Meeting

version 0.2

BTS SNIR LaSalle Avignon 2020

Table des matières

1 Le projet

Placé à l'extérieur d'une pièce (salle de réunion ou de travail, ...), le système **Meeting** permettra d'accéder en temps réel aux informations de l'espace concerné : il affichera la disponibilité et l'état de confort et permettra de réaliser sa réservation.

1.1 Table des matières

1.2 Informations

Auteur

Vincent Devine vincentdevine84@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/meeting/

2 Changelog

r1 | www-data | 2020-02-01 15 :03 :29 +0100 (sam. 01 févr. 2020) | 1 ligne

Creating initial repository structure

3 README

3.1 Projet

3.1.1 Présentation

Placé à l'extérieur d'une pièce (salle de réunion ou de travail, ...), le système **Meeting** permettra d'accéder en temps réel aux informations de l'espace concerné : il affichera la disponibilité et l'état de confort et permettra de réaliser sa réservation.

L'objectif est de proposer une solution simple, alliant flexibilité et ergonomie. Le système affiche de la disponibilité d'accès et le niveau de confort d'une salle de travail ou de réunion.

Le portier connecté est composé :

— d'un micro-contrôleur (ESP32, Z-duino,...)
— d'un écran tactile
— d'indicateurs lumineux (Leds)
— d'une liaison Bluetooth vers une sonde permettant d'évaluer un “indice de confort”

La sonde est composée :

— d'un capteur de température
— d'un capteur d'hygrométrie
— d'un capteur de qualité d'air

A partir d'une application mobile sous Android et communiquant en UPD via le WiFi, l'utilisateur pourra :

— Lorsqu'une salle a été sélectionnée, visualiser les informations sur la salle (son nom et des informations complémentaires qu'il pourra associer à celle-ci comme sa localisation, sa surface ...), sa disponibilité et les mesures en provenance du module sonde ainsi que son indice de confort.
— configurer les informations d'une salle.
— la possibilité de prendre une salle en indiquant une durée estimée d'occupation de celle-ci. Dans ce cas, le portier lui enverra un code qu'il utilisera pour augmenter la durée d'occupation ou de libérer la salle.

3.1.2 Informations

Auteur

Vincent Devine vincentdevine84@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/meeting/

4 A propos

Auteur

Vincent Devine vincentdevine84@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/meeting/

5 Licence GPL

6 Documentation des espaces de nommage

6.1 Paquetage com

Paquetages

— package lasalle

6.2 Paquetage com.lasalle

Paquetages

— package meeting

6.3 Paquetage com.lasalle.meeting

Classes

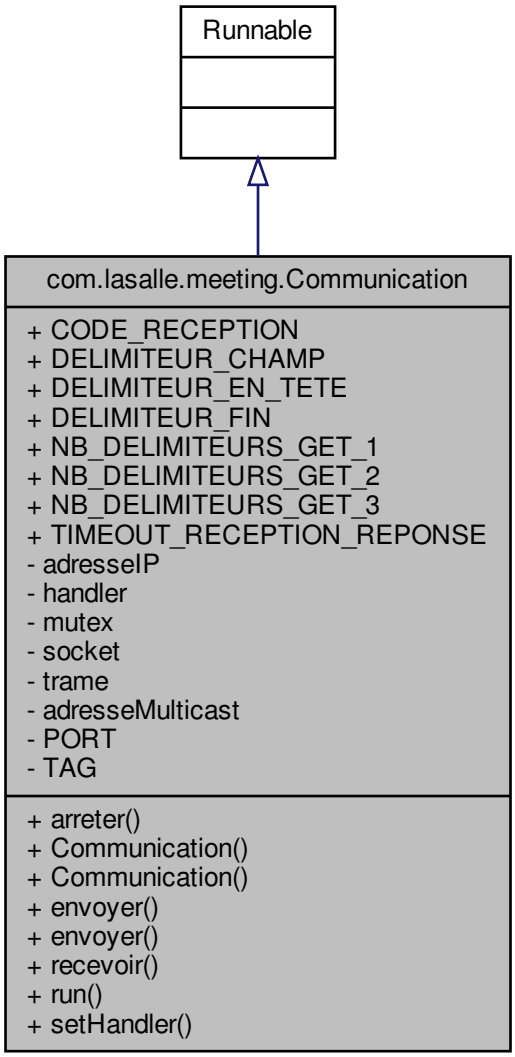
— class Communication
Déclaration de la classe Communication.
— class ConfigurationSalleActivity
Déclaration de la classe ConfigurationSalleActivity.
— class MainActivity
Déclaration de la classe MainActivity.
— class RechercherNomBoiteDialogue
— class Salle
Déclaration de la classe Salle.
— class SalleActivity
Déclaration de la classe SalleActivity.
— class SalleAdapter
Déclaration de la classe SalleAdapter.
— class SalleViewHolder
Déclaration de la classe SalleViewHolder.

7 Documentation des classes

7.1 Référence de la classe com.lasalle.meeting.Communication

Déclaration de la classe Communication.

Graphe de collaboration de com.lasalle.meeting.Communication :



Fonctions membres publiques

- void arreter ()
  méthode arrétant la socket, donc la communication avec les portiers
- Communication (Handler handler)
  constructeur de communication
- Communication (int port, Handler handler)
  constructeur de communication
- void envoyer (final String requete)

méthode envoyant une requête à l'adresse de multicast
- void envoyer (final String requete, final String adresseIP)
  méthode envoyant une requête à l'adresse de indiqué en paramètre
- void recevoir ()
  méthode recevant les trames des portiers
- void run ()
  méthode appelée automatiquement quand le socket reçois quelque chose
- void setHandler (Handler handler)
  change le handler par celui mis en paramètre

Attributs publics statiques

- static final int CODE_RECEPTION = 1
  code de reception correcte pour le portiers
- static final String DELIMITEUR_CHAMP = ";"
- static final String DELIMITEUR_EN_TETE = "$"
- static final String DELIMITEUR_FIN = "\r\n"
- static final int NB_DELIMITEURS_GET_1 = 6
- static final int NB_DELIMITEURS_GET_2 = 3
- static final int NB_DELIMITEURS_GET_3 = 1
- static final int TIMEOUT_RECEPTION_REPONSE = 30000
  temps maximum d'une réponse d'un portier

Attributs privés

- InetAddress adresseIP = null
  attribut récuperant l'adresse IP du portier
- Handler handler
  attribut permetant d'envoyer une requête par rapport a une autre activity
- final ReentrantLock mutex = new ReentrantLock()
- DatagramSocket socket
  attribut récuperant les informations de la socket
- String trame
  attribut récuperant la trame

Attributs privés statiques

- static final String adresseMulticast = "239.0.0.42"
  adresse de multicast des portiers
- static final int PORT = 5000
  port d'ecoute des portiers
- static final String TAG = "Communication"
  TAG utilisé dans les log.

7.1.1 Description détaillée

Déclaration de la classe Communication.

Définition à la ligne 23 du fichier Communication.java.

7.1.2 Documentation des constructeurs et destructeur

7.1.2.1 Communication() [1/2]

```
com.lasalle.meeting.Communication.Communication (
            Handler handler )
```

constructeur de communication

Paramètres

handler	Handler

Renvoie

void

Définition à la ligne 56 du fichier Communication.java.

Références com.lasalle.meeting.Communication.handler, et com.lasalle.meeting.Communication.TIMEOUT_RECEPTION_REPONSE.

```
00057     {
00058         this.handler = handler;
00059         try
00060         {
00061             socket = new DatagramSocket(PORT);
00062             socket.setSoTimeout(Communication.TIMEOUT_RECEPTION_REPONSE);
00063         }
00064         catch (SocketException se)
00065         {
00066             se.printStackTrace();
00067         }
00068 
00069         try
00070         {
00071             this.adresseIP = InetAddress.getByName(adresseMulticast);
00072         }
00073         catch (UnknownHostException e)
00074         {
00075             e.printStackTrace();
00076         }
00077     }
```

7.1.2.2 Communication() [2/2]

```
com.lasalle.meeting.Communication.Communication (
            int port,
            Handler handler )
```

constructeur de communication

Paramètres

handler	Handler, port int

Renvoie

void

Définition à la ligne 84 du fichier Communication.java.

Références com.lasalle.meeting.Communication.handler, et com.lasalle.meeting.Communication.TIMEOUT_RECEPTION_REPONSE.

```
00085     {
00086         this.handler = handler;
00087         try
00088         {
00089             socket = new DatagramSocket(port);
```

```
00090             socket.setSoTimeout(Communication.TIMEOUT_RECEPTION_REPONSE);
00091         }
00092         catch (SocketException se)
00093         {
00094             se.printStackTrace();
00095         }
00096 
00097         try
00098         {
00099             this.adresseIP = InetAddress.getByName(adresseMulticast);
00100         }
00101         catch (UnknownHostException e)
00102         {
00103             e.printStackTrace();
00104         }
00105     }
```

7.1.3 Documentation des fonctions membres

7.1.3.1 arreter()

```
void com.lasalle.meeting.Communication.arreter ( )
```

méthode arrétant la socket, donc la communication avec les portiers

Renvoie

void

Définition à la ligne 235 du fichier Communication.java.

```
00236     {
00237         if(socket == null)
00238             return;
00239         socket.close();
00240     }
```

7.1.3.2 envoyer() [1/2]

```
void com.lasalle.meeting.Communication.envoyer (
            final String requete )
```

méthode envoyant une requête à l'adresse de multicast

Paramètres

requete	String

Renvoie

void

Définition à la ligne 161 du fichier Communication.java.

Références com.lasalle.meeting.Communication.run().

Référencé par com.lasalle.meeting.MainActivity.rafraichir(), com.lasalle.meeting.SalleActivity.setListener(), et com.lasalle.meeting.↩ ConfigurationSalleActivity.setListener().

```
00162     {
00163         if(socket == null)
00164             return;
00165 
00166         new Thread()
00167         {
00168             @Override public void run()
00169             {
00170                 byte[] emission = new byte[1024];
00171 
00172                 try
00173                 {
00174                     emission = requete.getBytes();
00175                     DatagramPacket paquetRetour = new DatagramPacket(emission, emission.length,
      adresseIP, PORT);
00176                     socket.send(paquetRetour);
00177                     Log.d(TAG, "send() = " + requete);
00178                 }
00179                 catch (IOException e)
00180                 {
00181                     Log.d(TAG, "Erreur send() [socket.isClosed = " + socket.isClosed() + "]");
00182                     e.printStackTrace();
00183                 }
00184             }
00185         }.start();
00186     }
```

7.1.3.3 envoyer() [2/2]

```
void com.lasalle.meeting.Communication.envoyer (
             final String requete,
             final String adresseIP )
```

méthode envoyant une requête à l'adresse de indiqué en paramètre

Paramètres

requete	String, adresseIP String

Renvoie

void

Définition à la ligne 193 du fichier Communication.java.

Références com.lasalle.meeting.Communication.run().

```
00194     {
00195         if(socket == null)
00196             return;
00197 
00198         final InetAddress adresseIPDistante;
00199         try
00200         {
00201             adresseIPDistante = InetAddress.getByName(adresseIP);
00202         }
00203         catch (UnknownHostException e)
00204         {
00205             e.printStackTrace();
00206             return;
00207         }
00208 
00209         new Thread()
00210         {
00211             @Override public void run()
00212             {
00213                 byte[] emission = new byte[1024];
00214 
00215                 try
00216                 {
00217                     emission = requete.getBytes();
```

```
00218                     DatagramPacket paquetRetour = new DatagramPacket(emission, emission.length,
      adresseIPDistante, PORT);
00219                     socket.send(paquetRetour);
00220                     Log.d(TAG, "send() " + adresseIP + " = " + requete);
00221                 }
00222                 catch (IOException e)
00223                 {
00224                     Log.d(TAG, "Erreur send() [socket.isClosed = " + socket.isClosed() + "]");
00225                     e.printStackTrace();
00226                 }
00227             }
00228         }.start();
00229     }
```

7.1.3.4 recevoir()

```
void com.lasalle.meeting.Communication.recevoir ( )
```

méthode recevant les trames des portiers

Renvoie

void

Définition à la ligne 123 du fichier Communication.java.

Références com.lasalle.meeting.Communication.CODE_RECEPTION, et com.lasalle.meeting.Communication.trame.

Référencé par com.lasalle.meeting.Communication.run().

```
00124     {
00125         byte[] reception = new byte[1024];
00126 
00127         while (socket != null && !socket.isClosed())
00128         {
00129             try
00130             {
00131                 final DatagramPacket paquetRecu = new DatagramPacket(reception, reception.length);
00132                 socket.receive(paquetRecu);
00133 
00134                 trame = new String(paquetRecu.getData(), paquetRecu.getOffset(), paquetRecu.getLength(
      ));
00135                 Log.d(TAG, "Réception de " + paquetRecu.getAddress().getHostAddress() + ":" + paquetRecu
      .getPort() + " -> " + trame);
00136 
00137                 Message msg = Message.obtain();
00138                 Bundle b = new Bundle();
00139                 b.putString("adresseIP", paquetRecu.getAddress().getHostAddress());
00140                 b.putInt("port", paquetRecu.getPort());
00141                 b.putInt("etat", Communication.CODE_RECEPTION);
00142                 b.putString("trame", trame);
00143                 msg.setData(b);
00144                 mutex.lock();
00145                 handler.sendMessage(msg);
00146                 mutex.unlock();
00147             }
00148             catch (Exception e)
00149             {
00150                 Log.d(TAG, "Erreur recevoir() [socket.isClosed = " + socket.isClosed() + "]");
00151                 e.printStackTrace();
00152             }
00153         }
00154     }
```

7.1.3.5 run()

```
void com.lasalle.meeting.Communication.run ( )
```

méthode appelée automatiquement quand le socket reçois quelque chose

Renvoie

void

Définition à la ligne 247 du fichier Communication.java.

Références com.lasalle.meeting.Communication.recevoir().

Référencé par com.lasalle.meeting.Communication.envoyer().

```
00248     {
00249         recevoir();
00250     }
```

7.1.3.6 setHandler()

```
void com.lasalle.meeting.Communication.setHandler (
             Handler handler )
```

change le handler par celui mis en paramètre

Paramètres

handler	Handler

Renvoie

void

Définition à la ligne 112 du fichier Communication.java.

Références com.lasalle.meeting.Communication.handler.

```
00113     {
00114         mutex.lock();
00115         this.handler = handler;
00116         mutex.unlock();
00117     }
```

7.1.4 Documentation des données membres

7.1.4.1 adresseIP

```
InetAddress com.lasalle.meeting.Communication.adresseIP = null  [private]
```

attribut récuperant l'adresse IP du portier

Définition à la ligne 38 du fichier Communication.java.

7.1.4.2 adresseMulticast

```
final String com.lasalle.meeting.Communication.adresseMulticast = "239.0.0.42"  [static], [private]
```

adresse de multicast des portiers

Définition à la ligne 29 du fichier Communication.java.

7.1.4.3 CODE_RECEPTION

```
final int com.lasalle.meeting.Communication.CODE_RECEPTION = 1  [static]
```

code de reception correcte pour le portiers

Définition à la ligne 32 du fichier Communication.java.

Référencé par com.lasalle.meeting.Communication.recevoir().

7.1.4.4 DELIMITEUR_CHAMP

```
final String com.lasalle.meeting.Communication.DELIMITEUR_CHAMP = ";"  [static]
```

Définition à la ligne 45 du fichier Communication.java.

7.1.4.5 DELIMITEUR_EN_TETE

```
final String com.lasalle.meeting.Communication.DELIMITEUR_EN_TETE = "$"  [static]
```

Protocole

Définition à la ligne 44 du fichier Communication.java.

7.1.4.6 DELIMITEUR_FIN

```
final String com.lasalle.meeting.Communication.DELIMITEUR_FIN = "\r\n"  [static]
```

Définition à la ligne 46 du fichier Communication.java.

7.1.4.7 handler

```
Handler com.lasalle.meeting.Communication.handler  [private]
```

attribut permetant d'envoyer une requête par rapport a une autre activity

Définition à la ligne 39 du fichier Communication.java.

Référencé par com.lasalle.meeting.Communication.Communication(), et com.lasalle.meeting.Communication.setHandler().

7.1.4.8 mutex

```
final ReentrantLock com.lasalle.meeting.Communication.mutex = new ReentrantLock()  [private]
```

Définition à la ligne 33 du fichier Communication.java.

7.1.4.9 NB_DELIMITEURS_GET_1

```
final int com.lasalle.meeting.Communication.NB_DELIMITEURS_GET_1 = 6  [static]
```

Définition à la ligne 47 du fichier Communication.java.

7.1.4.10 NB_DELIMITEURS_GET_2

```
final int com.lasalle.meeting.Communication.NB_DELIMITEURS_GET_2 = 3  [static]
```

Définition à la ligne 48 du fichier Communication.java.

7.1.4.11 NB_DELIMITEURS_GET_3

```
final int com.lasalle.meeting.Communication.NB_DELIMITEURS_GET_3 = 1  [static]
```

Définition à la ligne 49 du fichier Communication.java.

7.1.4.12 PORT

```
final int com.lasalle.meeting.Communication.PORT = 5000  [static], [private]
```

port d'ecoute des portiers

Définition à la ligne 30 du fichier Communication.java.

7.1.4.13 socket

```
DatagramSocket com.lasalle.meeting.Communication.socket  [private]
```

attribut récuperant les informations de la socket

Attributs

Définition à la ligne 37 du fichier Communication.java.

7.1.4.14 TAG

```
final String com.lasalle.meeting.Communication.TAG = "Communication"  [static], [private]
```

TAG utilisé dans les log.

Constantes

Définition à la ligne 28 du fichier Communication.java.

7.1.4.15 TIMEOUT_RECEPTION_REPONSE

```
final int com.lasalle.meeting.Communication.TIMEOUT_RECEPTION_REPONSE = 30000  [static]
```

temps maximum d'une réponse d'un portier

Définition à la ligne 31 du fichier Communication.java.

Référencé par com.lasalle.meeting.Communication.Communication().

7.1.4.16 trame

```
String com.lasalle.meeting.Communication.trame  [private]
```

attribut récuperant la trame

Définition à la ligne 40 du fichier Communication.java.

Référencé par com.lasalle.meeting.Communication.recevoir().

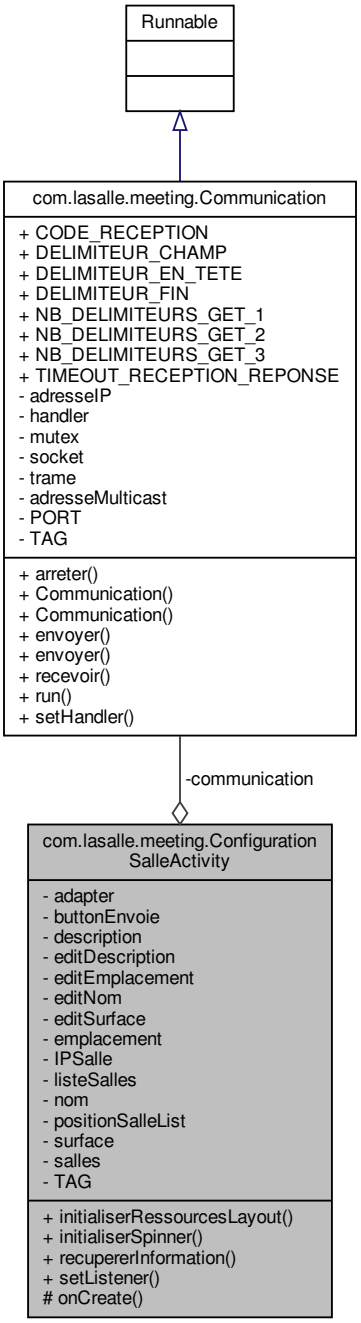
La documentation de cette classe a été générée à partir du fichier suivant :

— Communication.java

7.2 Référence de la classe com.lasalle.meeting.ConfigurationSalleActivity

Déclaration de la classe ConfigurationSalleActivity.

Graphe de collaboration de com.lasalle.meeting.ConfigurationSalleActivity :



Fonctions membres publiques

- void initialiserRessourcesLayout ()
  Récupère et initialise les widgets du layout activity_configuration_salle.
- void initialiserSpinner ()
  initialise la vue
- void recupererInformation ()
  recuepe et applique les informations mis dans les layouts
- void setListener ()
  applique les listener sur les layouts approprié

Fonctions membres protégées

— void onCreate (Bundle savedInstanceState)
Méthode appelée à la création de l'activité ConfigurationSalleActivity.

Attributs privés

— ArrayAdapter< String > adapter
l'adaptateur
— Button buttonEnvoie
layout du bouton envoie
— Communication communication = null
attribut permetant d'envoyer une requête
— String description =""
attribut de la description de la salle
— EditText editDescription
layout récuperant la description donné
— EditText editEmplacement
layout récuperant l'emplacement donné
— EditText editNom
layout récuperant le nom donné
— EditText editSurface
layout récuperant la surface donné
— String emplacement =""
attribut de l'emplacement de la salle
— List< String > IPSalle
les données traité
— Spinner listeSalles
la vue
— String nom =""
attribut du nom de la salle
— int positionSalleList = 0
position dans la vue
— String surface =""
attribut de la surface de la salle

Attributs privés statiques

— static Vector< Salle > salles
les données non traité
— static final String TAG = "ConfigurationSalleActivity"
TAG utilisé pour les logs.

7.2.1 Description détaillée

Déclaration de la classe ConfigurationSalleActivity.

Définition à la ligne 32 du fichier ConfigurationSalleActivity.java.

7.2.2 Documentation des fonctions membres

7.2.2.1 initialiserRessourcesLayout()

```
void com.lasalle.meeting.ConfigurationSalleActivity.initialiserRessourcesLayout ( )
```

Récupère et initialise les widgets du layout activity_configuration_salle.

Renvoie

void

Définition à la ligne 84 du fichier ConfigurationSalleActivity.java.

Référencé par com.lasalle.meeting.ConfigurationSalleActivity.onCreate().

```
00085     {
00086         Log.d(TAG, "initialiserRessourcesLayout()");
00087 
00088         listeSalles = (Spinner)findViewById(R.id.listeSalles);
00089         editNom = (EditText)findViewById(R.id.EditNom);
00090         editEmplacement= (EditText)findViewById(R.id.EditEmplacement);
00091         editDescription= (EditText)findViewById(R.id.EditDescription);
00092         editSurface= (EditText)findViewById(R.id.EditSurface);
00093         buttonEnvoie= (Button)findViewById(R.id.buttonEnvoie);
00094     }
```

7.2.2.2 initialiserSpinner()

```
void com.lasalle.meeting.ConfigurationSalleActivity.initialiserSpinner ( )
```

initialise la vue

Renvoie

void

Définition à la ligne 100 du fichier ConfigurationSalleActivity.java.

Références com.lasalle.meeting.MainActivity.getMesSalles(), et com.lasalle.meeting.ConfigurationSalleActivity.IPSalle.

Référencé par com.lasalle.meeting.ConfigurationSalleActivity.onCreate().

```
00101     {
00102         Log.d(TAG, "initialiserSpinner()");
00103 
00104         IPSalle = new ArrayList<String>();
00105 
00106         salles = MainActivity.getMesSalles();
00107 
00108         for(int i = 0; i < salles.size(); ++i)
00109         {
00110             Log.d(TAG, "Ajout adresse IP : " + salles.elementAt(i).getAdresseIP());
00111             IPSalle.add(salles.elementAt(i).getAdresseIP());
00112         }
00113 
00114         adapter = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item,
      IPSalle);
00115         adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
00116 
00117         listeSalles.setAdapter(adapter);
00118 
00119         listeSalles.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()
00120         {
00121             @Override
00122             public void onItemSelected(AdapterView<?> arg0, View arg1, int position, long id)
00123             {
00124                 positionSalleList = position;
00125                 Log.d(TAG, "position : " + position + " - " + "nom : " +
      IPSalle.get(position));
00126             }
00127             @Override
00128             public void onNothingSelected(AdapterView<?> arg0)
00129             {
00130             }
00131         });
00132     }
```

7.2.2.3 onCreate()

```
void com.lasalle.meeting.ConfigurationSalleActivity.onCreate (
            Bundle savedInstanceState )  [protected]
```

Méthode appelée à la création de l'activité ConfigurationSalleActivity.

