"/> 18	<input type="checkbox"/> 19	<input type="checkbox"/> 20
<input type="checkbox"/> 21	<input type="checkbox"/> 22	<input checked="" type="checkbox"/> 23	<input type="checkbox"/> 24	<input type="checkbox"/> 25
<input type="checkbox"/> 26	<input type="checkbox"/> 27	<input type="checkbox"/> 28	<input type="checkbox"/> 29	<input type="checkbox"/> 30

RÉINITIALISER ENREGISTRER

## 2.6 CONCLUSION

MACC Tech est une amélioration de situation réelle qui est propose durant le projet. Le test entre arduino et l'android marche avec la logique du fonctionnement mais reste avoir entre A.IR Shield Nano et l'arduino. J'estime que MACC Tech est moitié terminé.

### 3 CONCLUSION GENERALE

### 4 PLANIFICATION

		Planification	Nom de tâche	Durée	Début	Fin	Prédécesse	Progressi	Priorité	Ressources	Travail	Coût
1			Étudiant 4	101 jour	15/01/2018	04/06/2018		98%	500		0 h	€0.00
2			MACC	91 jours	15/01/2018	21/05/2018		100%	500		0 h	€0.00
3			Diagramme de Cas d'Utilisation	7 jours	15/01/2018	23/01/2018		100%	500		0 h	€0.00
4			Environnement de développement p...	7 jours	15/01/2018	23/01/2018		100%	500		0 h	€0.00
5			Hello world	4 jours	24/01/2018	29/01/2018	4	100%	500		0 h	€0.00
6			IHM	5 jours	30/01/2018	05/02/2018		100%	500		0 h	€0.00
7			Interface d'identification	5 jours	30/01/2018	05/02/2018	5	100%	500		0 h	€0.00
8			Interface d'accueil	5 jours	30/01/2018	05/02/2018	5	100%	500		0 h	€0.00
9			Interface de télécommande	5 jours	30/01/2018	05/02/2018	5	100%	500		0 h	€0.00
10			Interface de représentation graphi	5 jours	30/01/2018	05/02/2018	5	100%	500		0 h	€0.00
11			Diagramme de séquence	5 jours	30/01/2018	05/02/2018	5	100%	500		0 h	€0.00
12			Connexion à la base de donnée	19 jours	06/02/2018	02/03/2018		98%	500		0 h	€0.00
13			Vérifier l'identité	10 jours	06/02/2018	19/02/2018	7	100%	500		0 h	€0.00
14			Affichage du graphe	10 jours	19/02/2018	02/03/2018	10	95%	500		0 h	€0.00
15			Communication avec ESP32	17 jours	02/03/2018	26/03/2018		100%	500		0 h	€0.00
16			Envoyer une commande	10 jours	02/03/2018	15/03/2018		100%	500		0 h	€0.00
17			Récupérer le nombre de climatiseur	7 jours	16/03/2018	26/03/2018	16	100%	500		0 h	€0.00
18			Afficher les Climatiseurs	7 jours	27/03/2018	04/04/2018	17	100%	500		0 h	€0.00
19			Manifeste	7 jours	30/03/2018	09/04/2018		100%	500		0 h	€0.00
20			Passage entre l'interface Accueil et ...	10 jours	10/04/2018	23/04/2018		100%	500		0 h	€0.00
21			Déconnexions	3 jours	24/04/2018	26/04/2018		100%	500		0 h	€0.00
22			Diagramme de Classe	5 jours	27/04/2018	03/05/2018	18	100%	500		0 h	€0.00
23			Document d'installation	4 jours	04/05/2018	09/05/2018		100%	500		0 h	€0.00
24			Générer APK	2 jours	04/05/2018	07/05/2018		100%	500		0 h	€0.00
25			Installer sur Android	2 jours	08/05/2018	09/05/2018	24	100%	500		0 h	€0.00
26			Document d'utilisation	8 jours	10/05/2018	21/05/2018		100%	500		0 h	€0.00
27			S'identifier	2 jours	10/05/2018	11/05/2018	25	100%	500		0 h	€0.00
28			Sélectionner un/plusieurs climatise...	2 jours	14/05/2018	15/05/2018	27	100%	500		0 h	€0.00
29			Effectuer une commande	2 jours	16/05/2018	17/05/2018	28	100%	500		0 h	€0.00
30			Graphique	2 jours	18/05/2018	21/05/2018	29	100%	500		0 h	€0.00
31			Conclusion	2 jours	08/05/2018	09/05/2018		100%	500		0 h	€0.00
32			MACC Tech	18 jours?	10/05/2018	04/06/2018		93%	500		0 h	€0.00
33			Diagramme de Cas d'Utilisation	3 jours	10/05/2018	14/05/2018		100%	500		0 h	€0.00
34			Diagramme de séquence	3 jours	15/05/2018	17/05/2018	33	100%	500		0 h	€0.00
35			Matériel	7 jours	11/05/2018	21/05/2018		100%	500		0 h	€0.00
36			Communication entre smartpho...	10 jours	22/05/2018	04/06/2018		95%	500		0 h	€0.00
37			Arduino	7 jours	22/05/2018	30/05/2018	35	100%	500		0 h	€0.00
38			Android	10 jours	22/05/2018	04/06/2018	35	90%	500		0 h	€0.00
39			Résulta du Teste	5 jours	24/05/2018	30/05/2018		100%	500		0 h	€0.00
40			IHM	2 jours	30/05/2018	31/05/2018		50%	500		0 h	€0.00
41			Formulaire	2 jours	30/05/2018	31/05/2018	34	50%	500		0 h	€0.00
42			Conclusion	1 jour?	31/05/2018	31/05/2018		50%	500		0 h	€0.00
43			Conclusion Générale	1 jour?	31/05/2018	31/05/2018		100%	500		0 h	€0.00
44			Planification	2 jours	30/05/2018	31/05/2018		100%	500		0 h	€0.00



