Projet Bee-Honey't (Mobile)

version 0.2

BTS SNIR LaSalle Avignon 2020

Table des matières

1 Le projet

Système autonome permettant de connaître à distance certains paramètres d'une ruche afin d'assurer son suivi et d'évaluer la santé des abeilles.

Version : PC (*mobile*)

1.1 Table des matières

1.2 Informations

Auteur

Ethan Villesseche villesseche.ethan@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/bee-honey-t

2 Changelog

r38 | evillesseche | 2020-04-03 10 :42 :19 +0200 (ven. 03 avril 2020) | 1 ligne

diagramme de déploiement

r36 | evillesseche | 2020-04-03 05 :49 :12 +0200 (ven. 03 avril 2020) | 1 ligne

ajout de BOUML

r34 | evillesseche | 2020-04-03 03 :36 :37 +0200 (ven. 03 avril 2020) | 1 ligne

ajout de la gestion du port , affichage des donné , gestion du device id

r33 | tvaira | 2020-04-02 19 :06 :15 +0200 (jeu. 02 avril 2020) | 1 ligne

Réception des messages MQTT

r29 | tvaira | 2020-04-02 11 :40 :16 +0200 (jeu. 02 avril 2020) | 1 ligne

Fichiers inutiles

r28 | evillesseche | 2020-04-02 07 :46 :05 +0200 (jeu. 02 avril 2020) | 1 ligne

modification de la connections au TTN es de abonnement au ruche

r27 | evillesseche | 2020-04-02 07 :17 :14 +0200 (jeu. 02 avril 2020) | 1 ligne

Ajout de methode pour decoder le JSON et s'aboner a un device

r18 | tvaira | 2020-03-20 11 :47 :50 +0100 (ven. 20 mars 2020) | 1 ligne

Intégration de la base de MQTT

r14 | tvaira | 2020-03-18 09 :05 :08 +0100 (mer. 18 mars 2020) | 1 ligne

Revision de code

r13 | tvaira | 2020-03-18 08 :08 :52 +0100 (mer. 18 mars 2020) | 1 ligne

fichier à ignorer

r9 | evillesseche | 2020-03-14 21 :19 :16 +0100 (sam. 14 mars 2020) | 1 ligne

Ajout de circleimageview et de click lisenner sur les recyclerView

r7 | evillesseche | 2020-03-13 16 :30 :12 +0100 (ven. 13 mars 2020) | 2 lignes

Mise en place d'un recyclerView

3 README

3.1 Projet

3.1.1 Présentation

Système autonome permettant de connaître à distance certains paramètres d'une ruche afin d'assurer son suivi et d'évaluer la santé des abeilles.

Version : PC (*mobile*)

3.1.2 Base de données

3.1.3 Recette de l'application PC (mobile)

- Consulter les données d'une ruche
- Recevoir les données actuelles des ruches (MQTT/Json)

3.1.4 Informations

Auteur

Ethan Villesseche villesseche.ethan@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/bee-honey-t

4 A propos

Auteur

Ethan Villesseche villesseche.ethan@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/bee-honey-t

5 Licence GPL

6 Liste des choses à faire

Membre com.lasalle.beehoneyt.CommunicationMQTT.serverUri
Prévoir autre chose pour configurer/stocker les paramètres de connexion TTN

Membre com.lasalle.beehoneyt.MainActivity.communiquerTTN (final String deviceID)
Décoder les messages et mettre à jour l'affichage

Membre com.lasalle.beehoneyt.MainActivity.onResume ()
Il faut rappeler connecterTTN

Membre com.lasalle.beehoneyt.MainActivity.recupererRuches ()
Récupérer les ruches à partir de la base de données SQLite et vérifier si il y a des nouvelles ruches

Membre com.lasalle.beehoneyt.RucheActivity.formaterHorodatge (String horodatage)
Finaliser le formatage de l'horodatage

7 Documentation des espaces de nommage

7.1 Paquetage com

Paquetages

— package lasalle

7.2 Paquetage com.lasalle

Paquetages

— package beehoneyt

7.3 Paquetage com.lasalle.beehoneyt

Classes

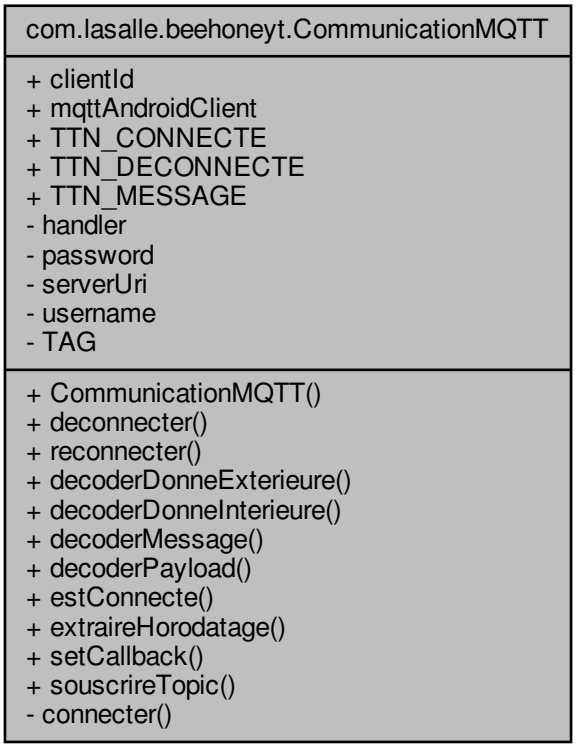
— class CommunicationMQTT
Déclaration de la classe CommunicationMQTT.
— class MainActivity
Déclaration de la classe MainActivity.
— class MenuActivity
Déclaration de la classe MenuActivity.
— class Ruche
Déclaration de la classe Ruche.
— class RucheActivity
Déclaration de la classe RucheActivity.
— class RucheAdapter
Déclaration de la classe RucheAdapter.
— class RuchesViewHolder
Déclaration de la classe RuchesViewHolder.

8 Documentation des classes

8.1 Référence de la classe com.lasalle.beehoneyt.CommunicationMQTT

Déclaration de la classe CommunicationMQTT.

Graphe de collaboration de com.lasalle.beehoneyt.CommunicationMQTT :



Fonctions membres publiques

- CommunicationMQTT (Context context, final Handler handler)
  Constructeur de la classe CommunicationMQTT.
- void deconnecter ()
  deconetion du ttn
- void reconnecter ()
  reconnection au ttn

Fonctions membres publiques statiques

- static void decoderDonneExterieure (String payload, Ruche ruche)
  decode les donné de l'exterieure de la ruche
- static void decoderDonneInterieure (String payload, Ruche ruche)
  decode les donné de l'interieur de la ruche
- static void decoderMessage (String message, Ruche ruche)
  decode le message recu
- static void decoderPayload (int port, String payload, Ruche ruche)
  d'ecode le payload

- static boolean estConnecte ()
  boolaine retourne si le ttn est connecter
- static String extraireHorodatage (String message)
  extrait l'horodatage
- static void setCallback (MqttCallbackExtended callback)
- static boolean souscrireTopic (String topic)
  S'abone a un device.

Attributs publics statiques

- static String clientId = "mes_ruches"
  Application ID.
- static MqttAndroidClient mqttAndroidClient
- static final int TTN_CONNECTE = 1
- static final int TTN_DECONNECTE = 2
- static final int TTN_MESSAGE = 3

Fonctions membres privées

- void connecter ()
  connection au ttn

Attributs privés

- Handler handler = null
- String password = "ttn-account-v2.vC-aqMRnLLzGkNjODWgy81kLqzxBPAT8_mE-L7U2C_w"
  mot de passe TTN
- String serverUri = "tcp ://eu.thethings.network :1883"
  lien vers TTN
- String username = "mes_ruches"
  nom d'utilisateur

Attributs privés statiques

- static final String TAG = "CommunicationMQTT"

8.1.1 Description détaillée

Déclaration de la classe CommunicationMQTT.

Définition à la ligne 32 du fichier CommunicationMQTT.java.

8.1.2 Documentation des constructeurs et destructeur

8.1.2.1 CommunicationMQTT()

```
com.lasalle.beehoneyt.CommunicationMQTT.CommunicationMQTT (
            Context context,
            final Handler handler )
```

Constructeur de la classe CommunicationMQTT.

Paramètres

context	

Définition à la ligne 63 du fichier CommunicationMQTT.java.

Références com.lasalle.beehoneyt.CommunicationMQTT.connecter(), et com.lasalle.beehoneyt.CommunicationMQTT.handler.

```
00064     {
00065         Log.v(TAG, "CommunicationMQTT() clientId = " + clientId);
00066         this.handler = handler;
00067         mqttAndroidClient = new MqttAndroidClient(context,
      serverUri, clientId);
00068         mqttAndroidClient.setCallback(new MqttCallbackExtended()
00069         {
00070             @Override
00071             public void connectComplete(boolean b, String s)
00072             {
00073                 Log.w(TAG, "connectComplete() serverUri = " + s + " connecte = " +
      mqttAndroidClient.isConnected());
00074                 Message msg = Message.obtain();
00075                 Bundle bundle = new Bundle();
00076                 bundle.putInt("etat", TTN_CONNECTE);
00077                 msg.setData(bundle);
00078                 handler.sendMessage(msg);
00079             }
00080
00081             @Override
00082             public void connectionLost(Throwable throwable)
00083             {
00084                 Log.w(TAG, "connectionLost()");
00085                 Message msg = Message.obtain();
00086                 Bundle b = new Bundle();
00087                 b.putInt("etat", TTN_DECONNECTE);
00088                 msg.setData(b);
00089                 handler.sendMessage(msg);
00090             }
00091
00092             @Override
00093             public void messageArrived(String topic, MqttMessage mqttMessage) throws Exception
00094             {
00095                 Log.w(TAG, "messageArrived() topic = " + topic + " message = " + mqttMessage.toString())
      ;
00096                 Message msg = Message.obtain();
00097                 Bundle b = new Bundle();
00098                 b.putInt("etat", TTN_MESSAGE);
00099                 b.putString("topic", topic);
00100                 b.putString("message", mqttMessage.toString());
00101                 msg.setData(b);
00102                 handler.sendMessage(msg);
00103             }
00104
00105             @Override
00106             public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken)
00107             {
00108                 Log.w(TAG, "deliveryComplete()");
00109             }
00110         });
00111
00112         connecter();
00113     }
```

8.1.3 Documentation des fonctions membres

8.1.3.1 connecter()

`com.lasalle.beehoneyt.CommunicationMQTT.connecter ( )  [private]`

connection au ttn

Définition à la ligne 131 du fichier CommunicationMQTT.java.

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.CommunicationMQTT(), et com.lasalle.beehoneyt.CommunicationMQ↩TT.reconnecter().

```
00132     {
00133         MqttConnectOptions mqttConnectOptions = new MqttConnectOptions();
00134         mqttConnectOptions.setAutomaticReconnect(true);
00135         mqttConnectOptions.setCleanSession(false);
00136         mqttConnectOptions.setUserName(username);
00137         mqttConnectOptions.setPassword(password.toCharArray());
00138 
00139         try
00140         {
00141             Log.d(TAG, "connecter() serverUri = " + serverUri + " clientId = " +
      clientId);
00142             mqttAndroidClient.connect(mqttConnectOptions, null, new IMqttActionListener()
00143             {
00144                 @Override
00145                 public void onSuccess(IMqttToken asyncActionToken)
00146                 {
00147                     DisconnectedBufferOptions disconnectedBufferOptions = new DisconnectedBufferOptions();
00148                     disconnectedBufferOptions.setBufferEnabled(true);
00149                     disconnectedBufferOptions.setBufferSize(100);
00150                     disconnectedBufferOptions.setPersistBuffer(false);
00151                     disconnectedBufferOptions.setDeleteOldestMessages(false);
00152                     mqttAndroidClient.setBufferOpts(disconnectedBufferOptions);
00153                     Log.d(TAG, "onSuccess() serverUri = " + serverUri + " clientId = " +
      clientId);
00154                 }
00155 
00156                 @Override
00157                 public void onFailure(IMqttToken asyncActionToken, Throwable exception)
00158                 {
00159                     Log.d(TAG, "onFailure() serverUri = " + serverUri + " clientId = " +
      clientId + " exception = " + exception.toString());
00160                 }
00161             });
00162         }
00163         catch (MqttException ex)
00164         {
00165             ex.printStackTrace();
00166         }
00167     }
```

8.1.3.2 decoderDonneExterieure()

```
static void com.lasalle.beehoneyt.CommunicationMQTT.decoderDonneExterieure (
            String payload,
            Ruche ruche )  [static]
```

decode les donné de l'exterieure de la ruche

Paramètres

payload	
ruche	

Définition à la ligne 366 du fichier CommunicationMQTT.java.