Paramètres

savedInstanceState	

Renvoie

void

Définition à la ligne 66 du fichier ConfigurationSalleActivity.java.

Références com.lasalle.meeting.MainActivity.getCommunication(), com.lasalle.meeting.ConfigurationSalleActivity.initialiser↩RessourcesLayout(), com.lasalle.meeting.ConfigurationSalleActivity.initialiserSpinner(), et com.lasalle.meeting.ConfigurationSalle↩Activity.setListener().

```
00067     {
00068         Log.d(TAG, "onCreate()");
00069 
00070         super.onCreate(savedInstanceState);
00071         setContentView(R.layout.activity_configuration_salle);
00072 
00073         communication = MainActivity.getCommunication();
00074 
00075         initialiserRessourcesLayout();
00076         initialiserSpinner();
00077         setListener();
00078     }
```

7.2.2.4 recupererInformation()

```
void com.lasalle.meeting.ConfigurationSalleActivity.recupererInformation ( )
```

recuepe et applique les informations mis dans les layouts

Renvoie

void

Définition à la ligne 164 du fichier ConfigurationSalleActivity.java.

Référencé par com.lasalle.meeting.ConfigurationSalleActivity.setListener().

```
00165     {
00166         Log.d(TAG, "recupererInformation()");
00167 
00168         nom = editNom.getText().toString();
00169         description = editDescription.getText().toString();
00170         emplacement = editEmplacement.getText().toString();
00171         surface = editSurface.getText().toString();
00172     }
```

7.2.2.5 setListener()

```
void com.lasalle.meeting.ConfigurationSalleActivity.setListener ( )
```

applique les listener sur les layouts approprié

Renvoie

void

Définition à la ligne 138 du fichier ConfigurationSalleActivity.java.

Références com.lasalle.meeting.Communication.envoyer(), et com.lasalle.meeting.ConfigurationSalleActivity.recupererInformation().

Référencé par com.lasalle.meeting.ConfigurationSalleActivity.onCreate().

```
00139     {
00140         Log.d(TAG, "setListener()");
00141 
00142         buttonEnvoie.setOnClickListener(
00143             new View.OnClickListener()
00144             {
00145                 @Override
00146                 public void onClick(View v)
00147                 {
00148                     recupererInformation();
00149 
00150                     Log.d(TAG,"trame : $SET;1;" + nom + ";" + description + ";" +
      emplacement + ";" + surface + "\r\n" + IPSalle.get(
      positionSalleList));
00151                     if(communication != null)
00152                     {
00153                         communication.envoyer("$SET;1;" +
      nom + ";" + description + ";" + emplacement + ";" +
      surface + "\r\n", IPSalle.get(positionSalleList));
00154                     }
00155                 }
00156             }
00157         );
00158     }
```

7.2.3 Documentation des données membres

7.2.3.1 adapter

```
ArrayAdapter<String> com.lasalle.meeting.ConfigurationSalleActivity.adapter  [private]
```

l'adaptateur

Définition à la ligne 43 du fichier ConfigurationSalleActivity.java.

7.2.3.2 buttonEnvoie

```
Button com.lasalle.meeting.ConfigurationSalleActivity.buttonEnvoie  [private]
```

layout du bouton envoie

Définition à la ligne 58 du fichier ConfigurationSalleActivity.java.

7.2.3.3 communication

```
Communication com.lasalle.meeting.ConfigurationSalleActivity.communication = null  [private]
```

attribut permetant d'envoyer une requête

Définition à la ligne 45 du fichier ConfigurationSalleActivity.java.

7.2.3.4 description

```
String com.lasalle.meeting.ConfigurationSalleActivity.description =""  [private]
```

attribut de la description de la salle

Définition à la ligne 49 du fichier ConfigurationSalleActivity.java.

7.2.3.5 editDescription

```
EditText com.lasalle.meeting.ConfigurationSalleActivity.editDescription  [private]
```

layout récuperant la description donné

Définition à la ligne 56 du fichier ConfigurationSalleActivity.java.

7.2.3.6 editEmplacement

```
EditText com.lasalle.meeting.ConfigurationSalleActivity.editEmplacement  [private]
```

layout récuperant l'emplacement donné

Définition à la ligne 55 du fichier ConfigurationSalleActivity.java.

7.2.3.7 editNom

```
EditText com.lasalle.meeting.ConfigurationSalleActivity.editNom  [private]
```

layout récuperant le nom donné

Ressources layout activity_main

Définition à la ligne 54 du fichier ConfigurationSalleActivity.java.

7.2.3.8 editSurface

```
EditText com.lasalle.meeting.ConfigurationSalleActivity.editSurface  [private]
```

layout récuperant la surface donné

Définition à la ligne 57 du fichier ConfigurationSalleActivity.java.

7.2.3.9 emplacement

```
String com.lasalle.meeting.ConfigurationSalleActivity.emplacement =""  [private]
```

attribut de l'emplacement de la salle

Définition à la ligne 48 du fichier ConfigurationSalleActivity.java.

7.2.3.10 IPSalle

```
List<String> com.lasalle.meeting.ConfigurationSalleActivity.IPSalle  [private]
```

les données traité

Définition à la ligne 42 du fichier ConfigurationSalleActivity.java.

Référencé par com.lasalle.meeting.ConfigurationSalleActivity.initialiserSpinner().

7.2.3.11 listeSalles

```
Spinner com.lasalle.meeting.ConfigurationSalleActivity.listeSalles  [private]
```

la vue

Attributs

Définition à la ligne 41 du fichier ConfigurationSalleActivity.java.

7.2.3.12 nom

```
String com.lasalle.meeting.ConfigurationSalleActivity.nom =""  [private]
```

attribut du nom de la salle

Définition à la ligne 47 du fichier ConfigurationSalleActivity.java.

7.2.3.13 positionSalleList

```
int com.lasalle.meeting.ConfigurationSalleActivity.positionSalleList = 0  [private]
```

position dans la vue

Définition à la ligne 46 du fichier ConfigurationSalleActivity.java.

7.2.3.14 salles

```
Vector<Salle> com.lasalle.meeting.ConfigurationSalleActivity.salles  [static], [private]
```

les données non traité

Définition à la ligne 44 du fichier ConfigurationSalleActivity.java.

7.2.3.15 surface

```
String com.lasalle.meeting.ConfigurationSalleActivity.surface =""  [private]
```

attribut de la surface de la salle

Définition à la ligne 50 du fichier ConfigurationSalleActivity.java.

7.2.3.16 TAG

```
final String com.lasalle.meeting.ConfigurationSalleActivity.TAG = "ConfigurationSalleActivity"  [static],
[private]
```

TAG utilisé pour les logs.

Constantes

Définition à la ligne 37 du fichier ConfigurationSalleActivity.java.

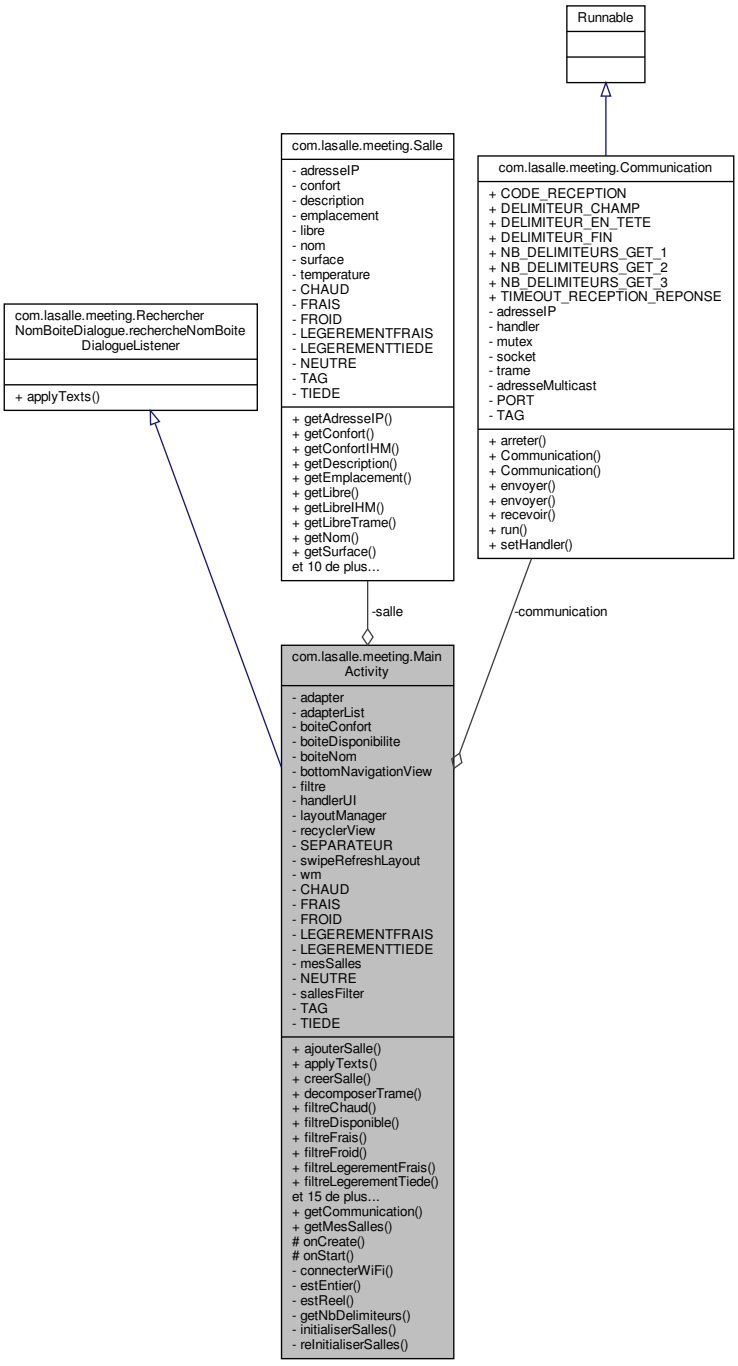
La documentation de cette classe a été générée à partir du fichier suivant :

— ConfigurationSalleActivity.java

7.3 Référence de la classe com.lasalle.meeting.MainActivity

Déclaration de la classe MainActivity.

Graphe de collaboration de com.lasalle.meeting.MainActivity :



Fonctions membres publiques

- void ajouterSalle (Salle maSalle)
  ajoute une salle au vecteur (mesSalles)
- void applyTexts (String nomSalleRechercher)
- Salle creerSalle (String[ ] trameDecompose, String adresseIP)

créer une salle a partir de la trame
- String [ ] decomposerTrame (String trame)
  découpe la trame
- void filtreChaud ()
- void filtreDisponible ()
- void filtreFrais ()
- void filtreFroid ()
- void filtreLegeremementFrais ()
- void filtreLegeremementTiede ()
- void filtreNeutre ()
- void filtreNom (String nomSalleRechercher)
- void filtreNonDisponible ()
- void filtreTiede ()
- void initialiserBoiteDialogue ()
  initialise les boites de dialogue
- void initialiserFiltre (int positionFiltre)
- void initialiserRessourcesLayout ()
  Récupère et initialise les widgets du layout activity_main.
- void initialiserSpinner ()
  initialise la vue
- boolean onCreateOptionsMenu (Menu menu)
  Méthode appelée au démarrage de l'activité MainActivity.
- boolean onOptionsItemSelected (MenuItem item)
  Méthode appelée quand on appuye sur boutons du menu.
- void rafraichir (Vector< Salle > mesSalles)
  rafraichis mon affichage
- void setListenBouton ()
  applique les listener sur les layouts approprié
- boolean verifierChangementSalle (Salle maSalle, int positionMemeSalle)
  Vérifie que la salle n'est pas different que dans le vecteur, r'envoye true si il y a eu une modifiaction, r'envoye false si il y en a pas eux.
- int verifierExistenceSalle (Salle maSalle)
  verifie que la salle n'exisite pas déjà dans le vecteur, r'envoye -1 si il n'en existe pas, r'envoye la position de la salle dans le vecteur si il existe
- int verifierExistenceSalle (String adresseIP)

Fonctions membres publiques statiques

- static Communication getCommunication ()
  retourne mon attribut communication
- static Vector< Salle > getMesSalles ()
  retourne le vecteur de salle

Fonctions membres protégées

- void onCreate (Bundle savedInstanceState)
  Méthode appelée à la création de l'activité MainActivity.
- void onStart ()
  Méthode appelée au démarrage de l'activité MainActivity.

Fonctions membres privées

- void connecterWiFi ()
  méthode permettant de se connecter au wi-fi
- boolean estEntier (String donnee)
- boolean estReel (String donnee)
- int getNbDelimiteurs (String trame)
  retourne le nombre de limiteurs dans la trame
- void initialiserSalles ()
  initialise le vecteur, les afficheurs pour les salles
- void reInitialiserSalles ()
  initialise le vecteur, les afficheurs pour les salles

Attributs privés

- RecyclerView.Adapter adapter
  l'adaptateur pour la vue des salles
- ArrayAdapter< String > adapterList
  l'adaptateur
- AlertDialog.Builder boiteConfort
  boite de dialogue pour le filtre confort
- AlertDialog.Builder boiteDisponibilite
  boite de dialogue pour le filtre disponibilité
- AlertDialog.Builder boiteNom
  boite de dialogue pour le filtre nom
- BottomNavigationView bottomNavigationView
  layout permettant d'avoir un menu de navigation (en haut)
- Spinner filtre
  la vue
- Handler handlerUI
  permet de récuperer les trames
- RecyclerView.LayoutManager layoutManager
  le gestionnaire de mise en page
- RecyclerView recyclerView
  la vue des salles
- Salle salle = null
  attribut salle
- final String SEPARATEUR = " ;"
  séparateur utilisé dans le protocole meeting
- SwipeRefreshLayout swipeRefreshLayout
  layout permettant de rafraichir
- WifiManager wm = null
  attribut permetant de voir la connection au wi-fi

Attributs privés statiques

- static final int CHAUD = 3
  Constant niveau de confort CHAUD.
- static Communication communication = null
  attribut permetant d'envoyer des requêtes
- static final int FRAIS = -2
  Constant niveau de confort FRAIS.
- static final int FROID = -3
  Constant niveau de confort FROID.
- static final int LEGEREMENTFRAIS = -1
  Constant niveau de confort LEGEREMENTFRAIS.
- static final int LEGEREMENTTIEDE = 1
  Constant niveau de confort LEGEREMENTTIEDE.
- static Vector< Salle > mesSalles
  Vecteur contenant mes salles (moyen de stockage)
- static final int NEUTRE = 0
  Constant niveau de confort NEUTRE.
- static Vector< Salle > sallesFilter
  Vecteur contenant mes salles aprés le filtre appliquer par l'utilisateur.
- static final String TAG = "MainActivity"
  TAG utilisé pour les logs.
- static final int TIEDE = 2
  Constant niveau de confort TIEDE.

7.3.1 Description détaillée

Déclaration de la classe MainActivity.

Définition à la ligne 52 du fichier MainActivity.java.

7.3.2 Documentation des fonctions membres

7.3.2.1 ajouterSalle()

```
void com.lasalle.meeting.MainActivity.ajouterSalle (
            Salle maSalle )
```

ajoute une salle au vecteur (mesSalles)

Paramètres

maSalle	Salle

Renvoie

void

Définition à la ligne 286 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.adapter, com.lasalle.meeting.MainActivity.verifierChangementSalle(), et com.lasalle.meeting.MainActivity.verifierExistenceSalle().

```
00287     {
00288         int positionMemeSalle = verifierExistenceSalle(maSalle);
00289
00290         if(positionMemeSalle == -1)
00291         {
00292             mesSalles.add(maSalle);
00293             adapter.notifyDataSetChanged();
00294         }
00295         else if(verifierChangementSalle(maSalle, positionMemeSalle))
00296         {
00297             mesSalles.removeElementAt(positionMemeSalle);
00298             mesSalles.add(maSalle);
00299             adapter.notifyDataSetChanged();
00300         }
00301     }
```

7.3.2.2 applyTexts()

```
void com.lasalle.meeting.MainActivity.applyTexts (
            String nomSalleRechercher )
```

Implémente com.lasalle.meeting.RechercherNomBoiteDialogue.rechercheNomBoiteDialogueListener.

Définition à la ligne 817 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.filtreNom().

```
00817                                                   {
00818         filtreNom(nomSalleRechercher);
00819     }
```

7.3.2.3 connecterWiFi()

```
void com.lasalle.meeting.MainActivity.connecterWiFi ( )  [private]
```

méthode permettant de se connecter au wi-fi

Renvoie

void

Définition à la ligne 217 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.handlerUI.

Référencé par com.lasalle.meeting.MainActivity.onCreate().

```
00218     {
00219         wm = (WifiManager) getApplicationContext().getSystemService(Context.WIFI_SERVICE);
00220         if (!wm.isWifiEnabled())
00221         {
00222             Log.d(TAG, "connecterWiFi() WiFi indisponible !");
00223             wm.setWifiEnabled(true);
00224         }
00225         else
00226         {
00227             Log.d(TAG, "connecterWiFi() WiFi disponible");
00228         }
00229
00230         WifiInfo wi = wm.getConnectionInfo();
00231         Log.d(TAG, "connecterWiFi() " + wi.toString() + " " + wi.getIpAddress() + " " + wi.getMacAddress
      ());
00232
00233         DhcpInfo di = wm.getDhcpInfo();
00234         Log.d(TAG, "connecterWiFi() " + di.toString());
00235
00236         communication = new Communication(handlerUI);
00237         Thread tCommunicationUDP = new Thread(communication, "Communication");
00238         tCommunicationUDP.start();
00239     }
```

7.3.2.4 creerSalle()

```
Salle com.lasalle.meeting.MainActivity.creerSalle (
            String [] trameDecompose,
            String adresseIP )
```

créer une salle a partir de la trame

Paramètres

trameDecompose	String[] , adresseIP String

Renvoie

salle

Définition à la ligne 445 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.estEntier(), com.lasalle.meeting.MainActivity.estReel(), et com.lasalle.meeting.Main↩ Activity.salle.

```
00446     {
00447         Log.d(TAG, "creerSalle() adresseIP = " + adresseIP);
00448         if(!trameDecompose[0].isEmpty())
00449         {
00450             Log.d(TAG, "creerSalle() trameDecompose[0] : " + trameDecompose[0] + " (nomSalle)");
00451             Log.d(TAG, "creerSalle() trameDecompose[1] : " + trameDecompose[1] + " (description)");
00452             Log.d(TAG, "creerSalle() trameDecompose[2] : " + trameDecompose[2] + " (emplacement)");
00453             Log.d(TAG, "creerSalle() trameDecompose[3] : " + trameDecompose[3] + " (surface)");
00454             Log.d(TAG, "creerSalle() trameDecompose[4] : " + trameDecompose[4] + " (disponibilité)");
00455             Log.d(TAG, "creerSalle() trameDecompose[5] : " + trameDecompose[5] + " (niveauDeConfort)");
00456             Log.d(TAG, "creerSalle() trameDecompose[6] : " + trameDecompose[6] + " (température)");
00457             int surface = 0;
00458             if(estEntier(trameDecompose[3]))
00459                 surface = Integer.parseInt(trameDecompose[3]);
00460             int disponible = 0;
00461             if(estEntier(trameDecompose[4]))
00462                 disponible = Integer.parseInt(trameDecompose[4]);
00463             int niveauConfort = 0;
00464             if(estEntier(trameDecompose[5]))
00465                 niveauConfort = Integer.parseInt(trameDecompose[5]);
00466             float temperature = 0;
00467             if(estReel(trameDecompose[6]))
00468                 temperature = Float.parseFloat(trameDecompose[6]);
00469 
00470             salle = new Salle(trameDecompose[0], trameDecompose[1], trameDecompose[2], disponible,
      surface, niveauConfort, temperature, adresseIP);
00471             return salle;
00472         }
00473         else
00474         {
00475             Log.d(TAG, "creerSalle() pas de nom de salle");
00476             String inconnuString = "???";
00477             int inconnuInt = 1;
00478 
00479             salle = new Salle(inconnuString, inconnuString , inconnuString , Integer.parseInt(
      trameDecompose[4]), inconnuInt,  Integer.parseInt(trameDecompose[5]), Float.parseFloat(trameDecompose[6]), adresseIP
      );
00480             return salle;
00481         }
00482     }
```

7.3.2.5 decomposerTrame()

```
String [] com.lasalle.meeting.MainActivity.decomposerTrame (
             String trame )
```

découpe la trame

Paramètres

trame	String

Renvoie

un tableau de string (String[])

Définition à la ligne 489 du fichier MainActivity.java.

```
00490     {
00491         String[] tramesDecompose = trame.split(SEPARATEUR);
00492         return tramesDecompose;
00493     }
```

7.3.2.6 estEntier()

```
boolean com.lasalle.meeting.MainActivity.estEntier (
            String donnee )  [private]
```

Définition à la ligne 531 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.creerSalle().

```
00532     {
00533         try
00534         {
00535             Integer.parseInt(donnee);
00536         }
00537         catch(NumberFormatException e)
00538         {
00539             return false;
00540         }
00541         catch(NullPointerException e)
00542         {
00543             return false;
00544         }
00545 
00546         return true;
00547     }
```

7.3.2.7 estReel()

```
boolean com.lasalle.meeting.MainActivity.estReel (
            String donnee )  [private]
```

Définition à la ligne 549 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.creerSalle().

```
00550     {
00551         try
00552         {
00553             Float.parseFloat(donnee);
00554         }
00555         catch(NumberFormatException e)
00556         {
00557             return false;
00558         }
00559         catch(NullPointerException e)
00560         {
00561             return false;
00562         }
00563 
00564         return true;
00565     }
```

7.3.2.8 filtreChaud()

```
void com.lasalle.meeting.MainActivity.filtreChaud ( )
```

Définition à la ligne 786 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.CHAUD, et com.lasalle.meeting.MainActivity.reInitialiserSalles().

Référencé par com.lasalle.meeting.MainActivity.initialiserBoiteDialogue().

```
00787     {
00788         Log.d(TAG, "filtreChaud()");
00789         sallesFilter = new Vector<Salle>();
00790 
00791         for(int i = 0; i < mesSalles.size(); i++)
00792         {
00793             if(mesSalles.elementAt(i).getConfort() == CHAUD)
00794             {
00795                 sallesFilter.add(mesSalles.elementAt(i));
00796             }
00797         }
00798         reInitialiserSalles();
00799     }
```

7.3.2.9 filtreDisponible()

```
void com.lasalle.meeting.MainActivity.filtreDisponible ( )
```

Définition à la ligne 681 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.reInitialiserSalles().

Référencé par com.lasalle.meeting.MainActivity.initialiserBoiteDialogue().

```
00682     {
00683         Log.d(TAG, "filtreDisponible()");
00684         sallesFilter = new Vector<Salle>();
00685 
00686         for(int i = 0; i < mesSalles.size(); i++)
00687         {
00688             if(mesSalles.elementAt(i).getLibre())
00689             {
00690                 sallesFilter.add(mesSalles.get(i));
00691             }
00692         }
00693         reInitialiserSalles();
00694     }
```

7.3.2.10 filtreFrais()

```
void com.lasalle.meeting.MainActivity.filtreFrais ( )
```

Définition à la ligne 771 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.FRAIS, et com.lasalle.meeting.MainActivity.reInitialiserSalles().

Référencé par com.lasalle.meeting.MainActivity.initialiserBoiteDialogue().

```
00772     {
00773         Log.d(TAG, "filtreFrais()");
00774         sallesFilter = new Vector<Salle>();
00775 
00776         for(int i = 0; i < mesSalles.size(); i++)
00777         {
00778             if(mesSalles.elementAt(i).getConfort() == FRAIS)
00779             {
00780                 sallesFilter.add(mesSalles.elementAt(i));
00781             }
00782         }
00783         reInitialiserSalles();
00784     }
```

7.3.2.11 filtreFroid()

```
void com.lasalle.meeting.MainActivity.filtreFroid ( )
```

Définition à la ligne 801 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.FROID, et com.lasalle.meeting.MainActivity.reInitialiserSalles().

Référencé par com.lasalle.meeting.MainActivity.initialiserBoiteDialogue().

```
00802     {
00803         Log.d(TAG, "filtreFroid()");
00804         sallesFilter = new Vector<Salle>();
00805 
00806         for(int i = 0; i < mesSalles.size(); i++)
00807         {
00808             if(mesSalles.elementAt(i).getConfort() == FROID)
00809             {
00810                 sallesFilter.add(mesSalles.elementAt(i));
00811             }
00812         }
00813         reInitialiserSalles();
00814     }
```

7.3.2.12 filtreLegerementFrais()

```
void com.lasalle.meeting.MainActivity.filtreLegerementFrais ( )
```

Définition à la ligne 756 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.LEGEREMENTFRAIS, et com.lasalle.meeting.MainActivity.reInitialiserSalles().

Référencé par com.lasalle.meeting.MainActivity.initialiserBoiteDialogue().

```
00757     {
00758         Log.d(TAG, "filtreLegerementFrais()");
00759         sallesFilter = new Vector<Salle>();
00760 
00761         for(int i = 0; i < mesSalles.size(); i++)
00762         {
00763             if(mesSalles.elementAt(i).getConfort() == LEGEREMENTFRAIS)
00764             {
00765                 sallesFilter.add(mesSalles.elementAt(i));
00766             }
00767         }
00768         reInitialiserSalles();
00769     }
```

7.3.2.13 filtreLegerementTiede()

```
void com.lasalle.meeting.MainActivity.filtreLegerementTiede ( )
```

Définition à la ligne 726 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.LEGEREMENTTIEDE, et com.lasalle.meeting.MainActivity.reInitialiserSalles().

Référencé par com.lasalle.meeting.MainActivity.initialiserBoiteDialogue().

```
00727     {
00728         Log.d(TAG, "filtreLegerementTiede()");
00729         sallesFilter = new Vector<Salle>();
00730 
00731         for(int i = 0; i < mesSalles.size(); i++)
00732         {
00733             if(mesSalles.elementAt(i).getConfort() == LEGEREMENTTIEDE)
00734             {
00735                 sallesFilter.add(mesSalles.elementAt(i));
00736             }
00737         }
00738         reInitialiserSalles();
00739     }
```

7.3.2.14 filtreNeutre()

```
void com.lasalle.meeting.MainActivity.filtreNeutre ( )
```

Définition à la ligne 741 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.NEUTRE, et com.lasalle.meeting.MainActivity.reInitialiserSalles().

Référencé par com.lasalle.meeting.MainActivity.initialiserBoiteDialogue().

```
00742     {
00743         Log.d(TAG, "filtreNeutre()");
00744         sallesFilter = new Vector<Salle>();
00745 
00746         for(int i = 0; i < mesSalles.size(); i++)
00747         {
00748             if(mesSalles.elementAt(i).getConfort() == NEUTRE)
00749             {
00750                 sallesFilter.add(mesSalles.elementAt(i));
00751             }
00752         }
00753         reInitialiserSalles();
00754     }
```

7.3.2.15 filtreNom()

```
void com.lasalle.meeting.MainActivity.filtreNom (
             String nomSalleRechercher )
```

Définition à la ligne 821 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.reInitialiserSalles().

Référencé par com.lasalle.meeting.MainActivity.applyTexts().

```
00822     {
00823         Log.d(TAG, "filtreNom()");
00824         sallesFilter = new Vector<Salle>();
00825 
00826         for(int i = 0; i < mesSalles.size(); i++)
00827         {
00828             if(mesSalles.elementAt(i).getNom().equals(nomSalleRechercher))
00829             {
00830                 sallesFilter.add(mesSalles.elementAt(i));
00831             }
00832         }
00833         reInitialiserSalles();
00834     }
```

7.3.2.16 filtreNonDisponible()

```
void com.lasalle.meeting.MainActivity.filtreNonDisponible ( )
```

Définition à la ligne 696 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.reInitialiserSalles().

Référencé par com.lasalle.meeting.MainActivity.initialiserBoiteDialogue().

```
00697     {
00698         Log.d(TAG, "filtreNonDisponible()");
00699         sallesFilter = new Vector<Salle>();
00700 
00701         for(int i = 0; i < mesSalles.size(); i++)
00702         {
00703             if(!mesSalles.elementAt(i).getLibre())
00704             {
00705                 sallesFilter.add(mesSalles.elementAt(i));
00706             }
00707         }
00708         reInitialiserSalles();
00709     }
```

7.3.2.17 filtreTiede()

```
void com.lasalle.meeting.MainActivity.filtreTiede ( )
```

Définition à la ligne 711 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.reInitialiserSalles(), et com.lasalle.meeting.MainActivity.TIEDE.

Référencé par com.lasalle.meeting.MainActivity.initialiserBoiteDialogue().

```
00712     {
00713         Log.d(TAG, "filtreTiede()");
00714         sallesFilter = new Vector<Salle>();
00715 
00716         for(int i = 0; i < mesSalles.size(); i++)
00717         {
00718             if(mesSalles.elementAt(i).getConfort() == TIEDE)
00719             {
00720                 sallesFilter.add(mesSalles.elementAt(i));
00721             }
00722         }
00723         reInitialiserSalles();
00724     }
```

7.3.2.18 getCommunication()

```
static Communication com.lasalle.meeting.MainActivity.getCommunication ( )  [static]
```

retourne mon attribut communication

Renvoie

communication

Définition à la ligne 526 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.communication.

Référencé par com.lasalle.meeting.SalleActivity.onCreate(), et com.lasalle.meeting.ConfigurationSalleActivity.onCreate().

```
00527     {
00528         return communication;
00529     }
```

7.3.2.19 getMesSalles()

```
static Vector<Salle> com.lasalle.meeting.MainActivity.getMesSalles ( )  [static]
```

retourne le vecteur de salle

Renvoie

Vector<Salle>

Définition à la ligne 517 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.mesSalles.

Référencé par com.lasalle.meeting.ConfigurationSalleActivity.initialiserSpinner().

```
00518     {
00519         return mesSalles;
00520     }
```

7.3.2.20 getNbDelimiteurs()

```
int com.lasalle.meeting.MainActivity.getNbDelimiteurs (
            String trame )  [private]
```

retourne le nombre de limiteurs dans la trame

Paramètres

trame	String

Renvoie

int

Définition à la ligne 500 du fichier MainActivity.java.

```
00501     {
00502         int nb = 0;
00503         for (int i = 0; i < trame.length(); i++)
00504         {
00505             if (trame.charAt(i) == ';')
00506             {
00507                 nb++;
00508             }
00509         }
00510         return nb;
00511     }
```

7.3.2.21 initialiserBoiteDialogue()

```
void com.lasalle.meeting.MainActivity.initialiserBoiteDialogue ( )
```

initialise les boites de dialogue

Renvoie

void

Définition à la ligne 571 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.boiteConfort, com.lasalle.meeting.MainActivity.boiteDisponibilite, com.lasalle.meeting.↩ MainActivity.filtreChaud(), com.lasalle.meeting.MainActivity.filtreDisponible(), com.lasalle.meeting.MainActivity.filtreFrais(), com.↩ lasalle.meeting.MainActivity.filtreFroid(), com.lasalle.meeting.MainActivity.filtreLegeremmentFrais(), com.lasalle.meeting.MainActivity.↩ filtreLegeremmentTiede(), com.lasalle.meeting.MainActivity.filtreNeutre(), com.lasalle.meeting.MainActivity.filtreNonDisponible(), et com.lasalle.meeting.MainActivity.filtreTiede().

Référencé par com.lasalle.meeting.MainActivity.initialiserRessourcesLayout().

```
00572     {
00573         boiteDisponibilite = new AlertDialog.Builder(this);
00574         boiteDisponibilite.setMessage("Voulez vous voir seulement les salles : ");
00575         boiteDisponibilite.setPositiveButton("Disponible", new DialogInterface.
      OnClickListener()
00576         {
00577             public void onClick(DialogInterface dialog, int which)
00578             {
00579                 filtreDisponible();
00580             }
00581         });
00582         boiteDisponibilite.setNegativeButton("Occupé", new DialogInterface.
      OnClickListener()
00583         {
00584             public void onClick(DialogInterface dialog, int which)
00585             {
00586                 filtreNonDisponible();
00587             }
00588         });
00589 
00590         boiteConfort = new AlertDialog.Builder(this);
00591         boiteConfort.setTitle("Choisir le niveau de confort");
00592         String[] niveauConfort = {"Chaud","Tiède","Légèrement tiède","Neutre","Légèrement fraiche","Fraiche
      ","Froid"};
00593         boiteConfort.setItems(niveauConfort, new DialogInterface.OnClickListener() {
00594             @Override
00595             public void onClick(DialogInterface dialog, int which) {
00596                 switch (which) {
00597                     case 0:
00598                         filtreChaud();
00599                         break;
00600                     case 1:
00601                         filtreTiede();
```

```
00602                       break;
00603                   case 2:
00604                       filtreLegerementTiede();
00605                       break;
00606                   case 3:
00607                       filtreNeutre();
00608                       break;
00609                   case 4:
00610                       filtreLegerementFrais();
00611                       break;
00612                   case 5:
00613                       filtreFrais();
00614                       break;
00615                   case 6:
00616                       filtreFroid();
00617                       break;
00618               }
00619           }
00620       });
00621   }
```

7.3.2.22 initialiserFiltre()

```
void com.lasalle.meeting.MainActivity.initialiserFiltre (
            int positionFiltre )
```

Définition à la ligne 660 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.boiteConfort, com.lasalle.meeting.MainActivity.boiteDisponibilite, et com.lasalle.↩ meeting.MainActivity.initialiserSalles().

Référencé par com.lasalle.meeting.MainActivity.initialiserSpinner().

```
00661   {
00662       if(positionFiltre == 0)
00663       {
00664           initialiserSalles();
00665       }
00666       else if(positionFiltre == 1)
00667       {
00668           boiteDisponibilite.show();
00669       }
00670       else if(positionFiltre == 2)
00671       {
00672           boiteConfort.show();
00673       }
00674       else if(positionFiltre == 3)
00675       {
00676           RechercherNomBoiteDialogue rechercherNomBoiteDialogue = new RechercherNomBoiteDialogue();
00677           rechercherNomBoiteDialogue.show(getSupportFragmentManager(), "rechercher une salle par son nom"
    );
00678       }
00679   }
```

7.3.2.23 initialiserRessourcesLayout()

```
void com.lasalle.meeting.MainActivity.initialiserRessourcesLayout ( )
```

Récupère et initialise les widgets du layout activity_main.

Renvoie

void

Définition à la ligne 160 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.initialiserBoiteDialogue(), com.lasalle.meeting.MainActivity.initialiserSpinner(), et com.lasalle.meeting.MainActivity.setListenBouton().

Référencé par com.lasalle.meeting.MainActivity.onCreate().

```
00161     {
00162         swipeRefreshLayout = (SwipeRefreshLayout)findViewById(R.id.swipeRefreshLayout);
00163         recyclerView = (RecyclerView)findViewById(R.id.listeSalle);
00164         bottomNavigationView = (BottomNavigationView) findViewById(R.id.
      bottomNavigationView);
00165         filtre = (Spinner)findViewById(R.id.filtre);
00166 
00167         setListenBouton();
00168         initialiserBoiteDialogue();
00169         initialiserSpinner();
00170     }
```

7.3.2.24 initialiserSalles()

```
void com.lasalle.meeting.MainActivity.initialiserSalles ( )  [private]
```

initialise le vecteur, les afficheurs pour les salles

Renvoie

void

Définition à la ligne 245 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.adapter, et com.lasalle.meeting.MainActivity.layoutManager.

Référencé par com.lasalle.meeting.MainActivity.initialiserFiltre(), et com.lasalle.meeting.MainActivity.onCreate().

```
00246     {
00247         Log.d(TAG, "initialiserSalles()");
00248 
00249         recyclerView.setHasFixedSize(true);
00250         layoutManager = new LinearLayoutManager(this);
00251 
00252         recyclerView.setLayoutManager(layoutManager);
00253 
00254         adapter = new SalleAdapter(mesSalles);
00255         recyclerView.setAdapter(adapter);
00256 
00257         // cf. appel à rafraichir() dans onStart()
00258         /*if(communication != null)
00259         {
00260             communication.envoyer("$GET;1\r\n"); // voir protocole
00261         }*/
00262     }
```

7.3.2.25 initialiserSpinner()

```
void com.lasalle.meeting.MainActivity.initialiserSpinner ( )
```

initialise la vue

Renvoie

void

Définition à la ligne 627 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.initialiserFiltre().

Référencé par com.lasalle.meeting.MainActivity.initialiserRessourcesLayout().

```
00628     {
00629         Log.d(TAG, "initialiserSpinner()");
00630 
00631         List choixFiltre;
00632         choixFiltre = new ArrayList<String>();
00633 
00634         choixFiltre.add("Aucun");
00635         choixFiltre.add("Disponibilité");
00636         choixFiltre.add("Confort");
00637         choixFiltre.add("Nom");
00638 
00639         adapterList = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item,
      choixFiltre);
00640         adapterList.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
00641 
00642         filtre.setAdapter(adapterList);
00643 
00644         filtre.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()
00645         {
00646             @Override
00647             public void onItemSelected(AdapterView<?> arg0, View arg1, int position, long id)
00648             {
00649                 Toast.makeText(getApplicationContext(), "Le choix du filtre est : " + position, Toast.
      LENGTH_SHORT).show();
00650                 Log.d(TAG, "position : " + position);
00651                 initialiserFiltre(position);
00652             }
00653             @Override
00654             public void onNothingSelected(AdapterView<?> arg0)
00655             {
00656             }
00657         });
00658     }
```

7.3.2.26 onCreate()

```
void com.lasalle.meeting.MainActivity.onCreate (
             Bundle savedInstanceState )  [protected]
```

Méthode appelée à la création de l'activité MainActivity.

Paramètres

savedInstanceState	

Renvoie

void

Définition à la ligne 96 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.connecterWiFi(), com.lasalle.meeting.MainActivity.initialiserRessourcesLayout(), et com.lasalle.meeting.MainActivity.initialiserSalles().

```
00097     {
00098         Log.d(TAG, "onCreate()");
00099
00100         super.onCreate(savedInstanceState);
00101         setContentView(R.layout.activity_main);
00102
00103         mesSalles = new Vector<Salle>();
00104
00105         initialiserRessourcesLayout();
00106         connecterWiFi();
00107         initialiserSalles();
00108     }
```

7.3.2.27 onCreateOptionsMenu()

```
boolean com.lasalle.meeting.MainActivity.onCreateOptionsMenu (
            Menu menu )
```

Méthode appelée au démarrage de l'activité MainActivity.

Renvoie

void

Définition à la ligne 128 du fichier MainActivity.java.

```
00129     {
00130         MenuInflater inflater = getMenuInflater();
00131         inflater.inflate(R.menu.menu, menu);
00132         return true;
00133     }
```

7.3.2.28 onOptionsItemSelected()

```
boolean com.lasalle.meeting.MainActivity.onOptionsItemSelected (
            MenuItem item )
```

Méthode appelée quand on appuye sur boutons du menu.

Renvoie

boolean

Définition à la ligne 140 du fichier MainActivity.java.

```
00141     {
00142         final Intent intent = new Intent(MainActivity.this, ConfigurationSalleActivity.class);
00143
00144         switch (item.getItemId())
00145         {
00146             case R.id.configurerSalle:
00147                 startActivity(intent);
00148                 return true;
00149             case R.id.aPropos:
00150                 Toast.makeText(getApplicationContext(), "La fonctionnalité à propos de l'application n'est
      pas encore disponible !", Toast.LENGTH_SHORT).show();
00151                 return true;
00152         }
00153         return super.onOptionsItemSelected(item);
00154     }
```

7.3.2.29 onStart()

```
void com.lasalle.meeting.MainActivity.onStart ( )  [protected]
```

Méthode appelée au démarrage de l'activité MainActivity.

Renvoie

void

Définition à la ligne 115 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.rafraichir().

```
00116     {
00117         super.onStart();
00118         Log.d(TAG, "onStart()");
00119 
00120         rafraichir(mesSalles);
00121     }
```

7.3.2.30 rafraichir()

```
void com.lasalle.meeting.MainActivity.rafraichir (
            Vector< Salle > mesSalles )
```

rafraichis mon affichage

Paramètres

mesSalles	Vector<Salle>

Renvoie

void

Définition à la ligne 377 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.adapter, et com.lasalle.meeting.Communication.envoyer().

Référencé par com.lasalle.meeting.MainActivity.onStart(), et com.lasalle.meeting.MainActivity.setListenBouton().

```
00378     {
00379         Log.d(TAG, "rafraichir()");
00380         if(communication != null)
00381             communication.envoyer("$GET;1\r\n"); // voir protocole
00382 
00383         swipeRefreshLayout.setRefreshing(false);
00384         adapter.notifyDataSetChanged();
00385     }
```

7.3.2.31 reInitialiserSalles()

```
void com.lasalle.meeting.MainActivity.reInitialiserSalles ( )  [private]
```

initialise le vecteur, les afficheurs pour les salles

Renvoie

void

Définition à la ligne 268 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.adapter, et com.lasalle.meeting.MainActivity.layoutManager.

Référencé par com.lasalle.meeting.MainActivity.filtreChaud(), com.lasalle.meeting.MainActivity.filtreDisponible(), com.lasalle.meeting.MainActivity.filtreFrais(), com.lasalle.meeting.MainActivity.filtreFroid(), com.lasalle.meeting.MainActivity.filtreLegerementFrais(), com.lasalle.meeting.MainActivity.filtreLegerementTiede(), com.lasalle.meeting.MainActivity.filtreNeutre(), com.lasalle.meeting.MainActivity.filtreNom(), com.lasalle.meeting.MainActivity.filtreNonDisponible(), et com.lasalle.meeting.MainActivity.filtreTiede().

```
00269     {
00270         Log.d(TAG, "reInitialiserSalles()");
00271 
00272         recyclerView.setHasFixedSize(true);
00273         layoutManager = new LinearLayoutManager(this);
00274 
00275         recyclerView.setLayoutManager(layoutManager);
00276 
00277         adapter = new SalleAdapter(sallesFilter);
00278         recyclerView.setAdapter(adapter);
00279     }
```

7.3.2.32 setListenBouton()

```
void com.lasalle.meeting.MainActivity.setListenBouton ( )
```

applique les listener sur les layouts approprié

Renvoie

void

Définition à la ligne 176 du fichier MainActivity.java.

Références com.lasalle.meeting.MainActivity.rafraichir().

Référencé par com.lasalle.meeting.MainActivity.initialiserRessourcesLayout().

```
00177     {
00178         swipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener(
    )
00179         {
00180             @Override
00181             public void onRefresh()
00182             {
00183                 rafraichir(mesSalles);
00184             }
00185         });
00186 
00187         bottomNavigationView.setOnNavigationItemSelectedListener(new
    BottomNavigationView.OnNavigationItemSelectedListener()
00188         {
00189             @Override
00190             public boolean onNavigationItemSelected(@NonNull MenuItem item)
00191             {
```

```
00192                 switch (item.getItemId())
00193                 {
00194                     case R.id.Salle:
00195                         rafraichir(mesSalles);
00196                         Toast.makeText(getApplicationContext(), "Rafraichisement", Toast.LENGTH_SHORT).show
      ();
00197                         return true;
00198                     case R.id.Favoris:
00199                         Toast.makeText(getApplicationContext(), "La fonctionnalité favoris n'est pas encore
       disponible !", Toast.LENGTH_SHORT).show();
00200                         return true;
00201                     case R.id.Rechercher:
00202                         /*final Intent intent = new Intent(MainActivity.this, RechercheActivity.class);
00203                         intent.putExtra("mesSalles", mesSalles);
00204                         startActivity(intent);*/
00205                         Toast.makeText(getApplicationContext(), "La fonctionnalité rechercher n'est pas
       encore disponible !", Toast.LENGTH_SHORT).show();
00206                         return true;
00207                 }
00208                 return false;
00209             }
00210         });
00211     }
```

7.3.2.33 verifierChangementSalle()

```
boolean com.lasalle.meeting.MainActivity.verifierChangementSalle (
            Salle maSalle,
            int positionMemeSalle )
```

Vérifie que la salle n'est pas different que dans le vecteur, r'envoye true si il y a eu une modifiaction, r'envoye false si il y en a pas eux.

Paramètres

maSalle	Salle, positionMemeSalle int

Renvoie

boolean

Définition à la ligne 338 du fichier MainActivity.java.

Références com.lasalle.meeting.Salle.getConfort(), com.lasalle.meeting.Salle.getDescription(), com.lasalle.meeting.Salle.getEmplacement(), com.lasalle.meeting.Salle.getLibre(), com.lasalle.meeting.Salle.getNom(), com.lasalle.meeting.Salle.getSurface(), et com.lasalle.meeting.Salle.getTemperature().

Référencé par com.lasalle.meeting.MainActivity.ajouterSalle().

```
00339     {
00340         if(maSalle.getLibre() == mesSalles.elementAt(positionMemeSalle).getLibre())
00341         {
00342             return true;
00343         }
00344         else if(maSalle.getTemperature() == mesSalles.elementAt(positionMemeSalle).getTemperature(
      ))
00345         {
00346             return true;
00347         }
00348         else if(maSalle.getConfort() == mesSalles.elementAt(positionMemeSalle).getConfort())
00349         {
00350             return true;
00351         }
00352         else if(maSalle.getNom().equals(mesSalles.elementAt(positionMemeSalle).getNom()))
00353         {
00354             return true;
00355         }
00356         else if(maSalle.getEmplacement().equals(mesSalles.elementAt(positionMemeSalle).
      getEmplacement()))
00357         {
```

```
00358             return true;
00359         }
00360         else if(maSalle.getDescription().equals(mesSalles.elementAt(positionMemeSalle).
      getDescription()))
00361         {
00362             return true;
00363         }
00364         else if(maSalle.getSurface() == mesSalles.elementAt(positionMemeSalle).getSurface())
00365         {
00366             return true;
00367         }
00368
00369         return false;
00370     }
```

7.3.2.34 verifierExistenceSalle() [1/2]

```
int com.lasalle.meeting.MainActivity.verifierExistenceSalle (
            Salle maSalle )
```

verifie que la salle n'exisite pas déjà dans le vecteur, r'envoye -1 si il n'en existe pas, r'envoye la position de la salle dans le vecteur si il existe

Paramètres

maSalle	Salle

Renvoie

int

Définition à la ligne 308 du fichier MainActivity.java.

Références com.lasalle.meeting.Salle.getAdresseIP().