Références com.lasalle.beehoneyt.RucheActivity.afficherEnsoleillement(), com.lasalle.beehoneyt.RucheActivity.afficherHumiditer↩Exterieure(), com.lasalle.beehoneyt.RucheActivity.afficherPoids(), com.lasalle.beehoneyt.RucheActivity.afficherPression(), com.↩lasalle.beehoneyt.RucheActivity.afficherTemperatureExterieure(), et com.lasalle.beehoneyt.Ruche.setPoids().

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.decoderPayload().

```
00367     {
00368         try {
00369             JSONObject json = null;
00370             json = new JSONObject(payload);
00371 
00372             Iterator<String> it = null;
00373             it = json.keys();
00374             while (it.hasNext()) {
00375                 String cle = it.next();
00376                 if (cle.equals("temperature"))
```

```
00377                 {
00378                     Log.i(TAG, "decoderDonneExterieure() temperature = " + json.getString(cle));
00379                     RucheActivity.afficherTemperatureExterieure(json.getString(cle));
00380                 }
00381                 else if (cle.equals("humidite"))
00382                 {
00383                     Log.i(TAG, "decoderDonneExterieure() humidite = " + json.getString(cle));
00384                     RucheActivity.afficherHumiditerExterieure(json.getString(cle));
00385                 }
00386                 else if (cle.equals("ensoleillement"))
00387                 {
00388                     Log.i(TAG, "decoderDonneExterieure() ensoleillement = " + json.getString(cle));
00389                     RucheActivity.afficherEnsoleillement(json.getString(cle));
00390                 }
00391                 else if (cle.equals("pression"))
00392                 {
00393                     Log.i(TAG, "decoderDonneExterieure() pression = " + json.getString(cle));
00394                     RucheActivity.afficherPression(json.getString(cle));
00395                 }
00396                 else if (cle.equals("poids"))
00397                 {
00398                     double poids = json.getDouble(cle);
00399                     poids = poids * 0.01;
00400                     Log.i(TAG, "decoderDonneExterieure() poids = " + json.getInt(cle));
00401                     ruche.setPoids(json.getInt(cle));
00402                     RucheActivity.afficherPoids(json.getInt(cle));
00403                 }
00404             }
00405         }
00406         catch (JSONException e)
00407         {
00408             e.printStackTrace();
00409         }
00410     }
```

8.1.3.3 decoderDonneInterieure()

```
static void com.lasalle.beehoneyt.CommunicationMQTT.decoderDonneInterieure (
            String payload,
            Ruche ruche )  [static]
```

decode les donné de l'interieur de la ruche

Paramètres

payload	
ruche	

Définition à la ligne 331 du fichier CommunicationMQTT.java.

Références com.lasalle.beehoneyt.RucheActivity.afficherHumiditeInterieure(), et com.lasalle.beehoneyt.RucheActivity.afficher↩ TemperatureInterieure().

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.decoderPayload().

```
00332     {
00333         try
00334         {
00335             JSONObject json = null;
00336             json = new JSONObject(payload);
00337 
00338             Iterator<String> it = null;
00339             it = json.keys();
00340             while (it.hasNext())
00341             {
00342                 String cle = it.next();
00343                 if (cle.equals("temperature"))
00344                 {
00345                     Log.i(TAG, "decoderDonneInterieure() temperature = " + json.getString(cle));
00346                     RucheActivity.afficherTemperatureInterieure(json.getString(cle));
00347                 }
```

```
00348                 else if (cle.equals("humidite"))
00349                 {
00350                     Log.i(TAG, "decoderDonneInterieure() humidite = " + json.getString(cle));
00351                     RucheActivity.afficherHumiditeInterieure(json.getString(cle));
00352                 }
00353             }
00354         }
00355         catch (JSONException e)
00356         {
00357             e.printStackTrace();
00358         }
00359     }
```

8.1.3.4 decoderMessage()

```
com.lasalle.beehoneyt.CommunicationMQTT.decoderMessage (
            String message,
            Ruche ruche )  [static]
```

decode le message recu

Paramètres

message	le message reçu
ruche	

Définition à la ligne 276 du fichier CommunicationMQTT.java.

Références com.lasalle.beehoneyt.CommunicationMQTT.decoderPayload().

Référencé par com.lasalle.beehoneyt.RucheActivity.communiquerTTN().

```
00277     {
00278         try
00279         {
00280             JSONObject json = null;
00281             Iterator<String> it = null;
00282 
00283             json = new JSONObject(message);
00284             int port = json.getInt("port");
00285 
00286             it = json.keys();
00287             while (it.hasNext())
00288             {
00289                 String cle = it.next();
00290                 Log.i(TAG, "decoderMessage() clé = " + cle);
00291                 Log.i(TAG, "decoderMessage() valeur = " + json.getString(cle));
00292                 //Log.i(TAG, "type = " + json.get(cle).getClass());
00293 
00294                 if (cle.equals("payload_fields"))
00295                 {
00296                     decoderPayload(port, json.getString(cle), ruche);
00297                 }
00298             }
00299         }
00300         catch (JSONException e)
00301         {
00302             e.printStackTrace();
00303         }
00304     }
```

8.1.3.5 decoderPayload()

```
static void com.lasalle.beehoneyt.CommunicationMQTT.decoderPayload (
            int port,
            String payload,
            Ruche ruche )  [static]
```

d'ecode le payload

Paramètres

port	
payload	
ruche	

Définition à la ligne 312 du fichier CommunicationMQTT.java.

Références com.lasalle.beehoneyt.CommunicationMQTT.decoderDonneExterieure(), et com.lasalle.beehoneyt.CommunicationMQTT.decoderDonneInterieure().

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.decoderMessage().

```
00313     {
00314
00315         Log.i(TAG, "decoderPayload() port = " + port );
00316         if ( port == 3)
00317         {
00318             decoderDonneInterieure(payload, ruche);
00319         }
00320         else
00321         {
00322             decoderDonneExterieure(payload, ruche);
00323         }
00324     }
```

8.1.3.6 deconnecter()

```
com.lasalle.beehoneyt.CommunicationMQTT.deconnecter ( )
```

deconetion du ttn

Définition à la ligne 187 du fichier CommunicationMQTT.java.

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.reconnecter().

```
00188     {
00189         Log.d(TAG, "deconnecter() serverUri = " + serverUri + " clientId = " +
      clientId);
00190         try
00191         {
00192             IMqttToken disconToken = mqttAndroidClient.disconnect();
00193             disconToken.setActionCallback(new IMqttActionListener()
00194             {
00195                 @Override
00196                 public void onSuccess(IMqttToken asyncActionToken)
00197                 {
00198                     Log.d(TAG, "onSuccess() serverUri = " + serverUri + " clientId = " +
      clientId);
00199                 }
00200
00201                 @Override
00202                 public void onFailure(IMqttToken asyncActionToken, Throwable exception)
00203                 {
00204                     Log.d(TAG, "onFailure() serverUri = " + serverUri + " clientId = " +
      clientId + " exception = " + exception.toString());
00205                 }
00206             });
00207         }
00208         catch (MqttException e)
00209         {
00210             e.printStackTrace();
00211         }
00212     }
```

8.1.3.7 estConnecte()

```
com.lasalle.beehoneyt.CommunicationMQTT.estConnecte ( )  [static]
```

boolaine retourne si le ttn est connecter

Définition à la ligne 219 du fichier CommunicationMQTT.java.

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.reconnecter().

```
00220     {
00221         Log.w(TAG, "estConnecte() " + mqttAndroidClient.isConnected());
00222 
00223         return mqttAndroidClient.isConnected();
00224     }
```

8.1.3.8 extraireHorodatage()

```
static String com.lasalle.beehoneyt.CommunicationMQTT.extraireHorodatage (
             String message )  [static]
```

extrait l'horodatage

Paramètres

message	

Définition à la ligne 416 du fichier CommunicationMQTT.java.

Référencé par com.lasalle.beehoneyt.RucheActivity.communiquerTTN().

```
00417     {
00418         String date = "";
00419 
00420         try
00421         {
00422             JSONObject json = null;
00423             json = new JSONObject(message);
00424 
00425             date = json.getJSONObject("metadata").getString("time");
00426             date = date.substring(0, 10) + " " + date.substring(11, 19);
00427             Log.d(TAG, "extraireHorodatage() time = " + json.getJSONObject("metadata").getString("time")
    );
00428             Log.d(TAG, "extraireHorodatage() horodatage = " + date);
00429 
00430         }
00431         catch (JSONException e)
00432         {
00433             e.printStackTrace();
00434         }
00435 
00436         return  date;
00437     }
```

8.1.3.9 reconnecter()

```
com.lasalle.beehoneyt.CommunicationMQTT.reconnecter ( )
```

reconnection au ttn

Définition à la ligne 174 du fichier CommunicationMQTT.java.

Références com.lasalle.beehoneyt.CommunicationMQTT.connecter(), com.lasalle.beehoneyt.CommunicationMQTT.deconnecter(), et com.lasalle.beehoneyt.CommunicationMQTT.estConnecte().

```
00175     {
00176         Log.w(TAG, "reconnecter ()");
00177         if (estConnecte())
00178             deconnecter();
00179         connecter();
00180     }
```

8.1.3.10 setCallback()

```
com.lasalle.beehoneyt.CommunicationMQTT.setCallback (
             MqttCallbackExtended callback )  [static]
```

Paramètres

callback	le retour

Définition à la ligne 121 du fichier CommunicationMQTT.java.

Référencé par com.lasalle.beehoneyt.RucheActivity.communiquerTTN(), et com.lasalle.beehoneyt.MainActivity.communiquerTTN().

```
00122     {
00123         mqttAndroidClient.setCallback(callback);
00124     }
```

8.1.3.11 souscrireTopic()

```
com.lasalle.beehoneyt.CommunicationMQTT.souscrireTopic (
             String topic )  [static]
```

S'abone a un device.

Paramètres

topic	le topic au quel s'aboner

Définition à la ligne 232 du fichier CommunicationMQTT.java.

Référencé par com.lasalle.beehoneyt.Ruche.souscrireTopic().

```
00233     {
00234         if(mqttAndroidClient == null && !mqttAndroidClient.isConnected())
00235         {
00236             return false;
00237         }
00238         final String subTopic = clientId + "/devices/" + topic + "/up";
00239         Log.w(TAG, "souscrireTopic() topic = " + subTopic);
00240         try
00241         {
00242             final boolean[] retour = {false};
00243             mqttAndroidClient.subscribe(subTopic, 0, null, new IMqttActionListener()
00244             {
00245                 @Override
00246                 public void onSuccess(IMqttToken asyncActionToken)
00247                 {
00248                     Log.w(TAG, "onSuccess() topic = " + subTopic);
00249                     retour[0] = true;
00250                 }
00251
```

```
00252                 @Override
00253                 public void onFailure(IMqttToken asyncActionToken, Throwable exception)
00254                 {
00255                     Log.w(TAG, "onFailure() topic = " + subTopic);
00256                     retour[0] = false;
00257                 }
00258             });
00259             return retour[0];
00260         }
00261         catch (MqttException ex)
00262         {
00263             Log.w(TAG, "Erreur topic = " + subTopic);
00264             ex.printStackTrace();
00265             return false;
00266         }
00267     }
```

8.1.4 Documentation des données membres

8.1.4.1 clientId

```
String com.lasalle.beehoneyt.CommunicationMQTT.clientId = "mes_ruches"  [static]
```

Application ID.

Définition à la ligne 53 du fichier CommunicationMQTT.java.

8.1.4.2 handler

```
Handler com.lasalle.beehoneyt.CommunicationMQTT.handler = null  [private]
```

Définition à la ligne 42 du fichier CommunicationMQTT.java.

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.CommunicationMQTT().

8.1.4.3 mqttAndroidClient

```
MqttAndroidClient com.lasalle.beehoneyt.CommunicationMQTT.mqttAndroidClient  [static]
```

Attributs

Définition à la ligne 41 du fichier CommunicationMQTT.java.

8.1.4.4 password

```
String com.lasalle.beehoneyt.CommunicationMQTT.password = "ttn-account-v2.vC-aqMRnLLzGkNjODWgy81kLqzxBPAT8_↩
mE-L7U2C_w"  [private]
```

mot de passe TTN

Définition à la ligne 55 du fichier CommunicationMQTT.java.

8.1.4.5 serverUri

```
String com.lasalle.beehoneyt.CommunicationMQTT.serverUri = "tcp://eu.thethings.network:1883"  [private]
```

lien vers TTN

A faire Prévoir autre chose pour configurer/stocker les paramètres de connexion TTN

Définition à la ligne 52 du fichier CommunicationMQTT.java.

8.1.4.6 TAG

```
final String com.lasalle.beehoneyt.CommunicationMQTT.TAG = "CommunicationMQTT"  [static], [private]
```

Constantes

Définition à la ligne 37 du fichier CommunicationMQTT.java.