Référencé par com.lasalle.meeting.MainActivity.ajouterSalle().

```
00309     {
00310         for(int i = 0; i < mesSalles.size(); ++i)
00311         {
00312             if(maSalle.getAdresseIP().equals(mesSalles.elementAt(i).getAdresseIP()))
00313             {
00314                 return i;
00315             }
00316         }
00317         return -1;
00318     }
```

7.3.2.35 verifierExistenceSalle() [2/2]

```
int com.lasalle.meeting.MainActivity.verifierExistenceSalle (
            String adresseIP )
```

Définition à la ligne 321 du fichier MainActivity.java.

```
00322     {
00323         for(int i = 0; i < mesSalles.size(); ++i)
00324         {
00325             if(mesSalles.elementAt(i).getAdresseIP().equals(adresseIP))
00326             {
00327                 return i;
00328             }
00329         }
00330         return -1;
00331     }
```

7.3.3 Documentation des données membres

7.3.3.1 adapter

```
RecyclerView.Adapter com.lasalle.meeting.MainActivity.adapter  [private]
```

l'adaptateur pour la vue des salles

Définition à la ligne 74 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.ajouterSalle(), com.lasalle.meeting.MainActivity.initialiserSalles(), com.lasalle.↩ meeting.MainActivity.rafraichir(), et com.lasalle.meeting.MainActivity.reInitialiserSalles().

7.3.3.2 adapterList

```
ArrayAdapter<String> com.lasalle.meeting.MainActivity.adapterList  [private]
```

l'adaptateur

Définition à la ligne 71 du fichier MainActivity.java.

7.3.3.3 boiteConfort

```
AlertDialog.Builder com.lasalle.meeting.MainActivity.boiteConfort  [private]
```

boite de dialogue pour le filtre confort

Définition à la ligne 77 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.initialiserBoiteDialogue(), et com.lasalle.meeting.MainActivity.initialiserFiltre().

7.3.3.4 boiteDisponibilite

```
AlertDialog.Builder com.lasalle.meeting.MainActivity.boiteDisponibilite  [private]
```

boite de dialogue pour le filtre disponibilité

Définition à la ligne 78 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.initialiserBoiteDialogue(), et com.lasalle.meeting.MainActivity.initialiserFiltre().

7.3.3.5 boiteNom

```
AlertDialog.Builder com.lasalle.meeting.MainActivity.boiteNom  [private]
```

boite de dialogue pour le filtre nom

Définition à la ligne 79 du fichier MainActivity.java.

7.3.3.6 bottomNavigationView

```
BottomNavigationView com.lasalle.meeting.MainActivity.bottomNavigationView  [private]
```

layout permettant d'avoir un menu de navigation (en haut)

Définition à la ligne 76 du fichier MainActivity.java.

7.3.3.7 CHAUD

```
final int com.lasalle.meeting.MainActivity.CHAUD = 3  [static], [private]
```

Constant niveau de confort CHAUD.

Définition à la ligne 65 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.filtreChaud().

7.3.3.8 communication

```
Communication com.lasalle.meeting.MainActivity.communication = null  [static], [private]
```

attribut permetant d'envoyer des requêtes

Définition à la ligne 86 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.getCommunication().

7.3.3.9 filtre

```
Spinner com.lasalle.meeting.MainActivity.filtre  [private]
```

la vue

Ressources layout activity_main

Définition à la ligne 70 du fichier MainActivity.java.

7.3.3.10 FRAIS

```
final int com.lasalle.meeting.MainActivity.FRAIS = -2  [static], [private]
```

Constant niveau de confort FRAIS.

Définition à la ligne 60 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.filtreFrais().

7.3.3.11 FROID

```
final int com.lasalle.meeting.MainActivity.FROID = -3  [static], [private]
```

Constant niveau de confort FROID.

Définition à la ligne 59 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.filtreFroid().

7.3.3.12 handlerUI

```
Handler com.lasalle.meeting.MainActivity.handlerUI  [private]
```

permet de récuperer les trames

Paramètres

Message	msg

Renvoie

void

Définition à la ligne 392 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.connecterWiFi().

7.3.3.13 layoutManager

```
RecyclerView.LayoutManager com.lasalle.meeting.MainActivity.layoutManager  [private]
```

le gestionnaire de mise en page

Définition à la ligne 75 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.initialiserSalles(), et com.lasalle.meeting.MainActivity.reInitialiserSalles().

7.3.3.14 LEGEREMENTFRAIS

```
final int com.lasalle.meeting.MainActivity.LEGEREMENTFRAIS = -1  [static], [private]
```

Constant niveau de confort LEGEREMENTFRAIS.

Définition à la ligne 61 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.filtreLegerementFrais().

7.3.3.15 LEGEREMENTTIEDE

```
final int com.lasalle.meeting.MainActivity.LEGEREMENTTIEDE = 1  [static], [private]
```

Constant niveau de confort LEGEREMENTTIEDE.

Définition à la ligne 63 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.filtreLegerementTiede().

7.3.3.16 mesSalles

```
Vector<Salle> com.lasalle.meeting.MainActivity.mesSalles  [static], [private]
```

Vecteur contenant mes salles (moyen de stockage)

Attributs

Définition à la ligne 84 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.getMesSalles().

7.3.3.17 NEUTRE

```
final int com.lasalle.meeting.MainActivity.NEUTRE = 0  [static], [private]
```

Constant niveau de confort NEUTRE.

Définition à la ligne 62 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.filtreNeutre().

7.3.3.18 recyclerView

```
RecyclerView com.lasalle.meeting.MainActivity.recyclerView  [private]
```

la vue des salles

Définition à la ligne 73 du fichier MainActivity.java.

7.3.3.19 salle

```
Salle com.lasalle.meeting.MainActivity.salle = null  [private]
```

attribut salle

Définition à la ligne 87 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.creerSalle().

7.3.3.20 sallesFilter

```
Vector<Salle> com.lasalle.meeting.MainActivity.sallesFilter  [static], [private]
```

Vecteur contenant mes salles aprés le filtre appliquer par l'utilisateur.

Définition à la ligne 85 du fichier MainActivity.java.

7.3.3.21 SEPARATEUR

```
final String com.lasalle.meeting.MainActivity.SEPARATEUR = ";"  [private]
```

séparateur utilisé dans le protocole meeting

Définition à la ligne 58 du fichier MainActivity.java.

7.3.3.22 swipeRefreshLayout

```
SwipeRefreshLayout com.lasalle.meeting.MainActivity.swipeRefreshLayout  [private]
```

layout permettant de rafraichir

Définition à la ligne 72 du fichier MainActivity.java.

7.3.3.23 TAG

```
final String com.lasalle.meeting.MainActivity.TAG = "MainActivity"  [static], [private]
```

TAG utilisé pour les logs.

Constantes

Définition à la ligne 57 du fichier MainActivity.java.

7.3.3.24 TIEDE

```
final int com.lasalle.meeting.MainActivity.TIEDE = 2  [static], [private]
```

Constant niveau de confort TIEDE.

Définition à la ligne 64 du fichier MainActivity.java.

Référencé par com.lasalle.meeting.MainActivity.filtreTiede().

7.3.3.25 wm

```
WifiManager com.lasalle.meeting.MainActivity.wm = null  [private]
```

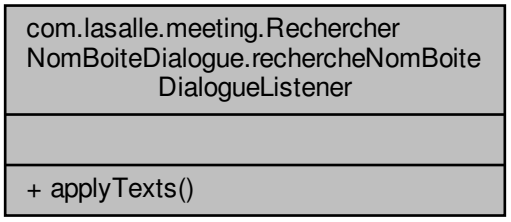
attribut permetant de voir la connection au wi-fi

Définition à la ligne 88 du fichier MainActivity.java.

La documentation de cette classe a été générée à partir du fichier suivant :
— MainActivity.java

7.4 Référence de l'interface com.lasalle.meeting.RechercherNomBoiteDialogue.rechercheNomBoiteDialogueListener

Graphe de collaboration de com.lasalle.meeting.RechercherNomBoiteDialogue.rechercheNomBoiteDialogueListener :



Fonctions membres publiques

— void applyTexts (String nomSalleRechercher)

7.4.1 Description détaillée

Définition à la ligne 63 du fichier RechercherNomBoiteDialogue.java.

7.4.2 Documentation des fonctions membres

7.4.2.1 applyTexts()

```
void com.lasalle.meeting.RechercherNomBoiteDialogue.rechercheNomBoiteDialogueListener.applyTexts (
            String nomSalleRechercher )
```

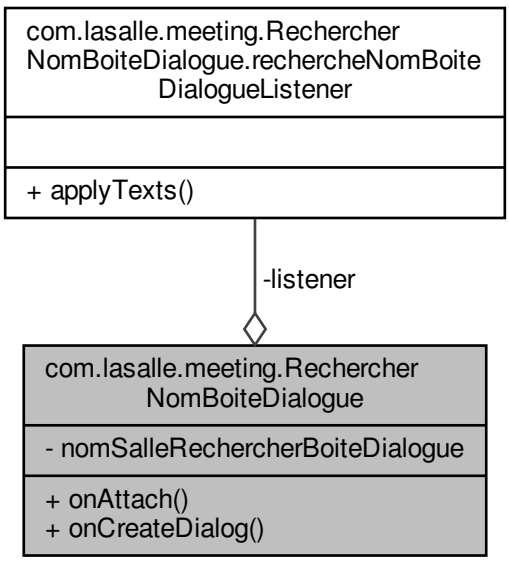
Implémenté dans com.lasalle.meeting.MainActivity.

Référencé par com.lasalle.meeting.RechercherNomBoiteDialogue.onCreateDialog().

La documentation de cette interface a été générée à partir du fichier suivant :
— RechercherNomBoiteDialogue.java

7.5 Référence de la classe com.lasalle.meeting.RechercherNomBoiteDialogue

Graphe de collaboration de com.lasalle.meeting.RechercherNomBoiteDialogue :



Classes

— interface rechercheNomBoiteDialogueListener

Fonctions membres publiques

— void onAttach (Context context)
— Dialog onCreateDialog (Bundle savedInstanceState)

Attributs privés

— rechercheNomBoiteDialogueListener listener
— EditText nomSalleRechercherBoiteDialogue

7.5.1 Description détaillée

Définition à la ligne 15 du fichier RechercherNomBoiteDialogue.java.

7.5.2 Documentation des fonctions membres

7.5.2.1 onAttach()

```
void com.lasalle.meeting.RechercherNomBoiteDialogue.onAttach (
             Context context )
```

Définition à la ligne 51 du fichier RechercherNomBoiteDialogue.java.

```
00051                                       {
00052         super.onAttach(context);
00053
00054         try
00055         {
00056             listener = (rechercheNomBoiteDialogueListener) context;
00057         }
00058         catch (ClassCastException e) {
00059             throw new ClassCastException(context.toString() + "doit implementer
      rechercheNomBoiteDiallogueListener");
00060         }
00061     }
```

7.5.2.2 onCreateDialog()

```
Dialog com.lasalle.meeting.RechercherNomBoiteDialogue.onCreateDialog (
            Bundle savedInstanceState )
```

Définition à la ligne 22 du fichier RechercherNomBoiteDialogue.java.

Références com.lasalle.meeting.RechercherNomBoiteDialogue.rechercheNomBoiteDialogueListener.applyTexts().

```
00023     {
00024         AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
00025 
00026         LayoutInflater inflater = getActivity().getLayoutInflater();
00027         View view = inflater.inflate(R.layout.dialog_nom, null);
00028 
00029         builder.setView(view)
00030                 .setTitle("Recherche par nom")
00031                 .setNegativeButton("Annulé", new DialogInterface.OnClickListener() {
00032                     @Override
00033                     public void onClick(DialogInterface dialog, int which) {
00034 
00035                     }
00036                 })
00037                 .setPositiveButton("ok", new DialogInterface.OnClickListener() {
00038                     @Override
00039                     public void onClick(DialogInterface dialog, int which) {
00040                     String nomSalleRechercher = nomSalleRechercherBoiteDialogue
      .getText().toString();
00041                     listener.applyTexts(nomSalleRechercher);
00042                     }
00043                 });
00044 
00045         nomSalleRechercherBoiteDialogue = view.findViewById(R.id.
      nomSalleRechercher);
00046 
00047         return builder.create();
00048     }
```

7.5.3 Documentation des données membres

7.5.3.1 listener

```
rechercheNomBoiteDialogueListener com.lasalle.meeting.RechercherNomBoiteDialogue.listener  [private]
```

Définition à la ligne 18 du fichier RechercherNomBoiteDialogue.java.

7.5.3.2 nomSalleRechercherBoiteDialogue

```
EditText com.lasalle.meeting.RechercherNomBoiteDialogue.nomSalleRechercherBoiteDialogue  [private]
```

Définition à la ligne 17 du fichier RechercherNomBoiteDialogue.java.

La documentation de cette classe a été générée à partir du fichier suivant :

— RechercherNomBoiteDialogue.java

7.6 Référence de la classe Runnable

Graphe de collaboration de Runnable :

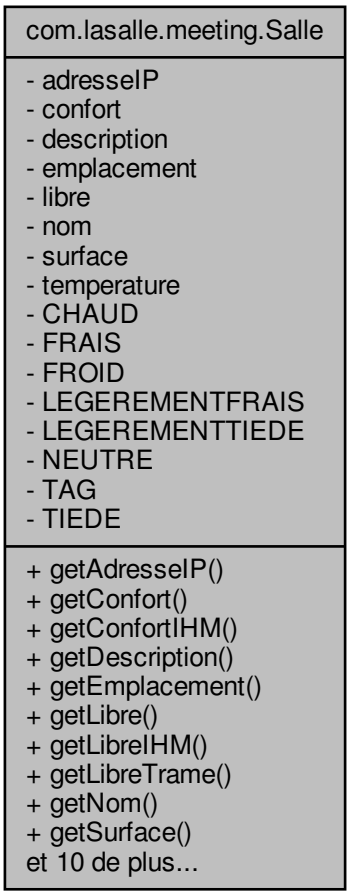
La documentation de cette classe a été générée à partir du fichier suivant :

— Communication.java

7.7 Référence de la classe com.lasalle.meeting.Salle

Déclaration de la classe Salle.

Graphe de collaboration de com.lasalle.meeting.Salle :



Fonctions membres publiques

- final String getAdresseIP ()
  Accesseur get l'adresse IP de la salle.
- final int getConfort ()
  Accesseur get le confort de la salle.
- final String getConfortIHM ()
  Accesseur get le confort de la salle.
- final String getDescription ()
  Accesseur get la description de la salle.
- final String getEmplacement ()
  Accesseur get de l'emplacement de la salle.
- final boolean getLibre ()
  Accesseur get de libre de la salle.
- final String getLibreIHM ()
  Accesseur get de libre de la salle.
- final String getLibreTrame ()
  Accesseur get de libre de la salle.
- final String getNom ()
  Accesseur get du nom de la salle.
- final int getSurface ()
  Accesseur get la surface de la salle.
- final float getTemperature ()
  Accesseur get la température de la salle.
- Salle (String nom, String description, String emplacement, int libre, int surface, int confort, float temperature, String adresseIP)

Constructeur de la classe Salle.
- void setAdresseIP (String adresseIP)
  Accesseur set l'adresse IP de la salle.
- void setConfort (int nouveauConfort)
  Accesseur set du confort de la salle.
- void setEmplacement (String nouvelleEmplacement)
  Accesseur set de l'emplacement de la salle.
- void setLibre (int libre)
  Accesseur set la disponibilité de la salle.
- void setLibre ()
  Accesseur set la disponibilité de la salle, change l'état de la salle.
- void setNom (String nouveauNom)
  Accesseur set du nom de la salle.
- void setSurface (int nouvelleSurface)
  Accesseur set la surface de la salle.
- void setTemperature (float temperature)
  Accesseur set la température de la salle.

Attributs privés

- String adresseIP
  l'adresse IP de la salle
- int confort
  le niveau de confort de la salle
- String description = ""
  La description de la salle.
- String emplacement = ""
  L'emplacement de la salle.
- boolean libre
  L'état booléen d'occupation Libre de la salle.
- String nom = ""
  Le nom de la salle.
- int surface
  la surface de la salle
- float temperature
  la température de la salle

Attributs privés statiques

- static final int CHAUD = 3
  Constant niveau de confort CHAUD.
- static final int FRAIS = -2
  Constant niveau de confort FRAIS.
- static final int FROID = -3
  Constant niveau de confort FROID.
- static final int LEGEREMENTFRAIS = -1
  Constant niveau de confort LEGEREMENTFRAIS.
- static final int LEGEREMENTTIEDE = 1
  Constant niveau de confort LEGEREMENTTIEDE.
- static final int NEUTRE = 0
  Constant niveau de confort NEUTRE.
- static final String TAG = "Salle"
  TAG utilisé pour les logs.
- static final int TIEDE = 2
  Constant niveau de confort TIEDE.

7.7.1 Description détaillée

Déclaration de la classe Salle.

Définition à la ligne 17 du fichier Salle.java.

7.7.2 Documentation des constructeurs et destructeur

7.7.2.1 Salle()

```
com.lasalle.meeting.Salle.Salle (
            String nom,
            String description,
            String emplacement,
            int libre,
            int surface,
            int confort,
            float temperature,
            String adresseIP )
```

Constructeur de la classe Salle.

Paramètres

nom	
description	
emplacement	
libre	
surface	
temperature	
adresseIP	

Définition à la ligne 54 du fichier Salle.java.

Références com.lasalle.meeting.Salle.adresseIP, com.lasalle.meeting.Salle.confort, com.lasalle.meeting.Salle.description, com.lasalle.meeting.Salle.emplacement, com.lasalle.meeting.Salle.nom, com.lasalle.meeting.Salle.setLibre(), com.lasalle.meeting.Salle.surface, et com.lasalle.meeting.Salle.temperature.

```
00055     {
00056         this.nom = nom;
00057         this.description = description;
00058         this.emplacement = emplacement;
00059         setLibre(libre);
00060         this.surface = surface;
00061         this.confort = confort;
00062         this.temperature = temperature;
00063         this.adresseIP = adresseIP;
00064         Log.d(TAG, "Salle : nom = " + nom + " - description = " +
      description + " - emplacement " + emplacement + " - libre = " +
      libre + " - surface = " + surface + " - confort = " + confort + " - température = " +
      temperature + " - adresseIP = " + adresseIP);
00065     }
```

7.7.3 Documentation des fonctions membres

7.7.3.1 getAdresseIP()

```
final String com.lasalle.meeting.Salle.getAdresseIP ( )
```

Accesseur get l'adresse IP de la salle.

Renvoie

String l'adresse IP de la salle

Définition à la ligne 285 du fichier Salle.java.

Références com.lasalle.meeting.Salle.adresseIP.

Référencé par com.lasalle.meeting.SalleActivity.setListener(), et com.lasalle.meeting.MainActivity.verifierExistenceSalle().

```
00286     {
00287         return adresseIP;
00288     }
```

7.7.3.2 getConfort()

```
final int com.lasalle.meeting.Salle.getConfort ( )
```

Accesseur get le confort de la salle.

Renvoie

int le niveau de confort de la salle

Définition à la ligne 224 du fichier Salle.java.

Références com.lasalle.meeting.Salle.confort.

Référencé par com.lasalle.meeting.MainActivity.verifierChangementSalle().

```
00225     {
00226         return confort;
00227     }
```

7.7.3.3 getConfortIHM()

```
final String com.lasalle.meeting.Salle.getConfortIHM ( )
```

Accesseur get le confort de la salle.

Renvoie

String le niveau de confort de la salle, pour l'affichage

Définition à la ligne 233 du fichier Salle.java.

Références com.lasalle.meeting.Salle.CHAUD, com.lasalle.meeting.Salle.FRAIS, com.lasalle.meeting.Salle.FROID, com.lasalle.↩ meeting.Salle.LEGEREMENTFRAIS, com.lasalle.meeting.Salle.LEGEREMENTTIEDE, com.lasalle.meeting.Salle.NEUTRE, et com.lasalle.meeting.Salle.TIEDE.

Référencé par com.lasalle.meeting.SalleViewHolder.afficher(), et com.lasalle.meeting.SalleActivity.afficherInformationSalle().

```
00234     {
00235         String message = "";
00236         switch (confort)
00237         {
00238             case FROID:
00239                 message = "Confort : Froid";
00240                 break;
00241             case FRAIS:
00242                 message = "Confort : Frais";
00243                 break;
00244             case LEGEREMENTFRAIS:
00245                 message = "Confort : Légèrement frais";
00246                 break;
00247             case NEUTRE:
00248                 message = "Confort : Neutre";
00249                 break;
00250             case LEGEREMENTTIEDE:
00251                 message = "Confort : Légèrement tiède";
00252                 break;
00253             case TIEDE:
00254                 message = "Confort : Tiède";
00255                 break;
00256             case CHAUD:
00257                 message = "Confort : Chaud";
00258                 break;
00259         }
00260         return message;
00261     }
```

7.7.3.4 getDescription()

```
final String com.lasalle.meeting.Salle.getDescription ( )
```

Accesseur get la description de la salle.

Renvoie

String la description de la salle

Définition à la ligne 276 du fichier Salle.java.

Références com.lasalle.meeting.Salle.description.

Référencé par com.lasalle.meeting.SalleActivity.afficherInformationSalle(), et com.lasalle.meeting.MainActivity.verifierChangement↩ Salle().

```
00277     {
00278         return description;
00279     }
```

7.7.3.5 getEmplacement()

```
final String com.lasalle.meeting.Salle.getEmplacement ( )
```

Accesseur get de l'emplacement de la salle.

Renvoie

String l'emplacement de la salle

Définition à la ligne 156 du fichier Salle.java.

Références com.lasalle.meeting.Salle.emplacement.

Référencé par com.lasalle.meeting.SalleActivity.afficherInformationSalle(), et com.lasalle.meeting.MainActivity.verifierChangementSalle().

```
00157     {
00158         return emplacement;
00159     }
```

7.7.3.6 getLibre()

```
final boolean com.lasalle.meeting.Salle.getLibre ( )
```

Accesseur get de libre de la salle.

Renvoie

boolean la disponibilité de la salle

Définition à la ligne 174 du fichier Salle.java.

Références com.lasalle.meeting.Salle.libre.

Référencé par com.lasalle.meeting.SalleViewHolder.afficher(), com.lasalle.meeting.SalleActivity.afficherInformationSalle(), com.lasalle.meeting.SalleActivity.setBoutonChangeEtat(), et com.lasalle.meeting.MainActivity.verifierChangementSalle().

```
00175     {
00176         return libre;
00177     }
```

7.7.3.7 getLibreIHM()

```
final String com.lasalle.meeting.Salle.getLibreIHM ( )
```

Accesseur get de libre de la salle.

Renvoie

String la disponibilité de la salle, pour l'affichage

Définition à la ligne 199 du fichier Salle.java.

Référencé par com.lasalle.meeting.SalleViewHolder.afficher().

```
00200     {
00201         if(libre == false)
00202         {
00203             return "occupée";
00204         }
00205         else
00206         {
00207             return "disponible";
00208         }
00209     }
```

7.7.3.8 getLibreTrame()

```
final String com.lasalle.meeting.Salle.getLibreTrame ( )
```

Accesseur get de libre de la salle.

Renvoie

String la disponibilité de la salle, pour la trame

Définition à la ligne 183 du fichier Salle.java.

Référencé par com.lasalle.meeting.SalleActivity.setListener().

```
00184     {
00185         if(libre == false)
00186         {
00187             return "0";
00188         }
00189         else
00190         {
00191             return "1";
00192         }
00193     }
```

7.7.3.9 getNom()

```
final String com.lasalle.meeting.Salle.getNom ( )
```

Accesseur get du nom de la salle.

Renvoie

String le nom de la salle

Définition à la ligne 165 du fichier Salle.java.

Références com.lasalle.meeting.Salle.nom.

Référencé par com.lasalle.meeting.SalleViewHolder.afficher(), com.lasalle.meeting.SalleActivity.afficherInformationSalle(), com.lasalle.meeting.SalleActivity.onCreate(), et com.lasalle.meeting.MainActivity.verifierChangementSalle().

```
00166     {
00167         return nom;
00168     }
```

7.7.3.10 getSurface()

```
final int com.lasalle.meeting.Salle.getSurface ( )
```

Accesseur get la surface de la salle.