8.1.4.7 TTN_CONNECTE

```
final int com.lasalle.beehoneyt.CommunicationMQTT.TTN_CONNECTE = 1  [static]
```

Constantes de communication avec l'activité

Définition à la ligne 46 du fichier CommunicationMQTT.java.

8.1.4.8 TTN_DECONNECTE

```
final int com.lasalle.beehoneyt.CommunicationMQTT.TTN_DECONNECTE = 2  [static]
```

Définition à la ligne 47 du fichier CommunicationMQTT.java.

8.1.4.9 TTN_MESSAGE

```
final int com.lasalle.beehoneyt.CommunicationMQTT.TTN_MESSAGE = 3  [static]
```

Définition à la ligne 48 du fichier CommunicationMQTT.java.

8.1.4.10 username

```
String com.lasalle.beehoneyt.CommunicationMQTT.username = "mes_ruches"  [private]
```

nom d'utilisateur

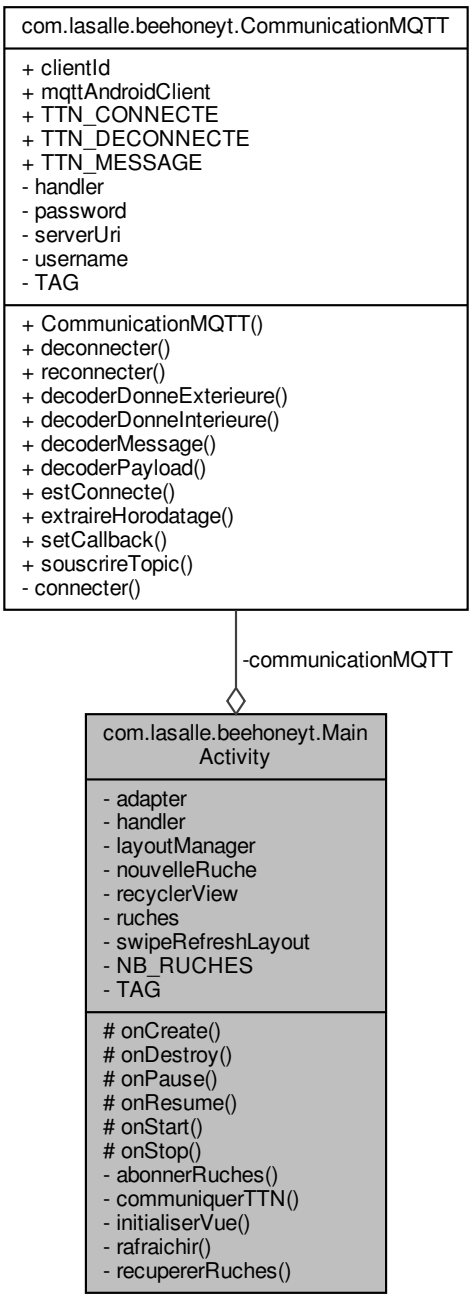
Définition à la ligne 54 du fichier CommunicationMQTT.java.

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

8.2 Référence de la classe com.lasalle.beehoneyt.MainActivity

Déclaration de la classe MainActivity.

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



Fonctions membres protégées

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

Méthode appelée à la destruction de l'application (après onStop() et détruite par le système Android)

— void onPause ()
Méthode appelée après qu'une boîte de dialogue s'est affichée (on reprend sur un onResume()) ou avant onStop() (activité plus visible)
— void onResume ()
Méthode appelée après onStart() ou après onPause()
— void onStart ()
Méthode appelée au démarrage après le onCreate() ou un restart après un onStop()
— void onStop ()
Méthode appelée lorsque l'activité n'est plus visible.

Fonctions membres privées

— void abonnerRuches ()
— void communiquerTTN (final String deviceID)
— void initialiserVue ()
— void rafraichir ()
Méthode pour rafraîchir la liste des ruches connues.
— void recupererRuches ()
Méthode pour récupérer les ruches.

Attributs privés

— RecyclerView.Adapter adapter
— CommunicationMQTT communicationMQTT
— final Handler handler
Handler de communication entre l'activité et la communication MQTT.
— RecyclerView.LayoutManager layoutManager
— boolean nouvelleRuche = false
indique si une nouvelle ruche a été récupérée
— RecyclerView recyclerView
— List< Ruche > ruches = new ArrayList<>()
la liste des ruches connues
— SwipeRefreshLayout swipeRefreshLayout

Attributs privés statiques

— static final int NB_RUCHES = 5
le nombre max. de ruches
— static final String TAG = "MainActivity"

8.2.1 Description détaillée

Déclaration de la classe MainActivity.

Logcat : MainActivity :|RucheActivity :|Ruche :|CommunicationMQTT :

Définition à la ligne 36 du fichier MainActivity.java.

8.2.2 Documentation des fonctions membres

8.2.2.1 abonnerRuches()

```
void com.lasalle.beehoneyt.MainActivity.abonnerRuches ( )  [private]
```

Définition à la ligne 259 du fichier MainActivity.java.

Références com.lasalle.beehoneyt.MainActivity.communiquerTTN().

```
00260     {
00261         for (int i = 0; i < ruches.size(); i++)
00262         {
00263             ruches.get(i).souscrireTopic();
00264             communiquerTTN(ruches.get(i).getDeviceID());
00265         }
00266     }
```

8.2.2.2 communiquerTTN()

```
void com.lasalle.beehoneyt.MainActivity.communiquerTTN (
            final String deviceID )  [private]
```

A faire Décoder les messages et mettre à jour l'affichage

Définition à la ligne 212 du fichier MainActivity.java.

Références com.lasalle.beehoneyt.MainActivity.rafraichir(), et com.lasalle.beehoneyt.CommunicationMQTT.setCallback().

Référencé par com.lasalle.beehoneyt.MainActivity.abonnerRuches().

```
00213     {
00214         Log.d(TAG, "communiquerTTN() deviceID = " + deviceID);
00215         CommunicationMQTT.setCallback(new MqttCallbackExtended()
00216         {
00217             @Override
00218             public void connectComplete(boolean b, String s)
00219             {
00220             }
00221             @Override
00222             public void connectionLost(Throwable throwable)
00223             {
00224             }
00225             @Override
00226             public void messageArrived(String topic, MqttMessage mqttMessage) throws Exception
00227             {
00228                 Log.d(TAG, "communiquerTTN() topic = " + topic);
00229                 Log.d(TAG, "communiquerTTN() mqttMessage = " + mqttMessage.toString());
00233                 // Juste pour tester (à retirer rapidement) !
00234                 JSONObject json = new JSONObject(mqttMessage.toString());
00235                 String deviceID = json.getString("dev_id");
00236                 for (int i = 0; i < ruches.size(); i++)
00237                 {
00238                     if(ruches.get(i).getDeviceID().equals(deviceID))
00239                     {
00240                         ruches.get(i).setInfos("Message reçu !");
00241                     }
00242                     else
00243                     {
00244                         ruches.get(i).setInfos("Simulateur");
00245                     }
00246                 }
00247                 // fin du test
00248
00249                 rafraichir();
00250             }
00251
00252             @Override
00253             public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken)
00254             {
00255             }
00256         });
00257     }
```

8.2.2.3 initialiserVue()

```
void com.lasalle.beehoneyt.MainActivity.initialiserVue ( )  [private]
```

Définition à la ligne 137 du fichier MainActivity.java.

Références com.lasalle.beehoneyt.MainActivity.adapter, com.lasalle.beehoneyt.MainActivity.layoutManager, et com.lasalle.beehoneyt.MainActivity.recupererRuches().

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

```
00138     {
00139         Log.d(TAG, "initialiserVue()");
00140         swipeRefreshLayout = (SwipeRefreshLayout)findViewById(R.id.swipeRefreshLayout);
00141         swipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener(
      )
00142         {
00143             @Override
00144             public void onRefresh()
00145             {
00146                 recupererRuches();
00147             }
00148         });
00149
00150         recyclerView = (RecyclerView) findViewById(R.id.listeRuche);
00151         recyclerView.setHasFixedSize(true);
00152
00153         layoutManager = new LinearLayoutManager(this);
00154         recyclerView.setLayoutManager(layoutManager);
00155
00156         adapter = new RucheAdapter(this, ruches);
00157         recyclerView.setAdapter(adapter);
00158     }
```

8.2.2.4 onCreate()

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

Méthode appelée à la création de l'activité MainActivity.

Paramètres

savedInstanceState	

Définition à la ligne 66 du fichier MainActivity.java.

Références com.lasalle.beehoneyt.MainActivity.handler, com.lasalle.beehoneyt.MainActivity.initialiserVue(), et com.lasalle.beehoneyt.MainActivity.recupererRuches().

```
00067     {
00068         super.onCreate(savedInstanceState);
00069         setContentView(R.layout.activity_main);
00070         Log.d(TAG, "onCreate()");
00071
00072         communicationMQTT = new CommunicationMQTT(getApplicationContext(),
      handler);
00073
00074         initialiserVue();
00075
00076         nouvelleRuche = true;
00077         recupererRuches();
00078     }
```

8.2.2.5 onDestroy()

```
void com.lasalle.beehoneyt.MainActivity.onDestroy ( )  [protected]
```

Méthode appelée à la destruction de l'application (après onStop() et détruite par le système Android)

Définition à la ligne 130 du fichier MainActivity.java.

```
00131     {
00132         super.onDestroy();
00133         Log.i(TAG,"onDestroy()");
00134 
00135     }
```

8.2.2.6 onPause()

```
void com.lasalle.beehoneyt.MainActivity.onPause ( )  [protected]
```

Méthode appelée après qu'une boîte de dialogue s'est affichée (on reprend sur un onResume()) ou avant onStop() (activité plus visible)

Définition à la ligne 108 du fichier MainActivity.java.

```
00109     {
00110         super.onPause();
00111         Log.i(TAG,"onPause()");
00112 
00113     }
```

8.2.2.7 onResume()

```
void com.lasalle.beehoneyt.MainActivity.onResume ( )  [protected]
```

Méthode appelée après onStart() ou après onPause()

A faire Il faut rappeler connecterTTN

Définition à la ligne 94 du fichier MainActivity.java.

```
00095     {
00096         super.onResume();
00097         Log.i(TAG,"onResume()");
00102     }
```

8.2.2.8 onStart()

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

Méthode appelée au démarrage après le onCreate() ou un restart après un onStop()

Définition à la ligne 84 du fichier MainActivity.java.

```
00085     {
00086         super.onStart();
00087         Log.i(TAG,"onStart()");
00088     }
```

8.2.2.9 onStop()

`void com.lasalle.beehoneyt.MainActivity.onStop ( ) [protected]`

Méthode appelée lorsque l'activité n'est plus visible.

Définition à la ligne 119 du fichier MainActivity.java.

```
00120     {
00121         super.onStop();
00122         Log.i(TAG,"onStop()");
00123
00124     }
```

8.2.2.10 rafraichir()

`com.lasalle.beehoneyt.MainActivity.rafraichir ( ) [private]`

Méthode pour rafraîchir la liste des ruches connues.

Définition à la ligne 206 du fichier MainActivity.java.

Références com.lasalle.beehoneyt.MainActivity.adapter.

Référencé par com.lasalle.beehoneyt.MainActivity.communiquerTTN(), et com.lasalle.beehoneyt.MainActivity.recupererRuches().

```
00207     {
00208         swipeRefreshLayout.setRefreshing(false);
00209         adapter.notifyDataSetChanged();
00210     }
```

8.2.2.11 recupererRuches()

`com.lasalle.beehoneyt.MainActivity.recupererRuches ( ) [private]`

Méthode pour récupérer les ruches.

A faire Récupérer les ruches à partir de la base de données SQLite et vérifier si il y a des nouvelles ruches

Définition à la ligne 165 du fichier MainActivity.java.

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

Référencé par com.lasalle.beehoneyt.MainActivity.initialiserVue(), et com.lasalle.beehoneyt.MainActivity.onCreate().

```
00166     {
00167         Log.d(TAG, "recupererRuches()");
00171         if(nouvelleRuche)
00172         {
00173             nouvelleRuche = false;
00174
00175             // Pour les tests
00176             List<Ruche> listeRuches = Arrays.asList(
00177                     new Ruche("Ruche 1", "ruche_1"),
00178                     new Ruche("Ruche 2", "ruche_2"),
00179                     new Ruche("Ruche 3", "ruche_3"),
00180                     new Ruche("Ruche 4", "ruche_3")
00181             );
00182
00183             listeRuches.get(0).setInfos("Simulateur");
00184             listeRuches.get(1).setPoids(35);
00185             listeRuches.get(1).setInfos("Simulateur");
00186             listeRuches.get(2).setPoids(32);
00187             listeRuches.get(2).setInfos("Simulateur");
00188             listeRuches.get(3).setPoids(40);
00189             listeRuches.get(3).setInfos("Simulateur");
00190
00191             ruches.clear();
00192             for (int i = 0; i < listeRuches.size(); i++)
00193             {
00194                 ruches.add(listeRuches.get(i));
00195             }
00196
00197             rafraichir();
00198         }
00199     }
```

8.2.3 Documentation des données membres

8.2.3.1 adapter

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

Définition à la ligne 55 du fichier MainActivity.java.

Référencé par com.lasalle.beehoneyt.MainActivity.initialiserVue(), et com.lasalle.beehoneyt.MainActivity.rafraichir().

8.2.3.2 communicationMQTT

```
CommunicationMQTT com.lasalle.beehoneyt.MainActivity.communicationMQTT  [private]
```

Définition à la ligne 49 du fichier MainActivity.java.

8.2.3.3 handler

```
final Handler com.lasalle.beehoneyt.MainActivity.handler  [private]
```

Valeur initiale :

```
= new Handler()
    {
        @Override
        public void handleMessage(Message msg)
        {
            super.handleMessage(msg);

            Bundle bundle = msg.getData();

            switch(bundle.getInt("etat"))
            {
                case CommunicationMQTT.TTN_CONNECTE:
                    Log.d(TAG, "handleMessage() TTN connecté");
                    abonnerRuches();
                    break;
                case CommunicationMQTT.TTN_DECONNECTE:
                    Log.d(TAG, "handleMessage() TTN déconnecté");
                    break;
                case CommunicationMQTT.TTN_MESSAGE:
                    Log.d(TAG, "handleMessage() TTN topic = " + bundle.getString("topic") + " message =
      " + bundle.getString("message"));
                    break;
            }
        }
    }
```

Handler de communication entre l'activité et la communication MQTT.

Définition à la ligne 271 du fichier MainActivity.java.

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

8.2.3.4 layoutManager

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

Définition à la ligne 56 du fichier MainActivity.java.

Référencé par com.lasalle.beehoneyt.MainActivity.initialiserVue().

8.2.3.5 NB_RUCHES

```
final int com.lasalle.beehoneyt.MainActivity.NB_RUCHES = 5  [static], [private]
```

le nombre max. de ruches

Définition à la ligne 42 du fichier MainActivity.java.

8.2.3.6 nouvelleRuche

```
boolean com.lasalle.beehoneyt.MainActivity.nouvelleRuche = false  [private]
```

indique si une nouvelle ruche a été récupérée

Définition à la ligne 48 du fichier MainActivity.java.

8.2.3.7 recyclerView

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

Ressources graphiques

Définition à la ligne 54 du fichier MainActivity.java.

8.2.3.8 ruches

```
List<Ruche> com.lasalle.beehoneyt.MainActivity.ruches = new ArrayList<>()  [private]
```

la liste des ruches connues

Attributs

Définition à la ligne 47 du fichier MainActivity.java.

8.2.3.9 swipeRefreshLayout

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

Définition à la ligne 57 du fichier MainActivity.java.

8.2.3.10 TAG

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

Constantes

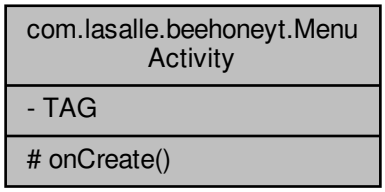
Définition à la ligne 41 du fichier MainActivity.java.

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

8.3 Référence de la classe com.lasalle.beehoneyt.MenuActivity

Déclaration de la classe MenuActivity.

Graphe de collaboration de com.lasalle.beehoneyt.MenuActivity :



Fonctions membres protégées

— void onCreate (@Nullable Bundle savedInstanceState)

Attributs privés statiques

— static final String TAG = "MenuActivity"

8.3.1 Description détaillée

Déclaration de la classe MenuActivity.

Auteur

Ethan VILLESSECHE

Définition à la ligne 21 du fichier MenuActivity.java.

8.3.2 Documentation des fonctions membres

8.3.2.1 onCreate()

```
void com.lasalle.beehoneyt.MenuActivity.onCreate (
            @Nullable Bundle savedInstanceState )  [protected]
```

Définition à la ligne 26 du fichier MenuActivity.java.

```
00026                                                       {
00027         super.onCreate(savedInstanceState);
00028         setContentView(R.layout.activity_menu);
00029         Log.d(TAG, "onCreate: started");
00030
00031     }
```

8.3.3 Documentation des données membres

8.3.3.1 TAG

```
final String com.lasalle.beehoneyt.MenuActivity.TAG = "MenuActivity"  [static], [private]
```

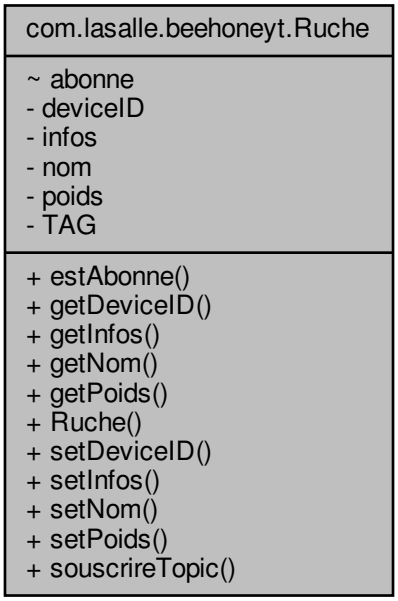
Définition à la ligne 23 du fichier MenuActivity.java.

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

8.4 Référence de la classe com.lasalle.beehoneyt.Ruche

Déclaration de la classe Ruche.

Graphe de collaboration de com.lasalle.beehoneyt.Ruche :



Fonctions membres publiques

- boolean estAbonne ()
  Accesseur get de abonne a.
- String getDeviceID ()
  Accesseur get du device id de la ruche.
- String getInfos ()
  Accesseur get des information de la ruche.
- String getNom ()
  Accesseur get du nom de la ruche.
- int getPoids ()
  Accesseur get du poids de la ruche.
- Ruche (String nom, String deviceID)
  Constructeur de la classe Ruche.
- void setDeviceID (String deviceID)
  Mutateur set du device id de la ruche.
- void setInfos (String Infos)
  Mutateur set des informations de la ruche.
- void setNom (String nom)
  Mutateur set du nom de la ruche.
- void setPoids (int Poids)
  Mutateur set du poids de la ruche.
- void souscrireTopic ()
  Permet de s'abonner au topic TTN du deviceID de la ruche.

Attributs privés

- String deviceID
  l'id de la ruche
- String infos
  Les informations sur la ruche.
- String nom
  Le nom de la ruche.
- int poids
  Le poids de la ruche.

Attributs privés statiques

- static final String TAG = "Ruche"

8.4.1 Description détaillée

Déclaration de la classe Ruche.

Définition à la ligne 16 du fichier Ruche.java.

8.4.2 Documentation des constructeurs et destructeur

8.4.2.1 Ruche()

```
com.lasalle.beehoneyt.Ruche.Ruche (
            String nom,
            String deviceID )
```

Constructeur de la classe Ruche.

Paramètres

nom	
deviceID	

Définition à la ligne 38 du fichier Ruche.java.

Références com.lasalle.beehoneyt.Ruche.deviceID, et com.lasalle.beehoneyt.Ruche.nom.

```
00039     {
00040         Log.d(TAG, "Ruche() nom = " + nom + " deviceID = " + deviceID + " instance = " + this
      );
00041         this.nom = nom;
00042         this.deviceID = deviceID;
00043         this.abonne = false;
00044     }
```

8.4.3 Documentation des fonctions membres

8.4.3.1 estAbonne()

```
boolean com.lasalle.beehoneyt.Ruche.estAbonne ( )
```

Accesseur get de abonne a.

Renvoie

boolean abonne

Définition à la ligne 137 du fichier Ruche.java.

```
00138     {
00139         return abonne;
00140     }
```

8.4.3.2 getDeviceID()

```
com.lasalle.beehoneyt.Ruche.getDeviceID ( )
```

Accesseur get du device id de la ruche.

Renvoie

String le device id de la ruche

Définition à la ligne 105 du fichier Ruche.java.

Références com.lasalle.beehoneyt.Ruche.deviceID.

Référencé par com.lasalle.beehoneyt.RucheActivity.afficherInfo(), et com.lasalle.beehoneyt.RucheActivity.onCreate().

```
00106     {
00107         return deviceID;
00108     }
```

8.4.3.3 getInfos()

```
com.lasalle.beehoneyt.Ruche.getInfos ( )
```

Accesseur get des information de la ruche.

Renvoie

String les information de la ruche

Définition à la ligne 116 du fichier Ruche.java.

Références com.lasalle.beehoneyt.Ruche.infos.

Référencé par com.lasalle.beehoneyt.RuchesViewHolder.afficher().

```
00117     {
00118         return infos;
00119     }
```

8.4.3.4 getNom()

```
com.lasalle.beehoneyt.Ruche.getNom ( )
```

Accesseur get du nom de la ruche.

Renvoie

String le nom de la ruche

Définition à la ligne 94 du fichier Ruche.java.

Références com.lasalle.beehoneyt.Ruche.nom.

Référencé par com.lasalle.beehoneyt.RuchesViewHolder.afficher(), com.lasalle.beehoneyt.RucheActivity.afficherInfo(), com.lasalle.↩beehoneyt.RucheAdapter.onBindViewHolder(), et com.lasalle.beehoneyt.RucheActivity.onCreate().

```
00095     {
00096         return nom;
00097     }
```

8.4.3.5 getPoids()

```
com.lasalle.beehoneyt.Ruche.getPoids ( )
```

Accesseur get du poids de la ruche.

Renvoie

String le poids de la ruche

Définition à la ligne 127 du fichier Ruche.java.

Références com.lasalle.beehoneyt.Ruche.poids.

```
00128     {
00129         return poids;
00130     }
```

8.4.3.6 setDeviceID()

```
void com.lasalle.beehoneyt.Ruche.setDeviceID (
             String deviceID )
```

Mutateur set du device id de la ruche.

Paramètres

deviceID	le nouveau device id de la ruche

Définition à la ligne 62 du fichier Ruche.java.

Références com.lasalle.beehoneyt.Ruche.deviceID.

```
00063     {
00064         this.deviceID = deviceID;
00065     }
```

8.4.3.7 setInfos()

```
void com.lasalle.beehoneyt.Ruche.setInfos (
             String Infos )
```

Mutateur set des informations de la ruche.

Paramètres

Infos	les nouvelles informations de la ruche

Définition à la ligne 72 du fichier Ruche.java.

```
00073     {
00074         this.infos = Infos;
00075     }
```

8.4.3.8 setNom()

```
com.lasalle.beehoneyt.Ruche.setNom (
             String nom )
```

Mutateur set du nom de la ruche.

Paramètres

nom	le nouveau nom de la ruche

Définition à la ligne 52 du fichier Ruche.java.

Références com.lasalle.beehoneyt.Ruche.nom.

```
00053     {
00054         this.nom = nom;
00055     }
```

8.4.3.9 setPoids()

```
void com.lasalle.beehoneyt.Ruche.setPoids (
            int Poids )
```

Mutateur set du poids de la ruche.

Paramètres

Poids	le nouveau poids de la ruche

Définition à la ligne 82 du fichier Ruche.java.

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.decoderDonneExterieure(), et com.lasalle.beehoneyt.RucheActivity.setPoids().

```
00083     {
00084         this.poids = Poids;
00085     }
```

8.4.3.10 souscrireTopic()

```
void com.lasalle.beehoneyt.Ruche.souscrireTopic ( )
```

Permet de s'abonner au topic TTN du deviceID de la ruche.

Définition à la ligne 146 du fichier Ruche.java.

Références com.lasalle.beehoneyt.CommunicationMQTT.souscrireTopic().

Référencé par com.lasalle.beehoneyt.RucheAdapter.onBindViewHolder().

```
00147     {
00148         if(!abonne)
00149         {
00150             CommunicationMQTT.souscrireTopic(deviceID);
00151             abonne = true;
00152             Log.d(TAG, "souscrireTopic() topic = " + deviceID);
00153         }
00154     }
```

8.4.4 Documentation des données membres

8.4.4.1 deviceID

```
String com.lasalle.beehoneyt.Ruche.deviceID  [private]
```

l'id de la ruche

Définition à la ligne 26 du fichier Ruche.java.

Référencé par com.lasalle.beehoneyt.Ruche.getDeviceID(), com.lasalle.beehoneyt.Ruche.Ruche(), et com.lasalle.beehoneyt.Ruche.setDeviceID().

8.4.4.2 infos

```
String com.lasalle.beehoneyt.Ruche.infos  [private]
```

Les informations sur la ruche.

Définition à la ligne 27 du fichier Ruche.java.

Référencé par com.lasalle.beehoneyt.Ruche.getInfos().