Renvoie

int la surface de la salle

Définition à la ligne 215 du fichier Salle.java.

Références com.lasalle.meeting.Salle.surface.

Référencé par com.lasalle.meeting.SalleActivity.afficherInformationSalle(), et com.lasalle.meeting.MainActivity.verifierChangementSalle().

```
00216     {
00217         return surface;
00218     }
```

7.7.3.11 getTemperature()

```
final float com.lasalle.meeting.Salle.getTemperature ( )
```

Accesseur get la température de la salle.

Renvoie

int la température de la salle

Définition à la ligne 267 du fichier Salle.java.

Références com.lasalle.meeting.Salle.temperature.

Référencé par com.lasalle.meeting.SalleActivity.afficherInformationSalle(), et com.lasalle.meeting.MainActivity.verifierChangementSalle().

```
00268     {
00269         return temperature;
00270     }
```

7.7.3.12 setAdresseIP()

```
void com.lasalle.meeting.Salle.setAdresseIP (
             String adresseIP )
```

Accesseur set l'adresse IP de la salle.

Paramètres

adresseIP	la nouvelle adresse IP de la salle

Définition à la ligne 147 du fichier Salle.java.

Références com.lasalle.meeting.Salle.adresseIP.

```
00148     {
00149         this.adresseIP = adresseIP;
00150     }
```

7.7.3.13 setConfort()

```
void com.lasalle.meeting.Salle.setConfort (
             int nouveauConfort )
```

Accesseur set du confort de la salle.

Paramètres

nouveauConfort	le nouveau confort de la salle

Définition à la ligne 129 du fichier Salle.java.

```
00130     {
00131         this.confort = nouveauConfort;
00132     }
```

7.7.3.14 setEmplacement()

```
void com.lasalle.meeting.Salle.setEmplacement (
             String nouvelleEmplacement )
```

Accesseur set de l'emplacement de la salle.

Paramètres

nouvelleEmplacement	l'emplacement de la salle

Définition à la ligne 71 du fichier Salle.java.

```
00072     {
00073         emplacement = nouvelleEmplacement;
00074     }
```

7.7.3.15 setLibre() [1/2]

```
void com.lasalle.meeting.Salle.setLibre (
             int libre )
```

Accesseur set la disponibilité de la salle.

Paramètres

libre	le nouvelle état de la salle

Définition à la ligne 89 du fichier Salle.java.

Référencé par com.lasalle.meeting.SalleActivity.setListener().

```
00090     {
00091         if (libre == 0)
00092         {
00093             this.libre = false;
00094         }
00095         else
00096         {
00097             this.libre = true;
00098         }
00099     }
```

7.7.3.16 setLibre() [2/2]

```
void com.lasalle.meeting.Salle.setLibre ( )
```

Accesseur set la disponibilité de la salle, change l'état de la salle.

Définition à la ligne 104 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.Salle().

```
00105     {
00106         if (libre == true)
00107         {
00108             this.libre = false;
00109         }
00110         else
00111         {
00112             this.libre = true;
00113         }
00114     }
```

7.7.3.17 setNom()

```
void com.lasalle.meeting.Salle.setNom (
             String nouveauNom )
```

Accesseur set du nom de la salle.

Paramètres

nouveauNom	le nom de la salle

Définition à la ligne 80 du fichier Salle.java.

```
00081     {
00082         nom = nouveauNom;
00083     }
```

7.7.3.18 setSurface()

```
void com.lasalle.meeting.Salle.setSurface (
             int nouvelleSurface )
```

Accesseur set la surface de la salle.

Paramètres

nouvelleSurface	le nouvelle surface de la salle

Définition à la ligne 120 du fichier Salle.java.

```
00121     {
00122         this.surface = nouvelleSurface;
00123     }
```

7.7.3.19 setTemperature()

```
void com.lasalle.meeting.Salle.setTemperature (
            float temperature )
```

Accesseur set la température de la salle.

Paramètres

temperature	la nouvelle température de la salle

Définition à la ligne 138 du fichier Salle.java.

Références com.lasalle.meeting.Salle.temperature.

```
00139     {
00140         this.temperature = temperature;
00141     }
```

7.7.4 Documentation des données membres

7.7.4.1 adresseIP

```
String com.lasalle.meeting.Salle.adresseIP  [private]
```

l'adresse IP de la salle

Définition à la ligne 41 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getAdresseIP(), com.lasalle.meeting.Salle.Salle(), et com.lasalle.meeting.Salle.setAdresseIP().

7.7.4.2 CHAUD

```
final int com.lasalle.meeting.Salle.CHAUD = 3  [static], [private]
```

Constant niveau de confort CHAUD.

Définition à la ligne 29 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getConfortIHM().

7.7.4.3 confort

```
int com.lasalle.meeting.Salle.confort  [private]
```

le niveau de confort de la salle

Définition à la ligne 39 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getConfort(), et com.lasalle.meeting.Salle.Salle().

7.7.4.4 description

```
String com.lasalle.meeting.Salle.description = ""  [private]
```

La description de la salle.

Définition à la ligne 35 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getDescription(), et com.lasalle.meeting.Salle.Salle().

7.7.4.5 emplacement

```
String com.lasalle.meeting.Salle.emplacement = ""  [private]
```

L'emplacement de la salle.

Définition à la ligne 36 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getEmplacement(), et com.lasalle.meeting.Salle.Salle().

7.7.4.6 FRAIS

```
final int com.lasalle.meeting.Salle.FRAIS = -2  [static], [private]
```

Constant niveau de confort FRAIS.

Définition à la ligne 24 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getConfortIHM().

7.7.4.7 FROID

```
final int com.lasalle.meeting.Salle.FROID = -3  [static], [private]
```

Constant niveau de confort FROID.

Définition à la ligne 23 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getConfortIHM().

7.7.4.8 LEGEREMENTFRAIS

```
final int com.lasalle.meeting.Salle.LEGEREMENTFRAIS = -1  [static], [private]
```

Constant niveau de confort LEGEREMENTFRAIS.

Définition à la ligne 25 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getConfortIHM().

7.7.4.9 LEGEREMENTTIEDE

```
final int com.lasalle.meeting.Salle.LEGEREMENTTIEDE = 1  [static], [private]
```

Constant niveau de confort LEGEREMENTTIEDE.

Définition à la ligne 27 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getConfortIHM().

7.7.4.10 libre

```
boolean com.lasalle.meeting.Salle.libre  [private]
```

L'état booléen d'occupation Libre de la salle.

Définition à la ligne 37 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getLibre().

7.7.4.11 NEUTRE

```
final int com.lasalle.meeting.Salle.NEUTRE = 0  [static], [private]
```

Constant niveau de confort NEUTRE.

Définition à la ligne 26 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getConfortIHM().

7.7.4.12 nom

```
String com.lasalle.meeting.Salle.nom = ""  [private]
```

Le nom de la salle.

Attributs

Définition à la ligne 34 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getNom(), et com.lasalle.meeting.Salle.Salle().

7.7.4.13 surface

```
int com.lasalle.meeting.Salle.surface  [private]
```

la surface de la salle

Définition à la ligne 38 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getSurface(), et com.lasalle.meeting.Salle.Salle().

7.7.4.14 TAG

```
final String com.lasalle.meeting.Salle.TAG = "Salle"  [static], [private]
```

TAG utilisé pour les logs.

Constantes

Définition à la ligne 22 du fichier Salle.java.

7.7.4.15 temperature

```
float com.lasalle.meeting.Salle.temperature  [private]
```

la température de la salle

Définition à la ligne 40 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getTemperature(), com.lasalle.meeting.Salle.Salle(), et com.lasalle.meeting.Salle.set↩ Temperature().

7.7.4.16 TIEDE

```
final int com.lasalle.meeting.Salle.TIEDE = 2  [static], [private]
```

Constant niveau de confort TIEDE.

Définition à la ligne 28 du fichier Salle.java.

Référencé par com.lasalle.meeting.Salle.getConfortIHM().

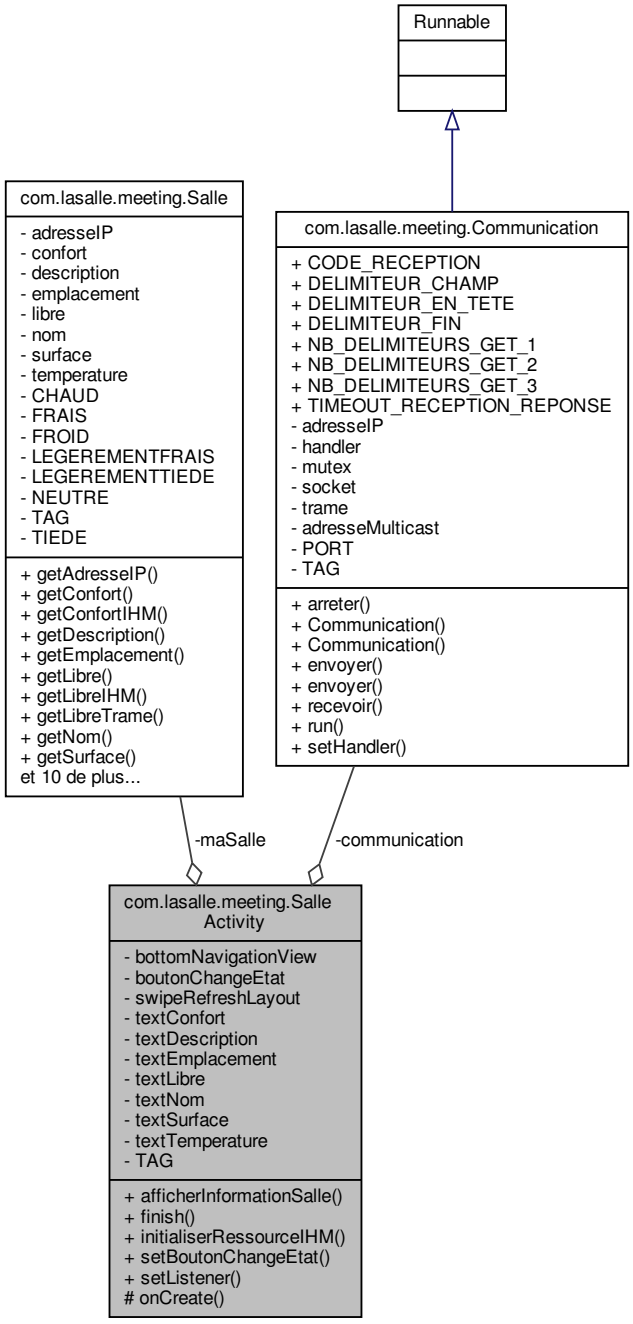
La documentation de cette classe a été générée à partir du fichier suivant :

— Salle.java

7.8 Référence de la classe com.lasalle.meeting.SalleActivity

Déclaration de la classe SalleActivity.

Graphe de collaboration de com.lasalle.meeting.SalleActivity :



Fonctions membres publiques

- void afficherInformationSalle ()
  Méthode affichant les informations de la salle dans les layouts.
- void finish ()

Méthode appelée à la fin de l'activité SalleActivity.

— void initialiserRessourceIHM ()
Récupère et initialise les widgets du layout activity_salle.

— void setBoutonChangeEtat ()
Méthode changeant le bouton dépendant de la disponibilité de la salle.

— void setListener ()
applique les listener sur les layouts approprié

Fonctions membres protégées

— void onCreate (Bundle savedInstanceState)
Méthode appelée à la création de l'activité SalleActivity.

Attributs privés

— BottomNavigationView bottomNavigationView
layout permettant d'avoir un menu de navigation (en haut)

— Button boutonChangeEtat
layout du prendre/liberer

— Communication communication = null
attribut permetant d'envoyer des requêtes

— Salle maSalle = null
attribut salle

— SwipeRefreshLayout swipeRefreshLayout
layout permettant de rafraichir

— TextView textConfort
layout texte du confort de la salle

— TextView textDescription
layout texte de la description de la salle

— TextView textEmplacement
layout texte de l'emplacement de la salle

— TextView textLibre
layout texte de la disponibilité de la salle

— TextView textNom
layout texte du nom de la salle

— TextView textSurface
layout texte de la surface de la salle

— TextView textTemperature
layout texte de la température de la salle

Attributs privés statiques

— static final String TAG = "SalleActivity"
TAG utilisé pour les logs.

7.8.1 Description détaillée

Déclaration de la classe SalleActivity.

Définition à la ligne 29 du fichier SalleActivity.java.

7.8.2 Documentation des fonctions membres

7.8.2.1 afficherInformationSalle()

```
void com.lasalle.meeting.SalleActivity.afficherInformationSalle ( )
```

Méthode affichant les informations de la salle dans les layouts.

Renvoie

void

Définition à la ligne 103 du fichier SalleActivity.java.

Références com.lasalle.meeting.Salle.getConfortIHM(), com.lasalle.meeting.Salle.getDescription(), com.lasalle.meeting.Salle.getEmplacement(), com.lasalle.meeting.Salle.getLibre(), com.lasalle.meeting.Salle.getNom(), com.lasalle.meeting.Salle.getSurface(), et com.lasalle.meeting.Salle.getTemperature().

Référencé par com.lasalle.meeting.SalleActivity.onCreate(), et com.lasalle.meeting.SalleActivity.setListener().

```
00104     {
00105         textNom.setText(maSalle.getNom());
00106         textNom.setTextSize(35);
00107         textDescription.setText(maSalle.getDescription());
00108         textDescription.setTextSize(25);
00109         textEmplacement.setText(maSalle.getEmplacement());
00110         textEmplacement.setTextSize(25);
00111         textConfort.setText(maSalle.getConfortIHM());
00112         textConfort.setTextSize(25);
00113         textSurface.setText(Integer.toString(maSalle.
      getSurface()) + " m²");
00114         textSurface.setTextSize(25);
00115         textLibre.setTextSize(25);
00116         if (maSalle.getLibre() == true)
00117         {
00118             textLibre.setText("État : Libre");
00119         }
00120         else
00121         {
00122             textLibre.setText("État : Occupée");
00123         }
00124         textTemperature.setText(Float.toString(maSalle.
      getTemperature()) + " °C");
00125         textTemperature.setTextSize(25);
00126     }
```

7.8.2.2 finish()

```
void com.lasalle.meeting.SalleActivity.finish ( )
```

Méthode appelée à la fin de l'activité SalleActivity.

Renvoie

void

Définition à la ligne 208 du fichier SalleActivity.java.

```
00209     {
00210         Log.d(TAG, "finish()");
00211
00212         Intent intent = new Intent();
00213
00214         intent.putExtra("salle", maSalle);
00215
00216         setResult(RESULT_OK, intent);
00217         super.finish();
00218     }
```

7.8.2.3 initialiserRessourceIHM()

```
void com.lasalle.meeting.SalleActivity.initialiserRessourceIHM ( )
```

Récupère et initialise les widgets du layout activity_salle.

Renvoie

void

Définition à la ligne 189 du fichier SalleActivity.java.

Référencé par com.lasalle.meeting.SalleActivity.onCreate().

```
00190     {
00191         boutonChangeEtat = (Button)findViewById(R.id.buttonChangeEtat);
00192         textNom = (TextView)findViewById(R.id.textViewNom);
00193         textEmplacement = (TextView)findViewById(R.id.textViewEmplacement);
00194         textLibre = (TextView)findViewById(R.id.textViewLibre);
00195         textConfort = (TextView)findViewById(R.id.textViewConfort);
00196         textSurface = (TextView)findViewById(R.id.textViewSurface);
00197         textTemperature = (TextView)findViewById(R.id.textViewTemperature);
00198         textDescription = (TextView)findViewById(R.id.textViewDescription);
00199         swipeRefreshLayout = (SwipeRefreshLayout)findViewById(R.id.swipeRefreshLayout);
00200         bottomNavigationView = (BottomNavigationView) findViewById(R.id.
      bottomNavigationView);
00201     }
```

7.8.2.4 onCreate()

```
void com.lasalle.meeting.SalleActivity.onCreate (
             Bundle savedInstanceState )  [protected]
```

Méthode appelée à la création de l'activité SalleActivity.

Paramètres

savedInstanceState	

Renvoie

void

Définition à la ligne 60 du fichier SalleActivity.java.

Références com.lasalle.meeting.SalleActivity.afficherInformationSalle(), com.lasalle.meeting.MainActivity.getCommunication(), com.lasalle.meeting.Salle.getNom(), com.lasalle.meeting.SalleActivity.initialiserRessourceIHM(), com.lasalle.meeting.SalleActivity.setBoutonChangeEtat(), et com.lasalle.meeting.SalleActivity.setListener().

```
00061     {
00062         super.onCreate(savedInstanceState);
00063         Log.d(TAG, "onCreate()");
00064
00065         setContentView(R.layout.activity_salle);
00066
00067         Intent intent = getIntent();
00068         maSalle = (Salle)intent.getSerializableExtra("Salle");
00069
00070         if(maSalle == null)
00071             Log.d(TAG, "Salle : " + maSalle.getNom());
00072
00073         communication = MainActivity.getCommunication();
```

```
00074
00075         initialiserRessourceIHM();
00076         setBoutonChangeEtat();
00077         afficherInformationSalle();
00078         setListener();
00079     }
```

7.8.2.5 setBoutonChangeEtat()

```
void com.lasalle.meeting.SalleActivity.setBoutonChangeEtat ( )
```

Méthode changeant le bouton dépendant de la disponibilité de la salle.

Renvoie

void

Définition à la ligne 85 du fichier SalleActivity.java.

Références com.lasalle.meeting.Salle.getLibre().

Référencé par com.lasalle.meeting.SalleActivity.onCreate(), et com.lasalle.meeting.SalleActivity.setListener().

```
00086     {
00087         if (maSalle.getLibre() == true)
00088         {
00089             boutonChangeEtat.setText("Prendre");
00090             boutonChangeEtat.setBackgroundColor(Color.rgb(39,195,26));
00091         }
00092         else
00093         {
00094             boutonChangeEtat.setText("Libérer");
00095             boutonChangeEtat.setBackgroundColor(Color.rgb(222,55,25));
00096         }
00097     }
```

7.8.2.6 setListener()

```
void com.lasalle.meeting.SalleActivity.setListener ( )
```

applique les listener sur les layouts approprié

Renvoie

void

Définition à la ligne 132 du fichier SalleActivity.java.

Références com.lasalle.meeting.SalleActivity.afficherInformationSalle(), com.lasalle.meeting.Communication.envoyer(), com.↩ lasalle.meeting.Salle.getAdresseIP(), com.lasalle.meeting.Salle.getLibreTrame(), com.lasalle.meeting.SalleActivity.setBouton↩ ChangeEtat(), et com.lasalle.meeting.Salle.setLibre().

Référencé par com.lasalle.meeting.SalleActivity.onCreate().

```
00133     {
00134
00135         boutonChangeEtat.setOnClickListener(
00136                 new View.OnClickListener()
00137                 {
00138                     @Override
00139                     public void onClick(View v)
00140                     {
00141                         maSalle.setLibre();
00142                         setBoutonChangeEtat();
00143                         afficherInformationSalle();
00144
00145                         if(communication != null)
00146                         {
00147                             communication.envoyer("$SET;3;" +
      maSalle.getLibreTrame() + "\r\n", maSalle.
      getAdresseIP());
00148                         }
00149                     }
00150                 }
00151         );
00152
00153         swipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener(
      )
00154         {
00155             @Override
00156             public void onRefresh()
00157             {
00158                 afficherInformationSalle();
00159                 swipeRefreshLayout.setRefreshing(false);
00160             }
00161         });
00162
00163         bottomNavigationView.setOnNavigationItemSelectedListener(new
      BottomNavigationView.OnNavigationItemSelectedListener()
00164         {
00165             @Override
00166             public boolean onNavigationItemSelected(@NonNull MenuItem item)
00167             {
00168                 switch (item.getItemId())
00169                 {
00170                     case R.id.Salle:
00171                         Toast.makeText(getApplicationContext(), "Salle", Toast.LENGTH_SHORT).show();
00172                         return true;
00173                     case R.id.Favoris:
00174                         Toast.makeText(getApplicationContext(), "La fonctionnalité favoris n'est pas encore
       disponible !", Toast.LENGTH_SHORT).show();
00175                         return true;
00176                     case R.id.Rechercher:
00177                         Toast.makeText(getApplicationContext(), "La fonctionnalité rechercher n'est pas
       encore disponible !", Toast.LENGTH_SHORT).show();
00178                         return true;
00179                 }
00180                 return false;
00181             }
00182         });
00183     }
```

7.8.3 Documentation des données membres

7.8.3.1 bottomNavigationView

`BottomNavigationView com.lasalle.meeting.SalleActivity.bottomNavigationView [private]`

layout permettant d'avoir un menu de navigation (en haut)

Définition à la ligne 47 du fichier SalleActivity.java.

7.8.3.2 boutonChangeEtat

`Button com.lasalle.meeting.SalleActivity.boutonChangeEtat [private]`

layout du prendre/liberer

Ressources layout activity_main

Définition à la ligne 38 du fichier SalleActivity.java.

7.8.3.3 communication

```
Communication com.lasalle.meeting.SalleActivity.communication = null  [private]
```

attribut permetant d'envoyer des requêtes

Définition à la ligne 52 du fichier SalleActivity.java.

7.8.3.4 maSalle

```
Salle com.lasalle.meeting.SalleActivity.maSalle = null  [private]
```

attribut salle

Attributs

Définition à la ligne 51 du fichier SalleActivity.java.

7.8.3.5 swipeRefreshLayout

```
SwipeRefreshLayout com.lasalle.meeting.SalleActivity.swipeRefreshLayout  [private]
```

layout permettant de rafraichir

Définition à la ligne 46 du fichier SalleActivity.java.

7.8.3.6 TAG

```
final String com.lasalle.meeting.SalleActivity.TAG = "SalleActivity"  [static], [private]
```

TAG utilisé pour les logs.

Constantes

Définition à la ligne 34 du fichier SalleActivity.java.

7.8.3.7 textConfort

```
TextView com.lasalle.meeting.SalleActivity.textConfort  [private]
```

layout texte du confort de la salle

Définition à la ligne 42 du fichier SalleActivity.java.

7.8.3.8 textDescription

```
TextView com.lasalle.meeting.SalleActivity.textDescription  [private]
```

layout texte de la description de la salle

Définition à la ligne 45 du fichier SalleActivity.java.

7.8.3.9 textEmplacement

```
TextView com.lasalle.meeting.SalleActivity.textEmplacement  [private]
```

layout texte de l'emplacement de la salle

Définition à la ligne 40 du fichier SalleActivity.java.

7.8.3.10 textLibre

```
TextView com.lasalle.meeting.SalleActivity.textLibre  [private]
```

layout texte de la disponibilité de la salle

Définition à la ligne 41 du fichier SalleActivity.java.

7.8.3.11 textNom

```
TextView com.lasalle.meeting.SalleActivity.textNom  [private]
```

layout texte du nom de la salle

Définition à la ligne 39 du fichier SalleActivity.java.

7.8.3.12 textSurface

```
TextView com.lasalle.meeting.SalleActivity.textSurface  [private]
```

layout texte de la surface de la salle

Définition à la ligne 43 du fichier SalleActivity.java.

7.8.3.13 textTemperature

```
TextView com.lasalle.meeting.SalleActivity.textTemperature  [private]
```

layout texte de la température de la salle

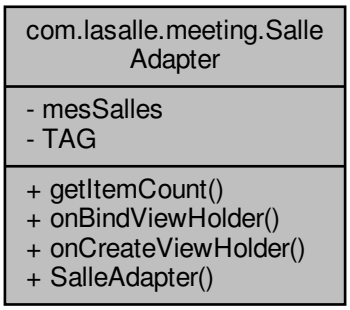
Définition à la ligne 44 du fichier SalleActivity.java.