8.4.4.3 nom

```
String com.lasalle.beehoneyt.Ruche.nom  [private]
```

Le nom de la ruche.

Attributs

Définition à la ligne 25 du fichier Ruche.java.

Référencé par com.lasalle.beehoneyt.Ruche.getNom(), com.lasalle.beehoneyt.Ruche.Ruche(), et com.lasalle.beehoneyt.Ruche.setNom().

8.4.4.4 poids

```
int com.lasalle.beehoneyt.Ruche.poids  [private]
```

Le poids de la ruche.

Définition à la ligne 28 du fichier Ruche.java.

Référencé par com.lasalle.beehoneyt.Ruche.getPoids().

8.4.4.5 TAG

```
final String com.lasalle.beehoneyt.Ruche.TAG = "Ruche"  [static], [private]
```

Constantes

Définition à la ligne 21 du fichier Ruche.java.

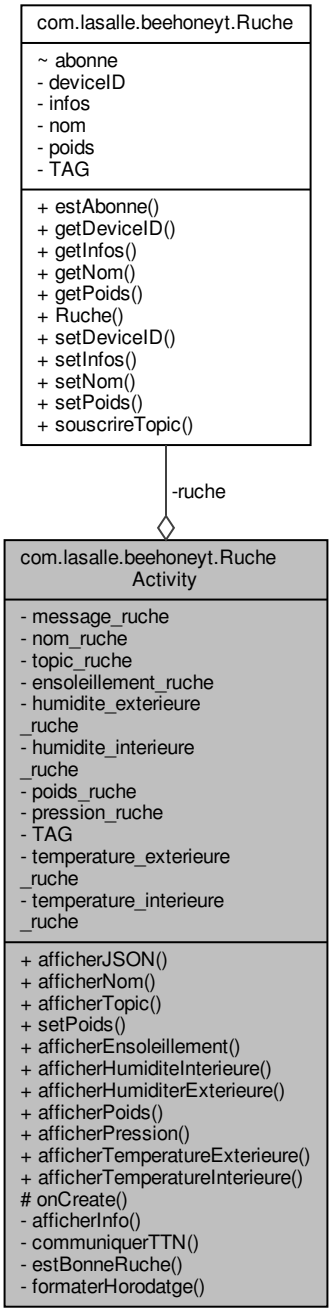
La documentation de cette classe a été générée à partir du fichier suivant :

— Ruche.java

8.5 Référence de la classe com.lasalle.beehoneyt.RucheActivity

Déclaration de la classe RucheActivity.

Graphe de collaboration de com.lasalle.beehoneyt.RucheActivity :



Fonctions membres publiques

— void afficherJSON (String message)
affiche le dernier message JSON recu
— void afficherNom (String message)

afficher le nom d'un ruche
- void afficherTopic (String message)
  afficher le topic
- void setPoids (int poids)

Fonctions membres publiques statiques

- static void afficherEnsoleillement (String message)
  affichie l'Ensoleillement dans le layout
- static void afficherHumiditeInterieure (String message)
  affichie l'humiditer interieure dans le layout
- static void afficherHumiditerExterieure (String message)
  affichie l'humiditer exterieure dans le layout
- static void afficherPoids (int message)
  affichie le poid dans le layout
- static void afficherPression (String message)
  affichie la pression dans le layout
- static void afficherTemperatureExterieure (String message)
  affichie la Temperature exterieure dans le layout
- static void afficherTemperatureInterieure (String message)
  affichie la Temperature interieure dans le layout

Fonctions membres protégées

- void onCreate (Bundle savedInstanceState)

Fonctions membres privées

- void afficherInfo ()
  afficher les information d'un ruche
- void communiquerTTN (final String deviceID)
  paramete la communication avec le ttn
- boolean estBonneRuche (String deviceID, String message)
  verifie si le json reçu correspond a la ruche souhaité
- String formaterHorodatge (String horodatage)
  formate l'horodatage

Attributs privés

- TextView message_ruche
- TextView nom_ruche
- Ruche ruche = null
- TextView topic_ruche

Attributs privés statiques

- static TextView ensoleillement_ruche
- static TextView humidite_exterieure_ruche
- static TextView humidite_interieure_ruche
- static TextView poids_ruche
- static TextView pression_ruche
- static final String TAG = "RucheActivity"
- static TextView temperature_exterieure_ruche
- static TextView temperature_interieure_ruche

8.5.1 Description détaillée

Déclaration de la classe RucheActivity.

Auteur

Ethan VILLESSECHE

Définition à la ligne 34 du fichier RucheActivity.java.

8.5.2 Documentation des fonctions membres

8.5.2.1 afficherEnsoleillement()

```
static void com.lasalle.beehoneyt.RucheActivity.afficherEnsoleillement (
            String message )  [static]
```

affichie l'Ensoleillement dans le layout

Paramètres

message	

Définition à la ligne 292 du fichier RucheActivity.java.

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.decoderDonneExterieure().

```
00293     {
00294         ensoleillement_ruche.setText("Ensoleillement : " + message + " lux");
00295     }
```

8.5.2.2 afficherHumiditeInterieure()

```
static void com.lasalle.beehoneyt.RucheActivity.afficherHumiditeInterieure (
            String message )  [static]
```

affichie l'humiditer interieure dans le layout

Paramètres

message	

Définition à la ligne 268 du fichier RucheActivity.java.

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.decoderDonneInterieure().

```
00269     {
00270         humidite_interieure_ruche.setText("Humidité interieure : " + message + " %
      ");
00271     }
```

8.5.2.3 afficherHumiditerExterieure()

```
static void com.lasalle.beehoneyt.RucheActivity.afficherHumiditerExterieure (
            String message )  [static]
```

affichie l'humiditer exterieure dans le layout

Paramètres

message	

Définition à la ligne 276 du fichier RucheActivity.java.

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.decoderDonneExterieure().

```
00277     {
00278         humidite_exterieure_ruche.setText("Humidité exterieure : " + message+ " %"
      );
00279     }
```

8.5.2.4 afficherInfo()

```
com.lasalle.beehoneyt.RucheActivity.afficherInfo ( )  [private]
```

afficher les information d'un ruche

Définition à la ligne 91 du fichier RucheActivity.java.

Références com.lasalle.beehoneyt.RucheActivity.afficherNom(), com.lasalle.beehoneyt.RucheActivity.afficherTopic(), com.lasalle.↩ beehoneyt.Ruche.getDeviceID(), et com.lasalle.beehoneyt.Ruche.getNom().

Référencé par com.lasalle.beehoneyt.RucheActivity.onCreate().

```
00092     {
00093         afficherNom(ruche.getNom());
00094         afficherTopic(ruche.getDeviceID());
00095     }
```

8.5.2.5 afficherJSON()

```
void com.lasalle.beehoneyt.RucheActivity.afficherJSON (
             String message )
```

affiche le dernier message JSON recu

Paramètres

message	

Définition à la ligne 122 du fichier RucheActivity.java.

Référencé par com.lasalle.beehoneyt.RucheActivity.communiquerTTN().

```
00123     {
00124         message_ruche.setText("Message : " + message);
00125     }
```

8.5.2.6 afficherNom()

```
com.lasalle.beehoneyt.RucheActivity.afficherNom (
            String message )
```

afficher le nom d'un ruche

Définition à la ligne 101 du fichier RucheActivity.java.

Référencé par com.lasalle.beehoneyt.RucheActivity.afficherInfo().

```
00102     {
00103         nom_ruche.setText("Nom : " + message);
00104     }
```

8.5.2.7 afficherPoids()

```
static void com.lasalle.beehoneyt.RucheActivity.afficherPoids (
            int message )  [static]
```

affichie le poid dans le layout

Paramètres

message	

Définition à la ligne 240 du fichier RucheActivity.java.

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.decoderDonneExterieure().

```
00241     {
00242         poids_ruche.setText("Poids : " + message + "kg");
00243     }
```

8.5.2.8 afficherPression()

```
static void com.lasalle.beehoneyt.RucheActivity.afficherPression (
            String message )  [static]
```

affichie la pression dans le layout

Paramètres

message	

Définition à la ligne 284 du fichier RucheActivity.java.

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.decoderDonneExterieure().

```
00285     {
00286         pression_ruche.setText("Pression atmosphérique : " + message + " hPa");
00287     }
```

8.5.2.9 afficherTemperatureExterieure()

```
static void com.lasalle.beehoneyt.RucheActivity.afficherTemperatureExterieure (
            String message )  [static]
```

affichie la Temperature exterieure dans le layout

Paramètres

message	

Définition à la ligne 260 du fichier RucheActivity.java.

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.decoderDonneExterieure().

```
00261     {
00262         temperature_exterieure_ruche.setText("Temperature exterieure : " +
      message + " °C");
00263     }
```

8.5.2.10 afficherTemperatureInterieure()

```
static void com.lasalle.beehoneyt.RucheActivity.afficherTemperatureInterieure (
            String message )  [static]
```

affichie la Temperature interieure dans le layout

Paramètres

message	

Définition à la ligne 252 du fichier RucheActivity.java.

Référencé par com.lasalle.beehoneyt.CommunicationMQTT.decoderDonneInterieure().

```
00253     {
00254         temperature_interieure_ruche.setText("Temperature interieure : " +
      message + " °C");
00255     }
```

8.5.2.11 afficherTopic()

```
com.lasalle.beehoneyt.RucheActivity.afficherTopic (
            String message )
```

afficher le topic

Paramètres

message	

Définition à la ligne 113 du fichier RucheActivity.java.

Référencé par com.lasalle.beehoneyt.RucheActivity.afficherInfo().

```
00114     {
00115         topic_ruche.setText("Topic : " + message);
00116     }
```

8.5.2.12 communiquerTTN()

```
void com.lasalle.beehoneyt.RucheActivity.communiquerTTN (
            final String deviceID )  [private]
```

paramete la communication avec le ttn

Paramètres

deviceID	

Définition à la ligne 132 du fichier RucheActivity.java.

Références com.lasalle.beehoneyt.RucheActivity.afficherJSON(), com.lasalle.beehoneyt.CommunicationMQTT.decoderMessage(), com.lasalle.beehoneyt.RucheActivity.estBonneRuche(), com.lasalle.beehoneyt.CommunicationMQTT.extraireHorodatage(), com.lasalle.beehoneyt.RucheActivity.formaterHorodatge(), com.lasalle.beehoneyt.RucheActivity.ruche, et com.lasalle.beehoneyt.CommunicationMQTT.setCallback().

Référencé par com.lasalle.beehoneyt.RucheActivity.onCreate().

```
00133     {
00134         Log.d(TAG, "deviceID : " + deviceID);
00135         CommunicationMQTT.setCallback(new MqttCallbackExtended()
00136         {
00137             @Override
00138             public void connectComplete(boolean b, String s) {
00139
00140             }
00141             @Override
00142             public void connectionLost(Throwable throwable) {
00143
00144             }
00145             @Override
00146             public void messageArrived(String topic, MqttMessage mqttMessage) throws Exception
00147             {
00148                 Log.d(TAG, "Topic : " + topic);
00149                 Log.d(TAG, "Message : " + mqttMessage.toString());
00150                 if(estBonneRuche(deviceID, mqttMessage.toString()))
00151                 {
00152                     afficherJSON(mqttMessage.toString());
00153                     CommunicationMQTT.decoderMessage(mqttMessage.toString(),
      ruche);
00154                     String horodatage = CommunicationMQTT.extraireHorodatage(mqttMessage.toString());
00155                     String horodatageFormate = formaterHorodatge(horodatage);
00156                     Log.d(TAG, "Horodatage formaté : " + horodatageFormate);
00157                 }
00158
00159             }
00160
00161             @Override
00162             public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken)
00163             {
00164             }
00165         });
00166     }
```

8.5.2.13 estBonneRuche()

```
boolean com.lasalle.beehoneyt.RucheActivity.estBonneRuche (
            String deviceID,
            String message )  [private]
```

verifie si le json reçu correspond a la ruche souhaité

Paramètres

deviceID	
message	

Renvoie

Définition à la ligne 174 du fichier RucheActivity.java.

Référencé par com.lasalle.beehoneyt.RucheActivity.communiquerTTN().

```
00175     {
00176         try
00177         {
00178             JSONObject json = null;
00179             Iterator<String> it = null;
00180 
00181             json = new JSONObject(message);
00182             int port = json.getInt("port");
00183 
00184             it = json.keys();
00185             while (it.hasNext())
00186             {
00187                 String cle = it.next();
00188                 Log.i(TAG, "clé = " + cle);
00189                 Log.i(TAG, "valeur = " + json.getString(cle));
00190                 //Log.i(TAG, "type = " + json.get(cle).getClass());
00191 
00192                 if (cle.equals("dev_id"))
00193                 {
00194                     if(json.getString(cle).equals(deviceID)) {
00195                         return true;
00196                     }
00197                 }
00198             }
00199             return false;
00200         }
00201         catch (JSONException e)
00202         {
00203             e.printStackTrace();
00204             return false;
00205         }
00206     }
```

8.5.2.14 formaterHorodatge()

```
String com.lasalle.beehoneyt.RucheActivity.formaterHorodatge (
            String horodatage )  [private]
```

formate l'horodatage

Paramètres

horodatage	

Renvoie

String

A faire Finaliser le formatage de l'horodatage

Définition à la ligne 213 du fichier RucheActivity.java.

Référencé par com.lasalle.beehoneyt.RucheActivity.communiquerTTN().

```
00214     {
00218         SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss", Locale.FRENCH);
00219         df.setTimeZone(TimeZone.getTimeZone("UTC"));
00220         Date date = null;
00221         String horodatageFormate = horodatage;
00222         try
00223         {
00224             date = df.parse(horodatage);
00225             df.setTimeZone(TimeZone.getDefault());
00226             horodatageFormate = df.format(date);
00227         }
00228         catch (ParseException e)
00229         {
00230             e.printStackTrace();
00231         }
00232
00233         return horodatageFormate;
00234     }
```

8.5.2.15 onCreate()

```
void com.lasalle.beehoneyt.RucheActivity.onCreate (
             Bundle savedInstanceState )  [protected]
```

Définition à la ligne 58 du fichier RucheActivity.java.

Références com.lasalle.beehoneyt.RucheActivity.afficherInfo(), com.lasalle.beehoneyt.RucheActivity.communiquerTTN(), com.lasalle.beehoneyt.Ruche.getDeviceID(), com.lasalle.beehoneyt.Ruche.getNom(), et com.lasalle.beehoneyt.RucheActivity.ruche.

```
00059     {
00060         super.onCreate(savedInstanceState);
00061         setContentView(R.layout.activity_ruche);
00062
00063         nom_ruche = (TextView) this.findViewById(R.id.nom_ruche);
00064         topic_ruche = (TextView) this.findViewById(R.id.topic_ruche);
00065         message_ruche = (TextView) this.findViewById(R.id.message_ruche);
00066         poids_ruche = (TextView) this.findViewById(R.id.poids_ruche);
00067         temperature_interieure_ruche = (TextView) this.findViewById(R.id.
      temperature_interieure_ruche);
00068         temperature_exterieure_ruche = (TextView) this.findViewById(R.id.
      temperature_exterieure_ruche);
00069         humidite_interieure_ruche = (TextView) this.findViewById(R.id.
      humidite_interieure_ruche);
00070         humidite_exterieure_ruche = (TextView) this.findViewById(R.id.
      humidite_exterieure_ruche);
00071         pression_ruche = (TextView) this.findViewById(R.id.pression_ruche);
00072         ensoleillement_ruche = (TextView) this.findViewById(R.id.ensoleillement_ruche);
00073
00074
00075
00076         Intent intent = getIntent();
00077         // Attention : un nouvel objet Ruche est créé
00078         ruche = (Ruche)intent.getSerializableExtra("Ruche");
00079         if(ruche != null)
00080         {
00081             Log.d(TAG, "Ruche : " + ruche.getNom() + " " + ruche);
00082             afficherInfo();
00083             communiquerTTN(ruche.getDeviceID());
00084         }
00085     }
```

8.5.2.16 setPoids()

```
void com.lasalle.beehoneyt.RucheActivity.setPoids (
            int poids )
```

Définition à la ligne 244 du fichier RucheActivity.java.

Références com.lasalle.beehoneyt.Ruche.setPoids().

```
00245     {
00246         ruche.setPoids(poids);
00247     }
```

8.5.3 Documentation des données membres

8.5.3.1 ensoleillement_ruche

```
TextView com.lasalle.beehoneyt.RucheActivity.ensoleillement_ruche  [static], [private]
```

Définition à la ligne 55 du fichier RucheActivity.java.

8.5.3.2 humidite_exterieure_ruche

```
TextView com.lasalle.beehoneyt.RucheActivity.humidite_exterieure_ruche  [static], [private]
```

Définition à la ligne 53 du fichier RucheActivity.java.

8.5.3.3 humidite_interieure_ruche

```
TextView com.lasalle.beehoneyt.RucheActivity.humidite_interieure_ruche  [static], [private]
```

Définition à la ligne 52 du fichier RucheActivity.java.

8.5.3.4 message_ruche

```
TextView com.lasalle.beehoneyt.RucheActivity.message_ruche  [private]
```

Définition à la ligne 47 du fichier RucheActivity.java.

8.5.3.5 nom_ruche

```
TextView com.lasalle.beehoneyt.RucheActivity.nom_ruche  [private]
```

Définition à la ligne 45 du fichier RucheActivity.java.

8.5.3.6 poids_ruche

```
TextView com.lasalle.beehoneyt.RucheActivity.poids_ruche  [static], [private]
```

Définition à la ligne 49 du fichier RucheActivity.java.

8.5.3.7 pression_ruche

```
TextView com.lasalle.beehoneyt.RucheActivity.pression_ruche  [static], [private]
```

Définition à la ligne 54 du fichier RucheActivity.java.

8.5.3.8 ruche

```
Ruche com.lasalle.beehoneyt.RucheActivity.ruche = null  [private]
```

Attributs

Définition à la ligne 44 du fichier RucheActivity.java.

Référencé par com.lasalle.beehoneyt.RucheActivity.communiquerTTN(), et com.lasalle.beehoneyt.RucheActivity.onCreate().

8.5.3.9 TAG

```
final String com.lasalle.beehoneyt.RucheActivity.TAG = "RucheActivity"  [static], [private]
```

Constantes

Définition à la ligne 39 du fichier RucheActivity.java.

8.5.3.10 temperature_exterieure_ruche

```
TextView com.lasalle.beehoneyt.RucheActivity.temperature_exterieure_ruche  [static], [private]
```

Définition à la ligne 51 du fichier RucheActivity.java.

8.5.3.11 temperature_interieure_ruche

```
TextView com.lasalle.beehoneyt.RucheActivity.temperature_interieure_ruche  [static], [private]
```

Définition à la ligne 50 du fichier RucheActivity.java.

8.5.3.12 topic_ruche

```
TextView com.lasalle.beehoneyt.RucheActivity.topic_ruche  [private]
```

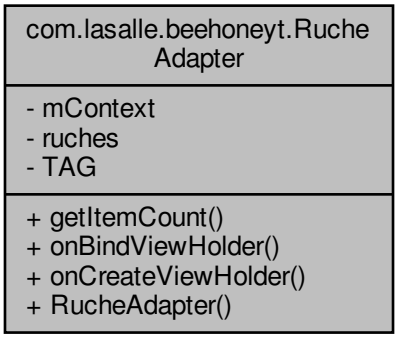
Définition à la ligne 46 du fichier RucheActivity.java.

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

8.6 Référence de la classe com.lasalle.beehoneyt.RucheAdapter

Déclaration de la classe RucheAdapter.

Graphe de collaboration de com.lasalle.beehoneyt.RucheAdapter :



Fonctions membres publiques

— int getItemCount ()
— void onBindViewHolder (@NonNull RuchesViewHolder holder, final int position)
— RuchesViewHolder onCreateViewHolder (@NonNull ViewGroup parent, int viewType)
— RucheAdapter (Context context, List< Ruche > ruches)

Attributs privés

— Context mContext
— List< Ruche > ruches = null

Attributs privés statiques

— static final String TAG = "RucheAdapter"

8.6.1 Description détaillée

Déclaration de la classe RucheAdapter.

Auteur

Ethan VILLESSECHE

Définition à la ligne 30 du fichier RucheAdapter.java.

8.6.2 Documentation des constructeurs et destructeur

8.6.2.1 RucheAdapter()

```
com.lasalle.beehoneyt.RucheAdapter.RucheAdapter (
            Context context,
            List< Ruche > ruches )
```

Définition à la ligne 36 du fichier RucheAdapter.java.

Références com.lasalle.beehoneyt.RucheAdapter.ruches.

```
00037     {
00038         if(ruches != null)
00039         {
00040             this.ruches = ruches;
00041             mContext = context;
00042         }
00043     }
```

8.6.3 Documentation des fonctions membres

8.6.3.1 getItemCount()

```
int com.lasalle.beehoneyt.RucheAdapter.getItemCount ( )
```

Définition à la ligne 76 du fichier RucheAdapter.java.

```
00077     {
00078         if(ruches != null)
00079             return ruches.size();
00080         return 0;
00081     }
```

8.6.3.2 onBindViewHolder()

```
void com.lasalle.beehoneyt.RucheAdapter.onBindViewHolder (
            @NonNull RuchesViewHolder holder,
            final int position )
```

Définition à la ligne 54 du fichier RucheAdapter.java.

Références com.lasalle.beehoneyt.Ruche.getNom(), et com.lasalle.beehoneyt.Ruche.souscrireTopic().

```
00055     {
00056         Log.d(TAG, "onBindViewHolder: appel.");
00057
00058         final Ruche ruche = ruches.get(position);
00059         holder.afficher(ruche);
00060
00061         holder.cardview.setOnClickListener(new View.OnClickListener(){
00062             @Override
00063             public  void onClick(View view) {
00064                 Log.d(TAG, "onClick: click sur : " + ruche);
00065                 Toast.makeText(mContext, ruche.getNom(), Toast.LENGTH_SHORT).show();
00066
00067                 ruche.souscrireTopic();
00068                 Intent intent = new Intent(mContext, RucheActivity.class);
00069                 intent.putExtra("Ruche", (Serializable) ruche);
00070                 mContext.startActivity(intent);
00071             }
00072         });
00073     }
```

8.6.3.3 onCreateViewHolder()

```
RuchesViewHolder com.lasalle.beehoneyt.RucheAdapter.onCreateViewHolder (
            @NonNull ViewGroup parent,
            int viewType )
```

Définition à la ligne 46 du fichier RucheAdapter.java.

```
00047     {
00048         LayoutInflater inflater = LayoutInflater.from(parent.getContext());
00049         View view = inflater.inflate(R.layout.ruche, parent, false);
00050         return new RuchesViewHolder(view);
00051     }
```

8.6.4 Documentation des données membres

8.6.4.1 mContext

```
Context com.lasalle.beehoneyt.RucheAdapter.mContext  [private]
```

Définition à la ligne 33 du fichier RucheAdapter.java.

8.6.4.2 ruches

```
List<Ruche> com.lasalle.beehoneyt.RucheAdapter.ruches = null  [private]
```

Définition à la ligne 34 du fichier RucheAdapter.java.

Référencé par com.lasalle.beehoneyt.RucheAdapter.RucheAdapter().

8.6.4.3 TAG

```
final String com.lasalle.beehoneyt.RucheAdapter.TAG = "RucheAdapter"  [static], [private]
```

Définition à la ligne 32 du fichier RucheAdapter.java.

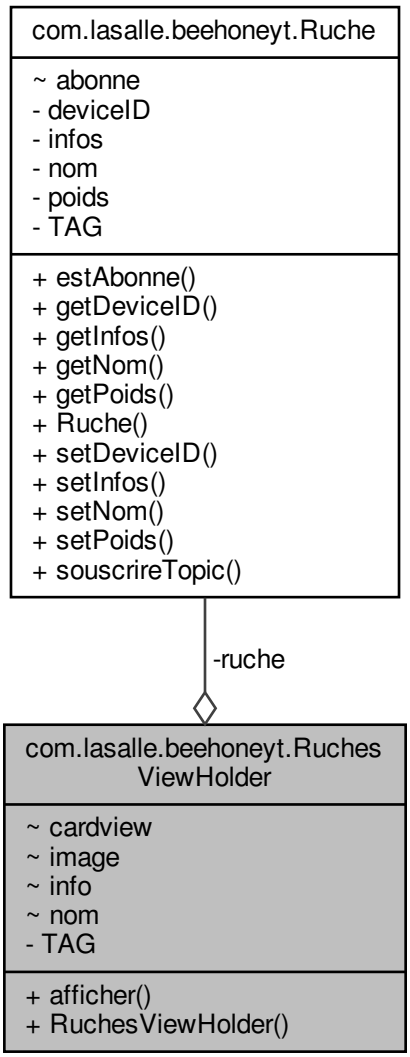
La documentation de cette classe a été générée à partir du fichier suivant :

— RucheAdapter.java

8.7 Référence de la classe com.lasalle.beehoneyt.RuchesViewHolder

Déclaration de la classe RuchesViewHolder.

Graphe de collaboration de com.lasalle.beehoneyt.RuchesViewHolder :



Fonctions membres publiques

— void afficher (Ruche ruche)
— RuchesViewHolder (final View itemView)

Attributs privés

— Ruche ruche

Attributs privés statiques

— static final String TAG = "RuchesViewHolder"

8.7.1 Description détaillée

Déclaration de la classe RuchesViewHolder.