La documentation de cette classe a été générée à partir du fichier suivant :
— SalleActivity.java

7.9 Référence de la classe com.lasalle.meeting.SalleAdapter

Déclaration de la classe SalleAdapter.

Graphe de collaboration de com.lasalle.meeting.SalleAdapter :



Fonctions membres publiques

- int getItemCount ()
  Méthode appelée à la création de l'activité SalleAdapter.
- void onBindViewHolder (@NonNull SalleViewHolder holder, int position)
  Méthode appelée à la création de l'activité SalleAdapter.
- SalleViewHolder onCreateViewHolder (@NonNull ViewGroup parent, int viewType)
  Méthode appelée à la création de l'activité SalleAdapter.
- SalleAdapter (Vector< Salle > mesSalles)
  constructeur de SalleAdapter

Attributs privés

- Vector< Salle > mesSalles = null
  Vecteur contenant mes salles.

Attributs privés statiques

- static final String TAG = "SalleAdapter"
  TAG utilisé pour les logs.

7.9.1 Description détaillée

Déclaration de la classe SalleAdapter.

Définition à la ligne 23 du fichier SalleAdapter.java.

7.9.2 Documentation des constructeurs et destructeur

7.9.2.1 SalleAdapter()

```
com.lasalle.meeting.SalleAdapter.SalleAdapter (
            Vector< Salle > mesSalles )
```

constructeur de SalleAdapter

Paramètres

mesSalles	Vector<Salle>

Définition à la ligne 38 du fichier SalleAdapter.java.

Références com.lasalle.meeting.SalleAdapter.mesSalles.

```
00039     {
00040         Log.d(TAG, "SalleAdapter (Vector<Salle>)");
00041
00042         if (mesSalles != null) {
00043             this.mesSalles = mesSalles;
00044         }
00045     }
```

7.9.3 Documentation des fonctions membres

7.9.3.1 getItemCount()

```
int com.lasalle.meeting.SalleAdapter.getItemCount ( )
```

Méthode appelée à la création de l'activité SalleAdapter.

Renvoie

int

Définition à la ligne 80 du fichier SalleAdapter.java.

```
00080                               {
00081         Log.d(TAG, "getItemCount()");
00082         if (mesSalles != null)
00083             return mesSalles.size();
00084         return 0;
00085     }
```

7.9.3.2 onBindViewHolder()

```
void com.lasalle.meeting.SalleAdapter.onBindViewHolder (
             @NonNull SalleViewHolder holder,
             int position )
```

Méthode appelée à la création de l'activité SalleAdapter.

Paramètres

holder	SalleViewHolder, position int

Renvoie

void

Définition à la ligne 68 du fichier SalleAdapter.java.

```
00069     {
00070         Log.d(TAG, "onBindViewHolder()");
00071         Salle salle = mesSalles.get(position);
00072         holder.afficher(salle);
00073     }
```

7.9.3.3 onCreateViewHolder()

```
SalleViewHolder com.lasalle.meeting.SalleAdapter.onCreateViewHolder (
             @NonNull ViewGroup parent,
             int viewType )
```

Méthode appelée à la création de l'activité SalleAdapter.

Paramètres

parent	ViewGroup, viewType int

Renvoie

SalleViewHolder

Définition à la ligne 54 du fichier SalleAdapter.java.

```
00055     {
00056         Log.d(TAG, "SalleViewHolder onCreateViewHolder()");
00057         LayoutInflater inflater = LayoutInflater.from(parent.getContext());
00058         View view = inflater.inflate(R.layout.salle, parent, false);
00059         return new SalleViewHolder(view);
00060     }
```

7.9.4 Documentation des données membres

7.9.4.1 mesSalles

```
Vector<Salle> com.lasalle.meeting.SalleAdapter.mesSalles = null  [private]
```

Vecteur contenant mes salles.

Attributs

Définition à la ligne 32 du fichier SalleAdapter.java.

Référencé par com.lasalle.meeting.SalleAdapter.SalleAdapter().

7.9.4.2 TAG

```
final String com.lasalle.meeting.SalleAdapter.TAG = "SalleAdapter"  [static], [private]
```

TAG utilisé pour les logs.

Constantes

Définition à la ligne 28 du fichier SalleAdapter.java.

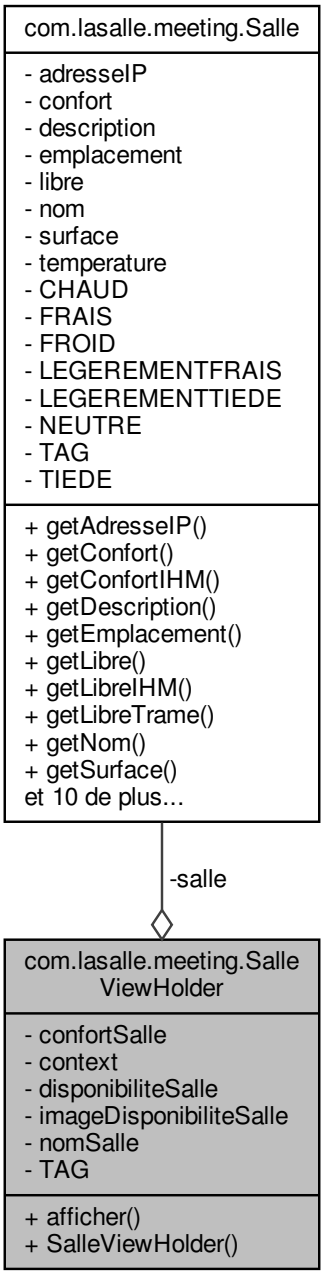
La documentation de cette classe a été générée à partir du fichier suivant :

— SalleAdapter.java

7.10 Référence de la classe com.lasalle.meeting.SalleViewHolder

Déclaration de la classe SalleViewHolder.

Graphe de collaboration de com.lasalle.meeting.SalleViewHolder :



Fonctions membres publiques

- void afficher (Salle salle)
  Méthode affichant les informations de la salle dans les layouts.
- SalleViewHolder (final View itemView)
  constructeur de SalleViewHolder

Attributs privés

- TextView confortSalle

layout texte du confort de la salle
— Context context
attribut permettant de communiquer avec une autre classe
— TextView disponibiliteSalle
layout texte de la disponibilité de la salle
— ImageView imageDisponibiliteSalle
layout image de la disponibilité de la salle
— TextView nomSalle
layout texte du nom de la salle
— Salle salle
attribut salle

Attributs privés statiques

— static final String TAG = "SalleViewHolder"
TAG utilisé pour les logs.

7.10.1 Description détaillée

Déclaration de la classe SalleViewHolder.

Définition à la ligne 25 du fichier SalleViewHolder.java.

7.10.2 Documentation des constructeurs et destructeur

7.10.2.1 SalleViewHolder()

```
com.lasalle.meeting.SalleViewHolder.SalleViewHolder (
            final View itemView )
```

constructeur de SalleViewHolder

Paramètres

itemView	final View

Définition à la ligne 49 du fichier SalleViewHolder.java.

```
00050     {
00051         super(itemView);
00052
00053         Log.d(TAG, "SalleViewHolder(final View itemView)");
00054
00055         nomSalle= ((TextView)itemView.findViewById(R.id.nomSalle));
00056         confortSalle = ((TextView)itemView.findViewById(R.id.confortSalle));
00057         disponibiliteSalle = ((TextView)itemView.findViewById(R.id.disponibiliteSalle));
00058         imageDisponibiliteSalle = ((ImageView)itemView.findViewById(R.id.
      imageDisponibiliteSalle));
00059         context = itemView.getContext();
00060
00061         itemView.setOnClickListener(new View.OnClickListener()
00062         {
00063             @Override
00064             public void onClick(View view)
00065             {
00066                 Intent intent = new Intent(context, SalleActivity.class);
00067                 intent.putExtra("Salle", (Serializable) salle);
00068                 context.startActivity(intent);
00069             }
00070         });
00071     }
```

7.10.3 Documentation des fonctions membres

7.10.3.1 afficher()

```
void com.lasalle.meeting.SalleViewHolder.afficher (
            Salle salle )
```

Méthode affichant les informations de la salle dans les layouts.

Renvoie

void

Définition à la ligne 77 du fichier SalleViewHolder.java.

Références com.lasalle.meeting.Salle.getConfortIHM(), com.lasalle.meeting.Salle.getLibre(), com.lasalle.meeting.Salle.getLibreIHM(), com.lasalle.meeting.Salle.getNom(), et com.lasalle.meeting.SalleViewHolder.salle.

```
00078     {
00079         Log.d(TAG, "afficher ()");
00080
00081         this.salle= salle;
00082         nomSalle.setText(salle.getNom());
00083         nomSalle.setTextSize(15);
00084         confortSalle.setText(salle.getConfortIHM());
00085         confortSalle.setTextSize(15);
00086         disponibiliteSalle.setText("La salle est "+ salle.
      getLibreIHM());
00087         disponibiliteSalle.setTextSize(15);
00088
00089         if(salle.getLibre())
00090         {
00091             imageDisponibiliteSalle.setImageResource(R.drawable.rond_vert);
00092         }
00093         else
00094         {
00095             imageDisponibiliteSalle.setImageResource(R.drawable.rond_rouge);
00096         }
00097     }
```

7.10.4 Documentation des données membres

7.10.4.1 confortSalle

```
TextView com.lasalle.meeting.SalleViewHolder.confortSalle  [private]
```

layout texte du confort de la salle

Définition à la ligne 35 du fichier SalleViewHolder.java.

7.10.4.2 context

```
Context com.lasalle.meeting.SalleViewHolder.context  [private]
```

attribut permettant de communiquer avec une autre classe

Attributs

Définition à la ligne 41 du fichier SalleViewHolder.java.

7.10.4.3 disponibiliteSalle

```
TextView com.lasalle.meeting.SalleViewHolder.disponibiliteSalle  [private]
```

layout texte de la disponibilité de la salle

Définition à la ligne 36 du fichier SalleViewHolder.java.

7.10.4.4 imageDisponibiliteSalle

```
ImageView com.lasalle.meeting.SalleViewHolder.imageDisponibiliteSalle  [private]
```

layout image de la disponibilité de la salle

Définition à la ligne 37 du fichier SalleViewHolder.java.

7.10.4.5 nomSalle

```
TextView com.lasalle.meeting.SalleViewHolder.nomSalle  [private]
```

layout texte du nom de la salle

Ressources layout activity_main

Définition à la ligne 34 du fichier SalleViewHolder.java.

7.10.4.6 salle

```
Salle com.lasalle.meeting.SalleViewHolder.salle  [private]
```

attribut salle

Définition à la ligne 42 du fichier SalleViewHolder.java.

Référencé par com.lasalle.meeting.SalleViewHolder.afficher().

7.10.4.7 TAG

```
final String com.lasalle.meeting.SalleViewHolder.TAG = "SalleViewHolder"  [static], [private]
```

TAG utilisé pour les logs.

Constantes

Définition à la ligne 30 du fichier SalleViewHolder.java.

La documentation de cette classe a été générée à partir du fichier suivant :
— SalleViewHolder.java

8 Documentation des fichiers

8.1 Référence du fichier Changelog.md

8.2 Changelog.md

```
00001 \page page_changelog Changelog
00002
00003 r1 | www-data | 2020-02-01 15:03:29 +0100 (sam. 01 févr. 2020) | 1 ligne
00004
00005 Creating initial repository structure
```

8.3 Référence du fichier Communication.java

Déclaration de la classe Communication.

Classes

— class com.lasalle.meeting.Communication
Déclaration de la classe Communication.

Paquetages

— package com.lasalle.meeting

8.3.1 Description détaillée

Déclaration de la classe Communication.

Auteur

Vincent DEVINE

Définition dans le fichier Communication.java.

8.4 Communication.java

```
00001 package com.lasalle.meeting;
00002
00003 import android.os.Bundle;
00004 import android.os.Handler;
00005 import android.os.Message;
00006 import android.util.Log;
00007 import java.net.*;
00008 import java.io.IOException;
00009 import java.net.DatagramPacket;
00010 import java.net.Socket;
00011 import java.util.concurrent.locks.ReentrantLock;
00012
00023 public class Communication implements Runnable
00024 {
00028     private final static String TAG = "Communication";
00029     private final static String adresseMulticast = "239.0.0.42";
00030     private final static int PORT = 5000;
00031     public static final int TIMEOUT_RECEPTION_REPONSE = 30000;
00032     public final static int CODE_RECEPTION = 1;
00033     private final ReentrantLock mutex = new ReentrantLock();
00037     private DatagramSocket socket;
00038     private InetAddress adresseIP = null;
00039     private Handler handler;
```

```
00040     private String trame;
00041
00044     public static final String DELIMITEUR_EN_TETE = "$";
00045     public static final String DELIMITEUR_CHAMP = ";";
00046     public static final String DELIMITEUR_FIN = "\r\n";
00047     public static final int NB_DELIMITEURS_GET_1 = 6; // $GET;1\r\n
00048     public static final int NB_DELIMITEURS_GET_2 = 3; // $GET;2\r\n
00049     public static final int NB_DELIMITEURS_GET_3 = 1; // $GET;3\r\n
00050
00056     public Communication(Handler handler)
00057     {
00058         this.handler = handler;
00059         try
00060         {
00061             socket = new DatagramSocket(PORT);
00062             socket.setSoTimeout(Communication.
      TIMEOUT_RECEPTION_REPONSE);
00063         }
00064         catch (SocketException se)
00065         {
00066             se.printStackTrace();
00067         }
00068
00069         try
00070         {
00071             this.adresseIP = InetAddress.getByName(adresseMulticast);
00072         }
00073         catch (UnknownHostException e)
00074         {
00075             e.printStackTrace();
00076         }
00077     }
00078
00084     public Communication(int port, Handler handler)
00085     {
00086         this.handler = handler;
00087         try
00088         {
00089             socket = new DatagramSocket(port);
00090             socket.setSoTimeout(Communication.
      TIMEOUT_RECEPTION_REPONSE);
00091         }
00092         catch (SocketException se)
00093         {
00094             se.printStackTrace();
00095         }
00096
00097         try
00098         {
00099             this.adresseIP = InetAddress.getByName(adresseMulticast);
00100         }
00101         catch (UnknownHostException e)
00102         {
00103             e.printStackTrace();
00104         }
00105     }
00106
00112     public void setHandler(Handler handler)
00113     {
00114         mutex.lock();
00115         this.handler = handler;
00116         mutex.unlock();
00117     }
00118
00123     public void recevoir()
00124     {
00125         byte[] reception = new byte[1024];
00126
00127         while (socket != null && !socket.isClosed())
00128         {
00129             try
00130             {
00131                 final DatagramPacket paquetRecu = new DatagramPacket(reception, reception.length);
00132                 socket.receive(paquetRecu);
00133
00134                 trame = new String(paquetRecu.getData(), paquetRecu.getOffset(), paquetRecu.getLength());
00135                 Log.d(TAG, "Réception de " + paquetRecu.getAddress().getHostAddress() + ":" + paquetRecu.
      getPort() + " -> " + trame);
00136
00137                 Message msg = Message.obtain();
00138                 Bundle b = new Bundle();
00139                 b.putString("adresseIP", paquetRecu.getAddress().getHostAddress());
00140                 b.putInt("port", paquetRecu.getPort());
00141                 b.putInt("etat", Communication.CODE_RECEPTION);
00142                 b.putString("trame", trame);
00143                 msg.setData(b);
00144                 mutex.lock();
00145                 handler.sendMessage(msg);
00146                 mutex.unlock();
00147             }
00148             catch (Exception e)
```

```
00149             {
00150                 Log.d(TAG, "Erreur recevoir() [socket.isClosed = " + socket.isClosed() + "]");
00151                 e.printStackTrace();
00152             }
00153         }
00154     }
00155
00161     public void envoyer(final String requete)
00162     {
00163         if(socket == null)
00164             return;
00165
00166         new Thread()
00167         {
00168             @Override public void run()
00169             {
00170                 byte[] emission = new byte[1024];
00171
00172                 try
00173                 {
00174                     emission = requete.getBytes();
00175                     DatagramPacket paquetRetour = new DatagramPacket(emission, emission.length, adresseIP,
      PORT);
00176                     socket.send(paquetRetour);
00177                     Log.d(TAG, "send() = " + requete);
00178                 }
00179                 catch (IOException e)
00180                 {
00181                     Log.d(TAG, "Erreur send() [socket.isClosed = " + socket.isClosed() + "]");
00182                     e.printStackTrace();
00183                 }
00184             }
00185         }.start();
00186     }
00187
00193     public void envoyer(final String requete, final String adresseIP)
00194     {
00195         if(socket == null)
00196             return;
00197
00198         final InetAddress adresseIPDistante;
00199         try
00200         {
00201             adresseIPDistante = InetAddress.getByName(adresseIP);
00202         }
00203         catch (UnknownHostException e)
00204         {
00205             e.printStackTrace();
00206             return;
00207         }
00208
00209         new Thread()
00210         {
00211             @Override public void run()
00212             {
00213                 byte[] emission = new byte[1024];
00214
00215                 try
00216                 {
00217                     emission = requete.getBytes();
00218                     DatagramPacket paquetRetour = new DatagramPacket(emission, emission.length,
      adresseIPDistante, PORT);
00219                     socket.send(paquetRetour);
00220                     Log.d(TAG, "send() " + adresseIP + " = " + requete);
00221                 }
00222                 catch (IOException e)
00223                 {
00224                     Log.d(TAG, "Erreur send() [socket.isClosed = " + socket.isClosed() + "]");
00225                     e.printStackTrace();
00226                 }
00227             }
00228         }.start();
00229     }
00230
00235     public void arreter()
00236     {
00237         if(socket == null)
00238             return;
00239         socket.close();
00240     }
00241
00246     @Override
00247     public void run()
00248     {
00249         recevoir();
00250     }
00251 }
```

8.5 Référence du fichier ConfigurationSalleActivity.java

Déclaration de la classe ConfigurationSalleActivity.

Classes

— class com.lasalle.meeting.ConfigurationSalleActivity
Déclaration de la classe ConfigurationSalleActivity.

Paquetages

— package com.lasalle.meeting

8.5.1 Description détaillée

Déclaration de la classe ConfigurationSalleActivity.

Auteur

Vincent DEVINE

Définition dans le fichier ConfigurationSalleActivity.java.

8.6 ConfigurationSalleActivity.java

```
00001 package com.lasalle.meeting;
00002
00003 import androidx.appcompat.app.AppCompatActivity;
00004
00005 import android.content.Intent;
00006 import android.os.Bundle;
00007 import android.os.Handler;
00008 import android.os.Message;
00009 import android.util.Log;
00010 import android.view.View;
00011 import android.widget.AdapterView;
00012 import android.widget.ArrayAdapter;
00013 import android.widget.Button;
00014 import android.widget.EditText;
00015 import android.widget.Spinner;
00016 import android.widget.TabHost;
00017
00018 import java.util.ArrayList;
00019 import java.util.List;
00020 import java.util.Vector;
00021
00032 public class ConfigurationSalleActivity extends AppCompatActivity
00033 {
00037     private final static String TAG = "ConfigurationSalleActivity";
00038
00041     private Spinner listeSalles;
00042     private List<String> IPSalle;
00043     private ArrayAdapter<String> adapter;
00044     private static Vector<Salle> salles;
00045     private Communication communication = null;
00046     private int positionSalleList = 0;
00047     private String nom ="";
00048     private String emplacement ="";
00049     private String description ="";
00050     private String surface ="";
00051
00054     private EditText editNom;
00055     private EditText editEmplacement;
00056     private EditText editDescription;
00057     private EditText editSurface;
00058     private Button buttonEnvoie;
00059
00065     @Override
00066     protected void onCreate(Bundle savedInstanceState)
```

```
00067     {
00068         Log.d(TAG, "onCreate()");
00069 
00070         super.onCreate(savedInstanceState);
00071         setContentView(R.layout.activity_configuration_salle);
00072 
00073         communication = MainActivity.getCommunication();
00074 
00075         initialiserRessourcesLayout();
00076         initialiserSpinner();
00077         setListener();
00078     }
00079 
00084     public void initialiserRessourcesLayout()
00085     {
00086         Log.d(TAG, "initialiserRessourcesLayout()");
00087 
00088         listeSalles = (Spinner)findViewById(R.id.listeSalles);
00089         editNom = (EditText)findViewById(R.id.EditNom);
00090         editEmplacement= (EditText)findViewById(R.id.EditEmplacement);
00091         editDescription= (EditText)findViewById(R.id.EditDescription);
00092         editSurface= (EditText)findViewById(R.id.EditSurface);
00093         buttonEnvoie= (Button)findViewById(R.id.buttonEnvoie);
00094     }
00095 
00100     public void initialiserSpinner()
00101     {
00102         Log.d(TAG, "initialiserSpinner()");
00103 
00104         IPSalle = new ArrayList<String>();
00105 
00106         salles = MainActivity.getMesSalles();
00107 
00108         for(int i = 0; i < salles.size(); ++i)
00109         {
00110             Log.d(TAG, "Ajout adresse IP : " + salles.elementAt(i).getAdresseIP());
00111             IPSalle.add(salles.elementAt(i).getAdresseIP());
00112         }
00113 
00114         adapter = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item,
    IPSalle);
00115         adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
00116 
00117         listeSalles.setAdapter(adapter);
00118 
00119         listeSalles.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()
00120         {
00121             @Override
00122             public void onItemSelected(AdapterView<?> arg0, View arg1, int position, long id)
00123             {
00124                 positionSalleList = position;
00125                 Log.d(TAG, "position : " + position + " - " + "nom : " + IPSalle.get(position));
00126             }
00127             @Override
00128             public void onNothingSelected(AdapterView<?> arg0)
00129             {
00130             }
00131         });
00132     }
00133 
00138     public void setListener()
00139     {
00140         Log.d(TAG, "setListener()");
00141 
00142         buttonEnvoie.setOnClickListener(
00143             new View.OnClickListener()
00144             {
00145                 @Override
00146                 public void onClick(View v)
00147                 {
00148                     recupererInformation();
00149 
00150                     Log.d(TAG,"trame : $SET;1;" + nom + ";" + description + ";" + emplacement + ";" +
    surface + "\r\n" + IPSalle.get(positionSalleList));
00151                     if(communication != null)
00152                     {
00153                         communication.envoyer("$SET;1;" + nom + ";" + description + ";" +
    emplacement + ";" + surface + "\r\n", IPSalle.get(positionSalleList));
00154                     }
00155                 }
00156             }
00157         );
00158     }
00159 
00164     public void recupererInformation()
00165     {
00166         Log.d(TAG, "recupererInformation()");
00167 
00168         nom = editNom.getText().toString();
00169         description = editDescription.getText().toString();
00170         emplacement = editEmplacement.getText().toString();
```

```
00171         surface = editSurface.getText().toString();
00172     }
00173 }
```

8.7 Référence du fichier MainActivity.java

Déclaration de la classe MainActivity.

Classes

— class com.lasalle.meeting.MainActivity
Déclaration de la classe MainActivity.

Paquetages

— package com.lasalle.meeting

8.7.1 Description détaillée

Déclaration de la classe MainActivity.

Auteur

Vincent DEVINE

Définition dans le fichier MainActivity.java.