Auteur

Ethan VILLESSECHE

Définition à la ligne 22 du fichier RuchesViewHolder.java.

8.7.2 Documentation des constructeurs et destructeur

8.7.2.1 RuchesViewHolder()

```
com.lasalle.beehoneyt.RuchesViewHolder.RuchesViewHolder (
             final View itemView )
```

Définition à la ligne 33 du fichier RuchesViewHolder.java.

```
00034     {
00035         super(itemView);
00036
00037         image = itemView.findViewById(R.id.circle_image);
00038         nom = ((TextView) itemView.findViewById(R.id.nom));
00039         info = ((TextView) itemView.findViewById(R.id.info));
00040        // poids = ((TextView) itemView.findViewById(R.id.poids));
00041         cardview = itemView.findViewById(R.id.card_view);
00042     }
```

8.7.3 Documentation des fonctions membres

8.7.3.1 afficher()

```
void com.lasalle.beehoneyt.RuchesViewHolder.afficher (
             Ruche ruche )
```

Définition à la ligne 44 du fichier RuchesViewHolder.java.

Références com.lasalle.beehoneyt.Ruche.getInfos(), com.lasalle.beehoneyt.Ruche.getNom(), et com.lasalle.beehoneyt.Ruches↩ ViewHolder.ruche.

```
00045     {
00046         this.ruche = ruche;
00047         nom.setText(ruche.getNom());
00048         info.setText(ruche.getInfos());
00049         //poids.setText(ruche.getPoids() + " Kg");
00050     }
```

8.7.4 Documentation des données membres

8.7.4.1 ruche

`Ruche com.lasalle.beehoneyt.RuchesViewHolder.ruche [private]`

Définition à la ligne 31 du fichier RuchesViewHolder.java.

Référencé par com.lasalle.beehoneyt.RuchesViewHolder.afficher().

8.7.4.2 TAG

`final String com.lasalle.beehoneyt.RuchesViewHolder.TAG = "RuchesViewHolder" [static], [private]`

Définition à la ligne 24 du fichier RuchesViewHolder.java.

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

— RuchesViewHolder.java

9 Documentation des fichiers

9.1 Référence du fichier Changelog.md

9.2 Changelog.md

```
00001 \page page_changelog Changelog
00002
00003
00004 r38 | evillesseche | 2020-04-03 10:42:19 +0200 (ven. 03 avril 2020) | 1 ligne
00005
00006 diagramme de déploiement
00007
00008 r36 | evillesseche | 2020-04-03 05:49:12 +0200 (ven. 03 avril 2020) | 1 ligne
00009
00010 ajout de BOUML
00011
00012 r34 | evillesseche | 2020-04-03 03:36:37 +0200 (ven. 03 avril 2020) | 1 ligne
00013
00014 ajout de la gestion du port , affichage des donné , gestion du device id
00015
00016 r33 | tvaira | 2020-04-02 19:06:15 +0200 (jeu. 02 avril 2020) | 1 ligne
00017
00018 Réception des messages MQTT
00019
00020 r29 | tvaira | 2020-04-02 11:40:16 +0200 (jeu. 02 avril 2020) | 1 ligne
00021
00022 Fichiers inutiles
00023
00024 r28 | evillesseche | 2020-04-02 07:46:05 +0200 (jeu. 02 avril 2020) | 1 ligne
00025
00026 modification de la connections au TTN es de abonnement  au ruche
00027
00028 r27 | evillesseche | 2020-04-02 07:17:14 +0200 (jeu. 02 avril 2020) | 1 ligne
00029
00030 Ajout de methode pour decoder le JSON et s'aboner a un device
00031
00032 r18 | tvaira | 2020-03-20 11:47:50 +0100 (ven. 20 mars 2020) | 1 ligne
00033
00034 Intégration de la base de MQTT
00035
00036 r14 | tvaira | 2020-03-18 09:05:08 +0100 (mer. 18 mars 2020) | 1 ligne
00037
00038 Revision de code
00039
00040 r13 | tvaira | 2020-03-18 08:08:52 +0100 (mer. 18 mars 2020) | 1 ligne
00041
00042 fichier à ignorer
00043
00044 r9 | evillesseche | 2020-03-14 21:19:16 +0100 (sam. 14 mars 2020) | 1 ligne
00045
00046 Ajout de circleimageview et de click lisenner sur les recyclerView
00047
00048 r7 | evillesseche | 2020-03-13 16:30:12 +0100 (ven. 13 mars 2020) | 2 lignes
00049
00050 Mise en place d'un recyclerView
00051
00052
```

9.3 Référence du fichier CommunicationMQTT.java

Déclaration de la classe CommunicationMQTT.

Classes

— class com.lasalle.beehoneyt.CommunicationMQTT
Déclaration de la classe CommunicationMQTT.

Paquetages

— package com.lasalle.beehoneyt

9.3.1 Description détaillée

Déclaration de la classe CommunicationMQTT.

Auteur

Ethan VILLESSECHE

Définition dans le fichier CommunicationMQTT.java.

9.4 CommunicationMQTT.java

```
00001 package com.lasalle.beehoneyt;
00002
00009 import android.content.Context;
00010 import android.os.Bundle;
00011 import android.os.Handler;
00012 import android.os.Message;
00013 import android.util.Log;
00014 import org.eclipse.paho.android.service.MqttAndroidClient;
00015 import org.eclipse.paho.client.mqttv3.DisconnectedBufferOptions;
00016 import org.eclipse.paho.client.mqttv3.IMqttActionListener;
00017 import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
00018 import org.eclipse.paho.client.mqttv3.IMqttToken;
00019 import org.eclipse.paho.client.mqttv3.MqttCallbackExtended;
00020 import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
00021 import org.eclipse.paho.client.mqttv3.MqttException;
00022 import org.eclipse.paho.client.mqttv3.MqttMessage;
00023 import org.json.JSONObject;
00024 import org.json.JSONArray;
00025 import org.json.JSONException;
00026 import java.util.Iterator;
00027
00032 public class CommunicationMQTT
00033 {
00037     private static final String TAG = "CommunicationMQTT";
00041     static public MqttAndroidClient mqttAndroidClient;
00042     private Handler handler = null;
00046     public static final int TTN_CONNECTE = 1;
00047     public static final int TTN_DECONNECTE = 2;
00048     public static final int TTN_MESSAGE = 3;
00052     private String serverUri = "tcp://eu.thethings.network:1883";
00053     static public String clientId = "mes_ruches";
00054     private String username = "mes_ruches";
00055     private String password = "ttn-account-v2.vC-aqMRnLLzGkNjODWgy81kLqzxBPAT8_mE-L7U2C_w";
00056
00063     public CommunicationMQTT(Context context, final Handler handler)
00064     {
00065         Log.v(TAG, "CommunicationMQTT() clientId = " + clientId);
00066         this.handler = handler;
00067         mqttAndroidClient = new MqttAndroidClient(context, serverUri, clientId);
00068         mqttAndroidClient.setCallback(new MqttCallbackExtended()
00069         {
00070             @Override
00071             public void connectComplete(boolean b, String s)
00072             {
```

```
00073                 Log.w(TAG, "connectComplete() serverUri = " + s + " connecte = " + mqttAndroidClient.
      isConnected());
00074                 Message msg = Message.obtain();
00075                 Bundle bundle = new Bundle();
00076                 bundle.putInt("etat", TTN_CONNECTE);
00077                 msg.setData(bundle);
00078                 handler.sendMessage(msg);
00079             }
00080
00081             @Override
00082             public void connectionLost(Throwable throwable)
00083             {
00084                 Log.w(TAG, "connectionLost()");
00085                 Message msg = Message.obtain();
00086                 Bundle b = new Bundle();
00087                 b.putInt("etat", TTN_DECONNECTE);
00088                 msg.setData(b);
00089                 handler.sendMessage(msg);
00090             }
00091
00092             @Override
00093             public void messageArrived(String topic, MqttMessage mqttMessage) throws Exception
00094             {
00095                 Log.w(TAG, "messageArrived() topic = " + topic + " message = " + mqttMessage.toString());
00096                 Message msg = Message.obtain();
00097                 Bundle b = new Bundle();
00098                 b.putInt("etat", TTN_MESSAGE);
00099                 b.putString("topic", topic);
00100                 b.putString("message", mqttMessage.toString());
00101                 msg.setData(b);
00102                 handler.sendMessage(msg);
00103             }
00104
00105             @Override
00106             public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken)
00107             {
00108                 Log.w(TAG, "deliveryComplete()");
00109             }
00110         });
00111
00112         connecter();
00113     }
00114
00121     static public void setCallback(MqttCallbackExtended callback)
00122     {
00123         mqttAndroidClient.setCallback(callback);
00124     }
00125
00131     private void connecter()
00132     {
00133         MqttConnectOptions mqttConnectOptions = new MqttConnectOptions();
00134         mqttConnectOptions.setAutomaticReconnect(true);
00135         mqttConnectOptions.setCleanSession(false);
00136         mqttConnectOptions.setUserName(username);
00137         mqttConnectOptions.setPassword(password.toCharArray());
00138
00139         try
00140         {
00141             Log.d(TAG, "connecter() serverUri = " + serverUri + " clientId = " + clientId);
00142             mqttAndroidClient.connect(mqttConnectOptions, null, new IMqttActionListener()
00143             {
00144                 @Override
00145                 public void onSuccess(IMqttToken asyncActionToken)
00146                 {
00147                     DisconnectedBufferOptions disconnectedBufferOptions = new DisconnectedBufferOptions();
00148                     disconnectedBufferOptions.setBufferEnabled(true);
00149                     disconnectedBufferOptions.setBufferSize(100);
00150                     disconnectedBufferOptions.setPersistBuffer(false);
00151                     disconnectedBufferOptions.setDeleteOldestMessages(false);
00152                     mqttAndroidClient.setBufferOpts(disconnectedBufferOptions);
00153                     Log.d(TAG, "onSuccess() serverUri = " + serverUri + " clientId = " + clientId);
00154                 }
00155
00156                 @Override
00157                 public void onFailure(IMqttToken asyncActionToken, Throwable exception)
00158                 {
00159                     Log.d(TAG, "onFailure() serverUri = " + serverUri + " clientId = " + clientId + "
      exception = " + exception.toString());
00160                 }
00161             });
00162         }
00163         catch (MqttException ex)
00164         {
00165             ex.printStackTrace();
00166         }
00167     }
00168
00174     public void reconnecter()
00175     {
00176         Log.w(TAG, "reconnecter ()");
00177         if (estConnecte())
```

```
00178             deconnecter();
00179         connecter();
00180     }
00181
00187     public void deconnecter()
00188     {
00189         Log.d(TAG, "deconnecter() serverUri = " + serverUri + " clientId = " + clientId);
00190         try
00191         {
00192             IMqttToken disconToken = mqttAndroidClient.disconnect();
00193             disconToken.setActionCallback(new IMqttActionListener()
00194             {
00195                 @Override
00196                 public void onSuccess(IMqttToken asyncActionToken)
00197                 {
00198                     Log.d(TAG, "onSuccess() serverUri = " + serverUri + " clientId = " + clientId);
00199                 }
00200
00201                 @Override
00202                 public void onFailure(IMqttToken asyncActionToken, Throwable exception)
00203                 {
00204                     Log.d(TAG, "onFailure() serverUri = " + serverUri + " clientId = " + clientId + "
      exception = " + exception.toString());
00205                 }
00206             });
00207         }
00208         catch (MqttException e)
00209         {
00210             e.printStackTrace();
00211         }
00212     }
00213
00219     static public boolean estConnecte()
00220     {
00221         Log.w(TAG, "estConnecte() " + mqttAndroidClient.isConnected());
00222
00223         return mqttAndroidClient.isConnected();
00224     }
00225
00232     static public boolean souscrireTopic(String topic)
00233     {
00234         if(mqttAndroidClient == null && !mqttAndroidClient.isConnected())
00235         {
00236             return false;
00237         }
00238         final String subTopic = clientId + "/devices/" + topic + "/up";
00239         Log.w(TAG, "souscrireTopic() topic = " + subTopic);
00240         try
00241         {
00242             final boolean[] retour = {false};
00243             mqttAndroidClient.subscribe(subTopic, 0, null, new IMqttActionListener()
00244             {
00245                 @Override
00246                 public void onSuccess(IMqttToken asyncActionToken)
00247                 {
00248                     Log.w(TAG, "onSuccess() topic = " + subTopic);
00249                     retour[0] = true;
00250                 }
00251
00252                 @Override
00253                 public void onFailure(IMqttToken asyncActionToken, Throwable exception)
00254                 {
00255                     Log.w(TAG, "onFailure() topic = " + subTopic);
00256                     retour[0] = false;
00257                 }
00258             });
00259             return retour[0];
00260         }
00261         catch (MqttException ex)
00262         {
00263             Log.w(TAG, "Erreur topic = " + subTopic);
00264             ex.printStackTrace();
00265             return false;
00266         }
00267     }
00268
00276     static public void decoderMessage(String message, Ruche ruche )
00277     {
00278         try
00279         {
00280             JSONObject json = null;
00281             Iterator<String> it = null;
00282
00283             json = new JSONObject(message);
00284             int port = json.getInt("port");
00285
00286             it = json.keys();
00287             while (it.hasNext())
00288             {
00289                 String cle = it.next();
00290                 Log.i(TAG, "decoderMessage() clé = " + cle);
```

```
00291                 Log.i(TAG, "decoderMessage() valeur = " + json.getString(cle));
00292                 //Log.i(TAG, "type = " + json.get(cle).getClass());
00293
00294                 if (cle.equals("payload_fields"))
00295                 {
00296                     decoderPayload(port, json.getString(cle), ruche);
00297                 }
00298             }
00299         }
00300         catch (JSONException e)
00301         {
00302             e.printStackTrace();
00303         }
00304     }
00305
00312     static public void decoderPayload(int port, String payload,
      Ruche ruche)
00313     {
00314
00315         Log.i(TAG, "decoderPayload() port = " + port );
00316         if ( port == 3)
00317         {
00318             decoderDonneInterieure(payload, ruche);
00319         }
00320         else
00321         {
00322             decoderDonneExterieure(payload, ruche);
00323         }
00324     }
00325
00331     static public void decoderDonneInterieure(String payload,
      Ruche ruche)
00332     {
00333         try
00334         {
00335             JSONObject json = null;
00336             json = new JSONObject(payload);
00337
00338             Iterator<String> it = null;
00339             it = json.keys();
00340             while (it.hasNext())
00341             {
00342                 String cle = it.next();
00343                 if (cle.equals("temperature"))
00344                 {
00345                     Log.i(TAG, "decoderDonneInterieure() temperature = " + json.getString(cle));
00346                     RucheActivity.afficherTemperatureInterieure(
      json.getString(cle));
00347                 }
00348                 else if (cle.equals("humidite"))
00349                 {
00350                     Log.i(TAG, "decoderDonneInterieure() humidite = " + json.getString(cle));
00351                     RucheActivity.afficherHumiditeInterieure(json.
      getString(cle));
00352                 }
00353             }
00354         }
00355         catch (JSONException e)
00356         {
00357             e.printStackTrace();
00358         }
00359     }
00360
00366     static public void decoderDonneExterieure(String payload,
      Ruche ruche)
00367     {
00368         try {
00369             JSONObject json = null;
00370             json = new JSONObject(payload);
00371
00372             Iterator<String> it = null;
00373             it = json.keys();
00374             while (it.hasNext()) {
00375                 String cle = it.next();
00376                 if (cle.equals("temperature"))
00377                 {
00378                     Log.i(TAG, "decoderDonneExterieure() temperature = " + json.getString(cle));
00379                     RucheActivity.afficherTemperatureExterieure(
      json.getString(cle));
00380                 }
00381                 else if (cle.equals("humidite"))
00382                 {
00383                     Log.i(TAG, "decoderDonneExterieure() humidite = " + json.getString(cle));
00384                     RucheActivity.afficherHumiditerExterieure(json.
      getString(cle));
00385                 }
00386                 else if (cle.equals("ensoleillement"))
00387                 {
00388                     Log.i(TAG, "decoderDonneExterieure() ensoleillement = " + json.getString(cle));
00389                     RucheActivity.afficherEnsoleillement(json.getString(
      cle));
```

```
00390                 }
00391                 else if (cle.equals("pression"))
00392                 {
00393                     Log.i(TAG, "decoderDonneExterieure() pression = " + json.getString(cle));
00394                     RucheActivity.afficherPression(json.getString(cle));
00395                 }
00396                 else if (cle.equals("poids"))
00397                 {
00398                     double poids = json.getDouble(cle);
00399                     poids = poids * 0.01;
00400                     Log.i(TAG, "decoderDonneExterieure() poids = " + json.getInt(cle));
00401                     ruche.setPoids(json.getInt(cle));
00402                     RucheActivity.afficherPoids(json.getInt(cle));
00403                 }
00404             }
00405         }
00406         catch (JSONException e)
00407         {
00408             e.printStackTrace();
00409         }
00410     }
00411
00416     static public String extraireHorodatage(String message)
00417     {
00418         String date = "";
00419
00420         try
00421         {
00422             JSONObject json = null;
00423             json = new JSONObject(message);
00424
00425             date = json.getJSONObject("metadata").getString("time");
00426             date = date.substring(0, 10) + " " + date.substring(11, 19);
00427             Log.d(TAG, "extraireHorodatage() time = " + json.getJSONObject("metadata").getString("time"));
00428             Log.d(TAG, "extraireHorodatage() horodatage = " + date);
00429
00430         }
00431         catch (JSONException e)
00432         {
00433             e.printStackTrace();
00434         }
00435
00436         return  date;
00437     }
00438 }
```