8.8 MainActivity.java

```
00001 package com.lasalle.meeting;
00002
00003 import androidx.annotation.NonNull;
00004 import androidx.appcompat.app.AlertDialog;
00005 import androidx.appcompat.app.AppCompatActivity;
00006 import androidx.recyclerview.widget.LinearLayoutManager;
00007 import androidx.recyclerview.widget.RecyclerView;
00008 import androidx.swiperefreshlayout.widget.SwipeRefreshLayout;
00009
00010 import android.content.Context;
00011 import android.content.DialogInterface;
00012 import android.content.Intent;
00013 import android.net.ConnectivityManager;
00014 import android.net.DhcpInfo;
00015 import android.net.wifi.WifiInfo;
00016 import android.net.wifi.WifiManager;
00017 import android.os.Bundle;
00018 import android.os.Handler;
00019 import android.os.Message;
00020 import android.util.Log;
00021 import android.view.LayoutInflater;
00022 import android.view.Menu;
00023 import android.view.MenuInflater;
00024 import android.view.MenuItem;
00025 import android.view.View;
00026 import android.widget.AdapterView;
00027 import android.widget.ArrayAdapter;
00028 import android.widget.Button;
00029 import android.widget.Spinner;
00030 import android.widget.Toast;
00031
00032 import com.google.android.material.bottomnavigation.BottomNavigationView;
00033
00034 import java.io.Serializable;
00035 import java.net.InetAddress;
00036 import java.util.ArrayList;
00037 import java.util.List;
```

```
00038 import java.util.Vector;
00039
00040 import static java.lang.Boolean.TRUE;
00041
00052 public class MainActivity extends AppCompatActivity  implements
      RechercherNomBoiteDialogue.rechercheNomBoiteDialogueListener
00053 {
00057     private static final String TAG = "MainActivity";
00058     private final String SEPARATEUR = ";";
00059     private static final int FROID = -3;
00060     private static final int FRAIS = -2;
00061     private static final int LEGEREMENTFRAIS = -1;
00062     private static final int NEUTRE = 0;
00063     private static final int LEGEREMENTTIEDE = 1;
00064     private static final int TIEDE = 2;
00065     private static final int CHAUD = 3;
00066
00070     private Spinner filtre;
00071     private ArrayAdapter<String> adapterList;
00072     private SwipeRefreshLayout swipeRefreshLayout;
00073     private RecyclerView recyclerView;
00074     private RecyclerView.Adapter adapter;
00075     private RecyclerView.LayoutManager layoutManager;
00076     private BottomNavigationView bottomNavigationView;
00077     private AlertDialog.Builder boiteConfort;
00078     private AlertDialog.Builder boiteDisponibilite;
00079     private AlertDialog.Builder boiteNom;
00080
00084     private static Vector<Salle> mesSalles;
00085     private static Vector<Salle> sallesFilter;
00086     private static Communication communication = null;
00087     private Salle salle = null;
00088     private WifiManager wm = null;
00089
00095     @Override
00096     protected void onCreate(Bundle savedInstanceState)
00097     {
00098         Log.d(TAG, "onCreate()");
00099
00100         super.onCreate(savedInstanceState);
00101         setContentView(R.layout.activity_main);
00102
00103         mesSalles = new Vector<Salle>();
00104
00105         initialiserRessourcesLayout();
00106         connecterWiFi();
00107         initialiserSalles();
00108     }
00109
00114     @Override
00115     protected void onStart()
00116     {
00117         super.onStart();
00118         Log.d(TAG, "onStart()");
00119
00120         rafraichir(mesSalles);
00121     }
00122
00127     @Override
00128     public boolean onCreateOptionsMenu(Menu menu)
00129     {
00130         MenuInflater inflater = getMenuInflater();
00131         inflater.inflate(R.menu.menu, menu);
00132         return true;
00133     }
00134
00139     @Override
00140     public boolean onOptionsItemSelected(MenuItem item)
00141     {
00142         final Intent intent = new Intent(MainActivity.this,
      ConfigurationSalleActivity.class);
00143
00144         switch (item.getItemId())
00145         {
00146             case R.id.configurerSalle:
00147                 startActivity(intent);
00148                 return true;
00149             case R.id.aPropos:
00150                 Toast.makeText(getApplicationContext(), "La fonctionnalité à propos de l'application n'est
       pas encore disponible !", Toast.LENGTH_SHORT).show();
00151                 return true;
00152         }
00153         return super.onOptionsItemSelected(item);
00154     }
00155
00160     public void initialiserRessourcesLayout()
00161     {
00162         swipeRefreshLayout = (SwipeRefreshLayout)findViewById(R.id.swipeRefreshLayout);
00163         recyclerView = (RecyclerView)findViewById(R.id.listeSalle);
00164         bottomNavigationView = (BottomNavigationView) findViewById(R.id.bottomNavigationView);
00165         filtre = (Spinner)findViewById(R.id.filtre);
```

```
00166
00167         setListenBouton();
00168         initialiserBoiteDialogue();
00169         initialiserSpinner();
00170     }
00171
00176     public void setListenBouton()
00177     {
00178         swipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener()
00179         {
00180             @Override
00181             public void onRefresh()
00182             {
00183                 rafraichir(mesSalles);
00184             }
00185         });
00186
00187         bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.
      OnNavigationItemSelectedListener()
00188         {
00189             @Override
00190             public boolean onNavigationItemSelected(@NonNull MenuItem item)
00191             {
00192                 switch (item.getItemId())
00193                 {
00194                     case R.id.Salle:
00195                         rafraichir(mesSalles);
00196                         Toast.makeText(getApplicationContext(), "Rafraichisement", Toast.LENGTH_SHORT).show
      ();
00197                         return true;
00198                     case R.id.Favoris:
00199                         Toast.makeText(getApplicationContext(), "La fonctionnalité favoris n'est pas encore
       disponible !", Toast.LENGTH_SHORT).show();
00200                         return true;
00201                     case R.id.Rechercher:
00202                         /*final Intent intent = new Intent(MainActivity.this, RechercheActivity.class);
00203                         intent.putExtra("mesSalles", mesSalles);
00204                         startActivity(intent);*/
00205                         Toast.makeText(getApplicationContext(), "La fonctionnalité rechercher n'est pas
       encore disponible !", Toast.LENGTH_SHORT).show();
00206                         return true;
00207                 }
00208                 return false;
00209             }
00210         });
00211     }
00212
00217     private void connecterWiFi()
00218     {
00219         wm = (WifiManager) getApplicationContext().getSystemService(Context.WIFI_SERVICE);
00220         if (!wm.isWifiEnabled())
00221         {
00222             Log.d(TAG, "connecterWiFi() WiFi indisponible !");
00223             wm.setWifiEnabled(true);
00224         }
00225         else
00226         {
00227             Log.d(TAG, "connecterWiFi() WiFi disponible");
00228         }
00229
00230         WifiInfo wi = wm.getConnectionInfo();
00231         Log.d(TAG, "connecterWiFi() " + wi.toString() + " " + wi.getIpAddress() + " " + wi.getMacAddress())
      ;
00232
00233         DhcpInfo di = wm.getDhcpInfo();
00234         Log.d(TAG, "connecterWiFi() " + di.toString());
00235
00236         communication = new Communication(handlerUI);
00237         Thread tCommunicationUDP = new Thread(communication, "Communication");
00238         tCommunicationUDP.start();
00239     }
00240
00245     private void initialiserSalles()
00246     {
00247         Log.d(TAG, "initialiserSalles()");
00248
00249         recyclerView.setHasFixedSize(true);
00250         layoutManager = new LinearLayoutManager(this);
00251
00252         recyclerView.setLayoutManager(layoutManager);
00253
00254         adapter = new SalleAdapter(mesSalles);
00255         recyclerView.setAdapter(adapter);
00256
00257         // cf. appel à rafraichir() dans onStart()
00258         /*if(communication != null)
00259         {
00260             communication.envoyer("$GET;1\r\n"); // voir protocole
00261         }*/
00262     }
00263
```

```
00268     private void reInitialiserSalles()
00269     {
00270         Log.d(TAG, "reInitialiserSalles()");
00271 
00272         recyclerView.setHasFixedSize(true);
00273         layoutManager = new LinearLayoutManager(this);
00274 
00275         recyclerView.setLayoutManager(layoutManager);
00276 
00277         adapter = new SalleAdapter(sallesFilter);
00278         recyclerView.setAdapter(adapter);
00279     }
00280 
00286     public void ajouterSalle(Salle maSalle)
00287     {
00288         int positionMemeSalle = verifierExistenceSalle(maSalle);
00289 
00290         if(positionMemeSalle == -1)
00291         {
00292             mesSalles.add(maSalle);
00293             adapter.notifyDataSetChanged();
00294         }
00295         else if(verifierChangementSalle(maSalle, positionMemeSalle))
00296         {
00297             mesSalles.removeElementAt(positionMemeSalle);
00298             mesSalles.add(maSalle);
00299             adapter.notifyDataSetChanged();
00300         }
00301     }
00302 
00308     public int verifierExistenceSalle(Salle maSalle)
00309     {
00310         for(int i = 0; i < mesSalles.size(); ++i)
00311         {
00312             if(maSalle.getAdresseIP().equals(mesSalles.elementAt(i).getAdresseIP()))
00313             {
00314                 return i;
00315             }
00316         }
00317         return -1;
00318     }
00319 
00320     //TODO a regarder
00321     public int verifierExistenceSalle(String adresseIP)
00322     {
00323         for(int i = 0; i < mesSalles.size(); ++i)
00324         {
00325             if(mesSalles.elementAt(i).getAdresseIP().equals(adresseIP))
00326             {
00327                 return i;
00328             }
00329         }
00330         return -1;
00331     }
00332 
00338     public boolean verifierChangementSalle(Salle maSalle, int positionMemeSalle
      )
00339     {
00340         if(maSalle.getLibre() == mesSalles.elementAt(positionMemeSalle).getLibre())
00341         {
00342             return true;
00343         }
00344         else if(maSalle.getTemperature() == mesSalles.elementAt(positionMemeSalle).
      getTemperature())
00345         {
00346             return true;
00347         }
00348         else if(maSalle.getConfort() == mesSalles.elementAt(positionMemeSalle).getConfort())
00349         {
00350             return true;
00351         }
00352         else if(maSalle.getNom().equals(mesSalles.elementAt(positionMemeSalle).getNom()))
00353         {
00354             return true;
00355         }
00356         else if(maSalle.getEmplacement().equals(mesSalles.elementAt(positionMemeSalle).
      getEmplacement()))
00357         {
00358             return true;
00359         }
00360         else if(maSalle.getDescription().equals(mesSalles.elementAt(positionMemeSalle).
      getDescription()))
00361         {
00362             return true;
00363         }
00364         else if(maSalle.getSurface() == mesSalles.elementAt(positionMemeSalle).getSurface())
00365         {
00366             return true;
00367         }
00368 
00369         return false;
```

```
00370     }
00371
00377     public void rafraichir(Vector<Salle> mesSalles)
00378     {
00379         Log.d(TAG, "rafraichir()");
00380         if(communication != null)
00381             communication.envoyer("$GET;1\r\n"); // voir protocole
00382
00383         swipeRefreshLayout.setRefreshing(false);
00384         adapter.notifyDataSetChanged();
00385     }
00386
00392     private Handler handlerUI = new Handler()
00393     {
00394         @Override
00395         public void handleMessage(Message msg)
00396         {
00397             super.handleMessage(msg);
00398             Bundle b = msg.getData();
00399
00400             switch(b.getInt("etat"))
00401             {
00402                 case Communication.CODE_RECEPTION:
00403                     String trame = b.getString("trame");
00404                     if(!trame.startsWith(Communication.
      DELIMITEUR_EN_TETE))
00405                         return;
00406                     if(!trame.endsWith(Communication.DELIMITEUR_FIN))
00407                         return;
00408                     Log.d(TAG,"handleMessage() trame reçue de " + b.getString("adresseIP") + ":" + b.getInt
      ("port"));
00409                     Log.d(TAG,"handleMessage() trame = " + trame.replace("\r\n", ""));
00413                     String nouvelleTrame = trame.replace(Communication.
      DELIMITEUR_EN_TETE, "");
00414                     nouvelleTrame = nouvelleTrame.replace(Communication.
      DELIMITEUR_FIN, "");
00415                     int nbDelimiteurs = getNbDelimiteurs(trame);
00416                     String[] tramesDecompose = decomposerTrame(nouvelleTrame);
00417                     if(nbDelimiteurs == Communication.
      NB_DELIMITEURS_GET_1)
00418                     {
00419                         Salle salle = creerSalle(tramesDecompose, b.getString("adresseIP"));
00420                         if (salle != null)
00421                         {
00422                             ajouterSalle(salle);
00423                         }
00424                     }
00425                     else if(nbDelimiteurs == Communication.
      NB_DELIMITEURS_GET_2)
00426                     {
00427
00428                     }
00429                     else if(nbDelimiteurs == Communication.
      NB_DELIMITEURS_GET_3)
00430                     {
00431
00432                     }
00433                     break;
00434                 default:
00435                     Log.d(TAG,"handleMessage() code état inconnu ! ");
00436             }
00437         }
00438     };
00439
00445     public Salle creerSalle(String[] trameDecompose, String adresseIP)
00446     {
00447         Log.d(TAG, "creerSalle() adresseIP = " + adresseIP);
00448         if(!trameDecompose[0].isEmpty())
00449         {
00450             Log.d(TAG, "creerSalle() trameDecompose[0] : " + trameDecompose[0] + " (nomSalle)");
00451             Log.d(TAG, "creerSalle() trameDecompose[1] : " + trameDecompose[1] + " (description)");
00452             Log.d(TAG, "creerSalle() trameDecompose[2] : " + trameDecompose[2] + " (emplacement)");
00453             Log.d(TAG, "creerSalle() trameDecompose[3] : " + trameDecompose[3] + " (surface)");
00454             Log.d(TAG, "creerSalle() trameDecompose[4] : " + trameDecompose[4] + " (disponibilité)");
00455             Log.d(TAG, "creerSalle() trameDecompose[5] : " + trameDecompose[5] + " (niveauDeConfort)");
00456             Log.d(TAG, "creerSalle() trameDecompose[6] : " + trameDecompose[6] + " (température)");
00457             int surface = 0;
00458             if(estEntier(trameDecompose[3]))
00459                 surface = Integer.parseInt(trameDecompose[3]);
00460             int disponible = 0;
00461             if(estEntier(trameDecompose[4]))
00462                 disponible = Integer.parseInt(trameDecompose[4]);
00463             int niveauConfort = 0;
00464             if(estEntier(trameDecompose[5]))
00465                 niveauConfort = Integer.parseInt(trameDecompose[5]);
00466             float temperature = 0;
00467             if(estReel(trameDecompose[6]))
00468                 temperature = Float.parseFloat(trameDecompose[6]);
00469
00470             salle = new Salle(trameDecompose[0], trameDecompose[1], trameDecompose[2], disponible,
      surface, niveauConfort, temperature, adresseIP);
```

```
00471             return salle;
00472         }
00473         else
00474         {
00475             Log.d(TAG, "creerSalle() pas de nom de salle");
00476             String inconnuString = "???";
00477             int inconnuInt = 1;
00478 
00479             salle = new Salle(inconnuString, inconnuString , inconnuString , Integer.parseInt(
    trameDecompose[4]), inconnuInt,  Integer.parseInt(trameDecompose[5]), Float.parseFloat(trameDecompose[6]), adresseIP
    );
00480             return salle;
00481         }
00482     }
00483 
00489     public String[] decomposerTrame(String trame)
00490     {
00491         String[] tramesDecompose = trame.split(SEPARATEUR);
00492         return tramesDecompose;
00493     }
00494 
00500     private int getNbDelimiteurs(String trame)
00501     {
00502         int nb = 0;
00503         for (int i = 0; i < trame.length(); i++)
00504         {
00505             if (trame.charAt(i) == ';')
00506             {
00507                 nb++;
00508             }
00509         }
00510         return nb;
00511     }
00512 
00517     public static Vector<Salle> getMesSalles()
00518     {
00519         return mesSalles;
00520     }
00521 
00526     public static Communication getCommunication()
00527     {
00528         return communication;
00529     }
00530 
00531     private boolean estEntier(String donnee)
00532     {
00533         try
00534         {
00535             Integer.parseInt(donnee);
00536         }
00537         catch(NumberFormatException e)
00538         {
00539             return false;
00540         }
00541         catch(NullPointerException e)
00542         {
00543             return false;
00544         }
00545 
00546         return true;
00547     }
00548 
00549     private boolean estReel(String donnee)
00550     {
00551         try
00552         {
00553             Float.parseFloat(donnee);
00554         }
00555         catch(NumberFormatException e)
00556         {
00557             return false;
00558         }
00559         catch(NullPointerException e)
00560         {
00561             return false;
00562         }
00563 
00564         return true;
00565     }
00566 
00571     public void initialiserBoiteDialogue()
00572     {
00573         boiteDisponibilite = new AlertDialog.Builder(this);
00574         boiteDisponibilite.setMessage("Voulez vous voir seulement les salles : ");
00575         boiteDisponibilite.setPositiveButton("Disponible", new DialogInterface.
    OnClickListener()
00576         {
00577             public void onClick(DialogInterface dialog, int which)
00578             {
00579                 filtreDisponible();
00580             }
```

```
00581         });
00582         boiteDisponibilite.setNegativeButton("Occupé", new DialogInterface.
      OnClickListener()
00583         {
00584             public void onClick(DialogInterface dialog, int which)
00585             {
00586                 filtreNonDisponible();
00587             }
00588         });
00589
00590         boiteConfort = new AlertDialog.Builder(this);
00591         boiteConfort.setTitle("Choisir le niveau de confort");
00592         String[] niveauConfort = {"Chaud","Tiède","Légèrement tiède","Neutre","Légèrement fraiche","Fraiche
      ","Froid"};
00593         boiteConfort.setItems(niveauConfort, new DialogInterface.OnClickListener() {
00594             @Override
00595             public void onClick(DialogInterface dialog, int which) {
00596                 switch (which) {
00597                     case 0:
00598                         filtreChaud();
00599                         break;
00600                     case 1:
00601                         filtreTiede();
00602                         break;
00603                     case 2:
00604                         filtreLegerementTiede();
00605                         break;
00606                     case 3:
00607                         filtreNeutre();
00608                         break;
00609                     case 4:
00610                         filtreLegerementFrais();
00611                         break;
00612                     case 5:
00613                         filtreFrais();
00614                         break;
00615                     case 6:
00616                         filtreFroid();
00617                         break;
00618                 }
00619             }
00620         });
00621     }
00622
00627     public void initialiserSpinner()
00628     {
00629         Log.d(TAG, "initialiserSpinner()");
00630
00631         List choixFiltre;
00632         choixFiltre = new ArrayList<String>();
00633
00634         choixFiltre.add("Aucun");
00635         choixFiltre.add("Disponibilité");
00636         choixFiltre.add("Confort");
00637         choixFiltre.add("Nom");
00638
00639         adapterList = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, choixFiltre);
00640         adapterList.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
00641
00642         filtre.setAdapter(adapterList);
00643
00644         filtre.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()
00645         {
00646             @Override
00647             public void onItemSelected(AdapterView<?> arg0, View arg1, int position, long id)
00648             {
00649                 Toast.makeText(getApplicationContext(), "Le choix du filtre est : " + position, Toast.
      LENGTH_SHORT).show();
00650                 Log.d(TAG, "position : " + position);
00651                 initialiserFiltre(position);
00652             }
00653             @Override
00654             public void onNothingSelected(AdapterView<?> arg0)
00655             {
00656             }
00657         });
00658     }
00659
00660     public void initialiserFiltre(int positionFiltre)
00661     {
00662         if(positionFiltre == 0)
00663         {
00664             initialiserSalles();
00665         }
00666         else if(positionFiltre == 1)
00667         {
00668             boiteDisponibilite.show();
00669         }
00670         else if(positionFiltre == 2)
00671         {
00672             boiteConfort.show();
```

```
00673         }
00674         else if(positionFiltre == 3)
00675         {
00676             RechercherNomBoiteDialogue rechercherNomBoiteDialogue = new
      RechercherNomBoiteDialogue();
00677             rechercherNomBoiteDialogue.show(getSupportFragmentManager(), "rechercher une salle par son nom"
      );
00678         }
00679     }
00680
00681     public void filtreDisponible()
00682     {
00683         Log.d(TAG, "filtreDisponible()");
00684         sallesFilter = new Vector<Salle>();
00685
00686         for(int i = 0; i < mesSalles.size(); i++)
00687         {
00688             if(mesSalles.elementAt(i).getLibre())
00689             {
00690                 sallesFilter.add(mesSalles.get(i));
00691             }
00692         }
00693         reInitialiserSalles();
00694     }
00695
00696     public void filtreNonDisponible()
00697     {
00698         Log.d(TAG, "filtreNonDisponible()");
00699         sallesFilter = new Vector<Salle>();
00700
00701         for(int i = 0; i < mesSalles.size(); i++)
00702         {
00703             if(!mesSalles.elementAt(i).getLibre())
00704             {
00705                 sallesFilter.add(mesSalles.elementAt(i));
00706             }
00707         }
00708         reInitialiserSalles();
00709     }
00710
00711     public void filtreTiede()
00712     {
00713         Log.d(TAG, "filtreTiede()");
00714         sallesFilter = new Vector<Salle>();
00715
00716         for(int i = 0; i < mesSalles.size(); i++)
00717         {
00718             if(mesSalles.elementAt(i).getConfort() == TIEDE)
00719             {
00720                 sallesFilter.add(mesSalles.elementAt(i));
00721             }
00722         }
00723         reInitialiserSalles();
00724     }
00725
00726     public void filtreLegerementTiede()
00727     {
00728         Log.d(TAG, "filtreLegerementTiede()");
00729         sallesFilter = new Vector<Salle>();
00730
00731         for(int i = 0; i < mesSalles.size(); i++)
00732         {
00733             if(mesSalles.elementAt(i).getConfort() == LEGEREMENTTIEDE)
00734             {
00735                 sallesFilter.add(mesSalles.elementAt(i));
00736             }
00737         }
00738         reInitialiserSalles();
00739     }
00740
00741     public void filtreNeutre()
00742     {
00743         Log.d(TAG, "filtreNeutre()");
00744         sallesFilter = new Vector<Salle>();
00745
00746         for(int i = 0; i < mesSalles.size(); i++)
00747         {
00748             if(mesSalles.elementAt(i).getConfort() == NEUTRE)
00749             {
00750                 sallesFilter.add(mesSalles.elementAt(i));
00751             }
00752         }
00753         reInitialiserSalles();
00754     }
00755
00756     public void filtreLegerementFrais()
00757     {
00758         Log.d(TAG, "filtreLegerementFrais()");
00759         sallesFilter = new Vector<Salle>();
00760
00761         for(int i = 0; i < mesSalles.size(); i++)
```

```
00762         {
00763             if(mesSalles.elementAt(i).getConfort() == LEGEREMENTFRAIS)
00764             {
00765                 sallesFilter.add(mesSalles.elementAt(i));
00766             }
00767         }
00768         reInitialiserSalles();
00769     }
00770
00771     public void filtreFrais()
00772     {
00773         Log.d(TAG, "filtreFrais()");
00774         sallesFilter = new Vector<Salle>();
00775
00776         for(int i = 0; i < mesSalles.size(); i++)
00777         {
00778             if(mesSalles.elementAt(i).getConfort() == FRAIS)
00779             {
00780                 sallesFilter.add(mesSalles.elementAt(i));
00781             }
00782         }
00783         reInitialiserSalles();
00784     }
00785
00786     public void filtreChaud()
00787     {
00788         Log.d(TAG, "filtreChaud()");
00789         sallesFilter = new Vector<Salle>();
00790
00791         for(int i = 0; i < mesSalles.size(); i++)
00792         {
00793             if(mesSalles.elementAt(i).getConfort() == CHAUD)
00794             {
00795                 sallesFilter.add(mesSalles.elementAt(i));
00796             }
00797         }
00798         reInitialiserSalles();
00799     }
00800
00801     public void filtreFroid()
00802     {
00803         Log.d(TAG, "filtreFroid()");
00804         sallesFilter = new Vector<Salle>();
00805
00806         for(int i = 0; i < mesSalles.size(); i++)
00807         {
00808             if(mesSalles.elementAt(i).getConfort() == FROID)
00809             {
00810                 sallesFilter.add(mesSalles.elementAt(i));
00811             }
00812         }
00813         reInitialiserSalles();
00814     }
00815
00816     @Override
00817     public void applyTexts(String nomSalleRechercher) {
00818         filtreNom(nomSalleRechercher);
00819     }
00820
00821     public void filtreNom(String nomSalleRechercher)
00822     {
00823         Log.d(TAG, "filtreNom()");
00824         sallesFilter = new Vector<Salle>();
00825
00826         for(int i = 0; i < mesSalles.size(); i++)
00827         {
00828             if(mesSalles.elementAt(i).getNom().equals(nomSalleRechercher))
00829             {
00830                 sallesFilter.add(mesSalles.elementAt(i));
00831             }
00832         }
00833         reInitialiserSalles();
00834     }
00835 }
```

8.9 Référence du fichier README.md

8.10 README.md

```
00001 \mainpage Le projet
00002
00003 \tableofcontents
00004
00005 Placé à l'extérieur d'une pièce (salle de réunion ou de travail, ...), le système **Meeting**
```

```
      permettra d'accéder en temps réel aux informations de l'espace concerné : il affichera la disponibilité et l'état de
      confort et permettra de réaliser sa réservation.
00006
00007 \section section_tdm Table des matières
00008 - \ref page_README
00009 - \ref page_changelog
00010 - \ref page_about
00011 - \ref page_licence
00012
00013 \section section_infos Informations
00014
00015 \author Vincent Devine <vincentdevine84@gmail.com>
00016 \date 2020
00017 \version 0.2
00018 \see https://svn.riouxsvn.com/meeting/
00019
00020
00021 \page page_README README
00022
00023 [TOC]
00024
00025 # Projet {#projet}
00026
00027 ## Présentation {#presentation}
00028
00029 Placé à l'extérieur d'une pièce (salle de réunion ou de travail, ...), le système **Meeting**
      permettra d'accéder en temps réel aux informations de l'espace concerné : il affichera la disponibilité et l'état de
      confort et permettra de réaliser sa réservation.
00030
00031 L'objectif est de proposer une solution simple, alliant flexibilité et ergonomie. Le système affiche
      de la disponibilité d'accès et le niveau de confort d'une salle de travail ou de réunion.
00032
00033 Le portier connecté est composé :
00034
00035 * d'un micro-contrôleur (ESP32, Z-duino,...)
00036 * d'un écran tactile
00037 * d'indicateurs lumineux (Leds)
00038 * d'une liaison Bluetooth vers une sonde permettant d'évaluer un "indice de confort"
00039
00040 La sonde est composée :
00041
00042 * d'un capteur de température
00043 * d'un capteur d'hygrométrie
00044 * d'un capteur de qualité d'air
00045
00046 A partir d'une application mobile sous Android et communiquant en UPD via le WiFi, l'utilisateur
      pourra :
00047
00048 * Lorsqu'une salle a été sélectionnée, visualiser les informations sur la salle (son nom et des
      informations complémentaires qu'il pourra associer à celle-ci comme sa localisation, sa surface ...), sa
      disponibilité et les mesures en provenance du module sonde ainsi que son indice de confort.
00049
00050 * configurer les informations d'une salle.
00051
00052 * la possibilité de prendre une salle en indiquant une durée estimée d'occupation de celle-ci. Dans ce
      cas, le portier lui enverra un code qu'il utilisera pour augmenter la durée d'occupation ou de libérer la
      salle.
00053
00054 ## Informations {#informations}
00055
00056 \author Vincent Devine <vincentdevine84@gmail.com>
00057 \date 2020
00058 \version 0.2
00059 \see https://svn.riouxsvn.com/meeting/
00060
00061
00062 \page page_about A propos
00063
00064 \author Vincent Devine <vincentdevine84@gmail.com>
00065 \date 2020
00066 \version 0.2
00067 \see https://svn.riouxsvn.com/meeting/
00068
00069
00070 \page page_licence Licence GPL
00071
00072 This program is free software; you can redistribute it and/or modify
00073 it under the terms of the GNU General Public License as published by
00074 the Free Software Foundation; either version 2 of the License, or
00075 (at your option) any later version.
00076
00077 This program is distributed in the hope that it will be useful,
00078 but WITHOUT ANY WARRANTY; without even the implied warranty of
00079 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00080 GNU General Public License for more details.
00081
00082 You should have received a copy of the GNU General Public License
00083 along with this program; if not, write to the Free Software
00084 Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```

8.11 Référence du fichier RechercherNomBoiteDialogue.java

Classes

— interface com.lasalle.meeting.RechercherNomBoiteDialogue.rechercheNomBoiteDialogueListener
— class com.lasalle.meeting.RechercherNomBoiteDialogue

Paquetages

— package com.lasalle.meeting

8.12 RechercherNomBoiteDialogue.java

```
00001 package com.lasalle.meeting;
00002
00003 import android.app.AlertDialog;
00004 import android.app.Dialog;
00005 import android.content.Context;
00006 import android.content.DialogInterface;
00007 import android.os.Bundle;
00008 import android.text.Layout;
00009 import android.view.LayoutInflater;
00010 import android.view.View;
00011 import android.widget.EditText;
00012
00013 import androidx.appcompat.app.AppCompatDialogFragment;
00014
00015 public class RechercherNomBoiteDialogue extends AppCompatDialogFragment
00016 {
00017     private EditText nomSalleRechercherBoiteDialogue;
00018     private rechercheNomBoiteDialogueListener
      listener;
00019
00020
00021     @Override
00022     public Dialog onCreateDialog(Bundle savedInstanceState)
00023     {
00024         AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
00025
00026         LayoutInflater inflater = getActivity().getLayoutInflater();
00027         View view = inflater.inflate(R.layout.dialog_nom, null);
00028
00029         builder.setView(view)
00030                 .setTitle("Recherche par nom")
00031                 .setNegativeButton("Annulé", new DialogInterface.OnClickListener() {
00032                     @Override
00033                     public void onClick(DialogInterface dialog, int which) {
00034
00035                     }
00036                 })
00037                 .setPositiveButton("ok", new DialogInterface.OnClickListener() {
00038                     @Override
00039                     public void onClick(DialogInterface dialog, int which) {
00040                     String nomSalleRechercher = nomSalleRechercherBoiteDialogue.getText().toString();
00041                     listener.applyTexts(nomSalleRechercher);
00042                     }
00043                 });
00044
00045         nomSalleRechercherBoiteDialogue = view.findViewById(R.id.nomSalleRechercher);
00046
00047         return builder.create();
00048     }
00049
00050     @Override
00051     public void onAttach(Context context) {
00052         super.onAttach(context);
00053
00054         try
00055         {
00056             listener = (rechercheNomBoiteDialogueListener) context;
00057         }
00058         catch (ClassCastException e) {
00059             throw new ClassCastException(context.toString() + "doit implementer
       rechercheNomBoiteDiallogueListener");
00060         }
00061     }
00062
00063     public interface rechercheNomBoiteDialogueListener
00064     {
00065         void applyTexts(String nomSalleRechercher);
00066     }
00067 }
```

8.13 Référence du fichier Salle.java

Déclaration de la classe Salle.

Classes

— class com.lasalle.meeting.Salle
Déclaration de la classe Salle.

Paquetages

— package com.lasalle.meeting

8.13.1 Description détaillée

Déclaration de la classe Salle.

Auteur

Vincent DEVINE

Définition dans le fichier Salle.java.

8.14 Salle.java

```
00001 package com.lasalle.meeting;
00002
00009 import android.util.Log;
00010
00011 import java.io.Serializable;
00012
00017 public class Salle implements Serializable
00018 {
00022     private static final String TAG = "Salle";
00023     private static final int FROID = -3;
00024     private static final int FRAIS = -2;
00025     private static final int LEGEREMENTFRAIS = -1;
00026     private static final int NEUTRE = 0;
00027     private static final int LEGEREMENTTIEDE = 1;
00028     private static final int TIEDE = 2;
00029     private static final int CHAUD = 3;
00030
00034     private String nom = "";
00035     private String description = "";
00036     private String emplacement = "";
00037     private boolean libre;
00038     private int surface;
00039     private int confort;
00040     private float temperature;
00041     private String adresseIP;
00042
00054     public Salle(String nom, String description, String emplacement, int libre, int surface, int
      confort, float temperature, String adresseIP)
00055     {
00056         this.nom = nom;
00057         this.description = description;
00058         this.emplacement = emplacement;
00059         setLibre(libre);
00060         this.surface = surface;
00061         this.confort = confort;
00062         this.temperature = temperature;
00063         this.adresseIP = adresseIP;
00064         Log.d(TAG, "Salle : nom = " + nom + " - description = " + description + " - emplacement " +
      emplacement + " - libre = " + libre + " - surface = " + surface + " - confort = " + confort + " - température = " +
       temperature + " - adresseIP = " + adresseIP);
00065     }
00066
00071     public void setEmplacement(String nouvelleEmplacement)
00072     {
```

```
00073         emplacement = nouvelleEmplacement;
00074     }
00075
00080     public void setNom(String nouveauNom)
00081     {
00082         nom = nouveauNom;
00083     }
00084
00089     public void setLibre(int libre)
00090     {
00091         if (libre == 0)
00092         {
00093             this.libre = false;
00094         }
00095         else
00096         {
00097             this.libre = true;
00098         }
00099     }
00100
00104     public void setLibre()
00105     {
00106         if (libre == true)
00107         {
00108             this.libre = false;
00109         }
00110         else
00111         {
00112             this.libre = true;
00113         }
00114     }
00115
00120     public void setSurface(int nouvelleSurface)
00121     {
00122         this.surface = nouvelleSurface;
00123     }
00124
00129     public void setConfort(int nouveauConfort)
00130     {
00131         this.confort = nouveauConfort;
00132     }
00133
00138     public void setTemperature(float temperature)
00139     {
00140         this.temperature = temperature;
00141     }
00142
00147     public void setAdresseIP (String adresseIP)
00148     {
00149         this.adresseIP = adresseIP;
00150     }
00151
00156     public final String getEmplacement()
00157     {
00158         return emplacement;
00159     }
00160
00165     public final String getNom()
00166     {
00167         return nom;
00168     }
00169
00174     public final boolean getLibre()
00175     {
00176         return libre;
00177     }
00178
00183     public final String getLibreTrame()
00184     {
00185         if(libre == false)
00186         {
00187             return "0";
00188         }
00189         else
00190         {
00191             return "1";
00192         }
00193     }
00194
00199     public final String getLibreIHM()
00200     {
00201         if(libre == false)
00202         {
00203             return "occupée";
00204         }
00205         else
00206         {
00207             return "disponible";
00208         }
00209     }
00210
```

```
00215     public final int getSurface()
00216     {
00217         return surface;
00218     }
00219
00224     public final int getConfort()
00225     {
00226         return confort;
00227     }
00228
00233     public final String getConfortIHM()
00234     {
00235         String message = "";
00236         switch (confort)
00237         {
00238             case FROID:
00239                 message = "Confort : Froid";
00240                 break;
00241             case FRAIS:
00242                 message = "Confort : Frais";
00243                 break;
00244             case LEGEREMENTFRAIS:
00245                 message = "Confort : Légèrement frais";
00246                 break;
00247             case NEUTRE:
00248                 message = "Confort : Neutre";
00249                 break;
00250             case LEGEREMENTTIEDE:
00251                 message = "Confort : Légèrement tiède";
00252                 break;
00253             case TIEDE:
00254                 message = "Confort : Tiède";
00255                 break;
00256             case CHAUD:
00257                 message = "Confort : Chaud";
00258                 break;
00259         }
00260         return message;
00261     }
00262
00267     public final float getTemperature()
00268     {
00269         return temperature;
00270     }
00271
00276     public final String getDescription()
00277     {
00278         return description;
00279     }
00280
00285     public final String getAdresseIP()
00286     {
00287         return adresseIP;
00288     }
00289 }
```

8.15 Référence du fichier SalleActivity.java

Déclaration de la classe SalleActivity.

Classes

— class com.lasalle.meeting.SalleActivity
Déclaration de la classe SalleActivity.

Paquetages

— package com.lasalle.meeting

8.15.1 Description détaillée

Déclaration de la classe SalleActivity.

Auteur

Vincent DEVINE

Définition dans le fichier SalleActivity.java.

8.16 SalleActivity.java

```
00001 package com.lasalle.meeting;
00002
00003 import androidx.annotation.NonNull;
00004 import androidx.appcompat.app.AppCompatActivity;
00005 import androidx.swiperefreshlayout.widget.SwipeRefreshLayout;
00006
00007 import android.content.Intent;
00008 import android.graphics.Color;
00009 import android.os.Bundle;
00010 import android.util.Log;
00011 import android.view.MenuItem;
00012 import android.view.View;
00013 import android.widget.Button;
00014 import android.widget.TextView;
00015 import android.widget.Toast;
00016
00017 import com.google.android.material.bottomnavigation.BottomNavigationView;
00018
00029 public class SalleActivity extends AppCompatActivity
00030 {
00034     private static final String TAG = "SalleActivity";
00035
00038     private Button boutonChangeEtat;
00039     private TextView textNom;
00040     private TextView textEmplacement;
00041     private TextView textLibre;
00042     private TextView textConfort;
00043     private TextView textSurface;
00044     private TextView textTemperature;
00045     private TextView textDescription;
00046     private SwipeRefreshLayout swipeRefreshLayout;
00047     private BottomNavigationView bottomNavigationView;
00048
00051     private Salle maSalle = null;
00052     private Communication communication = null;
00053
00059     @Override
00060     protected void onCreate(Bundle savedInstanceState)
00061     {
00062         super.onCreate(savedInstanceState);
00063         Log.d(TAG, "onCreate()");
00064
00065         setContentView(R.layout.activity_salle);
00066
00067         Intent intent = getIntent();
00068         maSalle = (Salle)intent.getSerializableExtra("Salle");
00069
00070         if(maSalle == null)
00071             Log.d(TAG, "Salle : " + maSalle.getNom());
00072
00073         communication = MainActivity.getCommunication();
00074
00075         initialiserRessourceIHM();
00076         setBoutonChangeEtat();
00077         afficherInformationSalle();
00078         setListener();
00079     }
00080
00085     public void setBoutonChangeEtat()
00086     {
00087         if (maSalle.getLibre() == true)
00088         {
00089             boutonChangeEtat.setText("Prendre");
00090             boutonChangeEtat.setBackgroundColor(Color.rgb(39,195,26));
00091         }
00092         else
00093         {
00094             boutonChangeEtat.setText("Libérer");
00095             boutonChangeEtat.setBackgroundColor(Color.rgb(222,55,25));
00096         }
00097     }
00098
00103     public void afficherInformationSalle()
00104     {
00105         textNom.setText(maSalle.getNom());
00106         textNom.setTextSize(35);
00107         textDescription.setText(maSalle.getDescription());
00108         textDescription.setTextSize(25);
00109         textEmplacement.setText(maSalle.getEmplacement());
00110         textEmplacement.setTextSize(25);
00111         textConfort.setText(maSalle.getConfortIHM());
00112         textConfort.setTextSize(25);
00113         textSurface.setText(Integer.toString(maSalle.getSurface()) + " m²");
00114         textSurface.setTextSize(25);
00115         textLibre.setTextSize(25);
00116         if (maSalle.getLibre() == true)
00117         {
00118             textLibre.setText("État : Libre");
```

```
00119         }
00120         else
00121         {
00122             textLibre.setText("État : Occupée");
00123         }
00124         textTemperature.setText(Float.toString(maSalle.getTemperature()) + " °C");
00125         textTemperature.setTextSize(25);
00126     }
00127
00132     public void setListener()
00133     {
00134
00135         boutonChangeEtat.setOnClickListener(
00136                 new View.OnClickListener()
00137                 {
00138                     @Override
00139                     public void onClick(View v)
00140                     {
00141                         maSalle.setLibre();
00142                         setBoutonChangeEtat();
00143                         afficherInformationSalle();
00144
00145                         if(communication != null)
00146                         {
00147                             communication.envoyer("$SET;3;" + maSalle.
      getLibreTrame() + "\r\n", maSalle.getAdresseIP());
00148                         }
00149                     }
00150                 }
00151         );
00152
00153         swipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener()
00154         {
00155             @Override
00156             public void onRefresh()
00157             {
00158                 afficherInformationSalle();
00159                 swipeRefreshLayout.setRefreshing(false);
00160             }
00161         });
00162
00163         bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.
      OnNavigationItemSelectedListener()
00164         {
00165             @Override
00166             public boolean onNavigationItemSelected(@NonNull MenuItem item)
00167             {
00168                 switch (item.getItemId())
00169                 {
00170                     case R.id.Salle:
00171                         Toast.makeText(getApplicationContext(), "Salle", Toast.LENGTH_SHORT).show();
00172                         return true;
00173                     case R.id.Favoris:
00174                         Toast.makeText(getApplicationContext(), "La fonctionnalité favoris n'est pas encore
       disponible !", Toast.LENGTH_SHORT).show();
00175                         return true;
00176                     case R.id.Rechercher:
00177                         Toast.makeText(getApplicationContext(), "La fonctionnalité rechercher n'est pas
       encore disponible !", Toast.LENGTH_SHORT).show();
00178                         return true;
00179                 }
00180                 return false;
00181             }
00182         });
00183     }
00184
00189     public void initialiserRessourceIHM()
00190     {
00191         boutonChangeEtat = (Button)findViewById(R.id.buttonChangeEtat);
00192         textNom = (TextView)findViewById(R.id.textViewNom);
00193         textEmplacement = (TextView)findViewById(R.id.textViewEmplacement);
00194         textLibre = (TextView)findViewById(R.id.textViewLibre);
00195         textConfort = (TextView)findViewById(R.id.textViewConfort);
00196         textSurface = (TextView)findViewById(R.id.textViewSurface);
00197         textTemperature = (TextView)findViewById(R.id.textViewTemperature);
00198         textDescription = (TextView)findViewById(R.id.textViewDescription);
00199         swipeRefreshLayout = (SwipeRefreshLayout)findViewById(R.id.swipeRefreshLayout);
00200         bottomNavigationView = (BottomNavigationView) findViewById(R.id.bottomNavigationView);
00201     }
00202
00207     @Override
00208     public void finish()
00209     {
00210         Log.d(TAG, "finish()");
00211
00212         Intent intent = new Intent();
00213
00214         intent.putExtra("salle", maSalle);
00215
00216         setResult(RESULT_OK, intent);
00217         super.finish();
```

```
00218     }
00219 }
```

8.17 Référence du fichier SalleAdapter.java

Déclaration de la classe SalleAdapter.

Classes

— class com.lasalle.meeting.SalleAdapter
Déclaration de la classe SalleAdapter.

Paquetages

— package com.lasalle.meeting

8.17.1 Description détaillée

Déclaration de la classe SalleAdapter.

Auteur

Vincent DEVINE

Définition dans le fichier SalleAdapter.java.

8.18 SalleAdapter.java

```
00001 package com.lasalle.meeting;
00002
00003 import android.util.Log;
00004 import android.view.LayoutInflater;
00005 import android.view.View;
00006 import android.view.ViewGroup;
00007
00008 import androidx.annotation.NonNull;
00009 import androidx.recyclerview.widget.RecyclerView;
00010
00011 import java.util.Vector;
00012
00023 public class SalleAdapter extends RecyclerView.Adapter<SalleViewHolder>
00024 {
00028     private static final String TAG = "SalleAdapter";
00029
00032     private Vector<Salle> mesSalles = null;
00033
00038     public SalleAdapter(Vector<Salle> mesSalles)
00039     {
00040         Log.d(TAG, "SalleAdapter (Vector<Salle>)");
00041
00042         if (mesSalles != null) {
00043             this.mesSalles = mesSalles;
00044         }
00045     }
00046
00052     @NonNull
00053     @Override
00054     public SalleViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
      int viewType)
00055     {
00056         Log.d(TAG, "SalleViewHolder onCreateViewHolder()");
00057         LayoutInflater inflater = LayoutInflater.from(parent.getContext());
00058         View view = inflater.inflate(R.layout.salle, parent, false);
00059         return new SalleViewHolder(view);
00060     }
00061
```

```
00067     @Override
00068     public void onBindViewHolder(@NonNull SalleViewHolder holder, int
      position)
00069     {
00070         Log.d(TAG, "onBindViewHolder()");
00071         Salle salle = mesSalles.get(position);
00072         holder.afficher(salle);
00073     }
00074
00079     @Override
00080     public int getItemCount() {
00081         Log.d(TAG, "getItemCount()");
00082         if (mesSalles != null)
00083             return mesSalles.size();
00084         return 0;
00085     }
00086 }
```

8.19 Référence du fichier SalleViewHolder.java

Déclaration de la classe SalleViewHolder.

Classes

— class com.lasalle.meeting.SalleViewHolder
Déclaration de la classe SalleViewHolder.

Paquetages

— package com.lasalle.meeting

8.19.1 Description détaillée

Déclaration de la classe SalleViewHolder.

Auteur

Vincent DEVINE

Définition dans le fichier SalleViewHolder.java.

8.20 SalleViewHolder.java

```
00001 package com.lasalle.meeting;
00002
00003 import android.app.AlertDialog;
00004 import android.content.Context;
00005 import android.content.Intent;
00006 import android.util.Log;
00007 import android.view.View;
00008 import android.widget.ImageView;
00009 import android.widget.TextView;
00010
00011 import java.io.Serializable;
00012
00013 import androidx.recyclerview.widget.RecyclerView;
00014
00025 public class SalleViewHolder extends RecyclerView.ViewHolder
00026 {
00030     private static final String TAG = "SalleViewHolder";
00031
00034     private TextView nomSalle;
00035     private TextView confortSalle;
00036     private TextView disponibiliteSalle;
00037     private ImageView imageDisponibiliteSalle;
00038
```

```
00041     private Context context;
00042     private Salle salle;
00043
00044
00049     public SalleViewHolder(final View itemView)
00050     {
00051         super(itemView);
00052
00053         Log.d(TAG, "SalleViewHolder(final View itemView)");
00054
00055         nomSalle= ((TextView)itemView.findViewById(R.id.nomSalle));
00056         confortSalle = ((TextView)itemView.findViewById(R.id.confortSalle));
00057         disponibiliteSalle = ((TextView)itemView.findViewById(R.id.disponibiliteSalle));
00058         imageDisponibiliteSalle = ((ImageView)itemView.findViewById(R.id.imageDisponibiliteSalle));
00059         context = itemView.getContext();
00060
00061         itemView.setOnClickListener(new View.OnClickListener()
00062         {
00063             @Override
00064             public void onClick(View view)
00065             {
00066                 Intent intent = new Intent(context, SalleActivity.class);
00067                 intent.putExtra("Salle", (Serializable) salle);
00068                 context.startActivity(intent);
00069             }
00070         });
00071     }
00072
00077     public void afficher(Salle salle)
00078     {
00079         Log.d(TAG, "afficher ()");
00080
00081         this.salle= salle;
00082         nomSalle.setText(salle.getNom());
00083         nomSalle.setTextSize(15);
00084         confortSalle.setText(salle.getConfortIHM());
00085         confortSalle.setTextSize(15);
00086         disponibiliteSalle.setText("La salle est "+ salle.getLibreIHM());
00087         disponibiliteSalle.setTextSize(15);
00088
00089         if(salle.getLibre())
00090         {
00091             imageDisponibiliteSalle.setImageResource(R.drawable.rond_vert);
00092         }
00093         else
00094         {
00095             imageDisponibiliteSalle.setImageResource(R.drawable.rond_rouge);
00096         }
00097     }
00098 }
```

Index