9.5 Référence du fichier MainActivity.java

Déclaration de la classe MainActivity.

Classes

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

Paquetages

— package com.lasalle.beehoneyt

9.5.1 Description détaillée

Déclaration de la classe MainActivity.

Auteur

Ethan VILLESSECHE

Définition dans le fichier MainActivity.java.

9.6 MainActivity.java

```
00001 package com.lasalle.beehoneyt;
00002
00003 import androidx.appcompat.app.AppCompatActivity;
00004 import androidx.recyclerview.widget.LinearLayoutManager;
00005 import androidx.recyclerview.widget.RecyclerView;
00006 import androidx.swiperefreshlayout.widget.SwipeRefreshLayout;
00007
00008 import android.os.Bundle;
00009 import android.os.Handler;
00010 import android.os.Message;
00011 import android.util.Log;
00012
00013 import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
00014 import org.eclipse.paho.client.mqttv3.MqttCallbackExtended;
00015 import org.eclipse.paho.client.mqttv3.MqttMessage;
00016 import org.json.JSONObject;
00017
00018 import java.util.ArrayList;
00019 import java.util.Arrays;
00020 import java.util.List;
00021
00036 public class MainActivity extends AppCompatActivity
00037 {
00041     private static final String TAG = "MainActivity";
00042     private static final int NB_RUCHES = 5;
00043
00047     private List<Ruche> ruches = new ArrayList<>();
00048     private boolean nouvelleRuche = false;
00049     private CommunicationMQTT communicationMQTT;
00050
00054     private RecyclerView recyclerView; // la vue
00055     private RecyclerView.Adapter adapter; // l'adaptateur
00056     private RecyclerView.LayoutManager layoutManager; // le gestionnaire de mise en page
00057     private SwipeRefreshLayout swipeRefreshLayout;
00058
00065     @Override
00066     protected void onCreate(Bundle savedInstanceState)
00067     {
00068         super.onCreate(savedInstanceState);
00069         setContentView(R.layout.activity_main);
00070         Log.d(TAG, "onCreate()");
00071
00072         communicationMQTT = new CommunicationMQTT(getApplicationContext(),
      handler);
00073
00074         initialiserVue();
00075
00076         nouvelleRuche = true;
00077         recupererRuches();
00078     }
00079
00083     @Override
00084     protected void onStart()
00085     {
00086         super.onStart();
00087         Log.i(TAG,"onStart()");
00088     }
00089
00093     @Override
00094     protected void onResume()
00095     {
00096         super.onResume();
00097         Log.i(TAG,"onResume()");
00102     }
00103
00107     @Override
00108     protected void onPause()
00109     {
00110         super.onPause();
00111         Log.i(TAG,"onPause()");
00112
00113     }
00114
00118     @Override
00119     protected void onStop()
00120     {
00121         super.onStop();
00122         Log.i(TAG,"onStop()");
00123
00124     }
00125
00129     @Override
00130     protected void onDestroy()
00131     {
00132         super.onDestroy();
00133         Log.i(TAG,"onDestroy()");
00134
00135     }
```

```
00136
00137     private void initialiserVue()
00138     {
00139         Log.d(TAG, "initialiserVue()");
00140         swipeRefreshLayout = (SwipeRefreshLayout)findViewById(R.id.swipeRefreshLayout);
00141         swipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener()
00142         {
00143             @Override
00144             public void onRefresh()
00145             {
00146                 recupererRuches();
00147             }
00148         });
00149
00150         recyclerView = (RecyclerView) findViewById(R.id.listeRuche);
00151         recyclerView.setHasFixedSize(true);
00152
00153         layoutManager = new LinearLayoutManager(this);
00154         recyclerView.setLayoutManager(layoutManager);
00155
00156         adapter = new RucheAdapter(this, ruches);
00157         recyclerView.setAdapter(adapter);
00158     }
00159
00165     private void recupererRuches()
00166     {
00167         Log.d(TAG, "recupererRuches()");
00171         if(nouvelleRuche)
00172         {
00173             nouvelleRuche = false;
00174
00175             // Pour les tests
00176             List<Ruche> listeRuches = Arrays.asList(
00177                     new Ruche("Ruche 1", "ruche_1"),
00178                     new Ruche("Ruche 2", "ruche_2"),
00179                     new Ruche("Ruche 3", "ruche_3"),
00180                     new Ruche("Ruche 4", "ruche_3")
00181             );
00182
00183             listeRuches.get(0).setInfos("Simulateur");
00184             listeRuches.get(1).setPoids(35);
00185             listeRuches.get(1).setInfos("Simulateur");
00186             listeRuches.get(2).setPoids(32);
00187             listeRuches.get(2).setInfos("Simulateur");
00188             listeRuches.get(3).setPoids(40);
00189             listeRuches.get(3).setInfos("Simulateur");
00190
00191             ruches.clear();
00192             for (int i = 0; i < listeRuches.size(); i++)
00193             {
00194                 ruches.add(listeRuches.get(i));
00195             }
00196
00197             rafraichir();
00198         }
00199     }
00200
00206     private void rafraichir()
00207     {
00208         swipeRefreshLayout.setRefreshing(false);
00209         adapter.notifyDataSetChanged();
00210     }
00211
00212     private void communiquerTTN(final String deviceID)
00213     {
00214         Log.d(TAG, "communiquerTTN() deviceID = " + deviceID);
00215         CommunicationMQTT.setCallback(new MqttCallbackExtended()
00216         {
00217             @Override
00218             public void connectComplete(boolean b, String s)
00219             {
00220             }
00221             @Override
00222             public void connectionLost(Throwable throwable)
00223             {
00224             }
00225             @Override
00226             public void messageArrived(String topic, MqttMessage mqttMessage) throws Exception
00227             {
00228                 Log.d(TAG, "communiquerTTN() topic = " + topic);
00229                 Log.d(TAG, "communiquerTTN() mqttMessage = " + mqttMessage.toString());
00233                 // Juste pour tester (à retirer rapidement) !
00234                 JSONObject json = new JSONObject(mqttMessage.toString());
00235                 String deviceID = json.getString("dev_id");
00236                 for (int i = 0; i < ruches.size(); i++)
00237                 {
00238                     if(ruches.get(i).getDeviceID().equals(deviceID))
00239                     {
00240                         ruches.get(i).setInfos("Message reçu !");
00241                     }
00242                     else
```

```
00243                     {
00244                         ruches.get(i).setInfos("Simulateur");
00245                     }
00246                 }
00247                 // fin du test
00248
00249                 rafraichir();
00250             }
00251
00252             @Override
00253             public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken)
00254             {
00255             }
00256         });
00257     }
00258
00259     private void abonnerRuches()
00260     {
00261         for (int i = 0; i < ruches.size(); i++)
00262         {
00263             ruches.get(i).souscrireTopic();
00264             communiquerTTN(ruches.get(i).getDeviceID());
00265         }
00266     }
00267
00271     final private Handler handler = new Handler()
00272     {
00273         @Override
00274         public void handleMessage(Message msg)
00275         {
00276             super.handleMessage(msg);
00277
00278             Bundle bundle = msg.getData();
00279
00280             switch(bundle.getInt("etat"))
00281             {
00282                 case CommunicationMQTT.TTN_CONNECTE:
00283                     Log.d(TAG, "handleMessage() TTN connecté");
00284                     abonnerRuches();
00285                     break;
00286                 case CommunicationMQTT.TTN_DECONNECTE:
00287                     Log.d(TAG, "handleMessage() TTN déconnecté");
00288                     break;
00289                 case CommunicationMQTT.TTN_MESSAGE:
00290                     Log.d(TAG, "handleMessage() TTN topic = " + bundle.getString("topic") + " message = " +
      bundle.getString("message"));
00291                     break;
00292             }
00293         }
00294     };
00295 }
```

9.7 Référence du fichier MenuActivity.java

Déclaration de la classe MenuActivity.

Classes

— class com.lasalle.beehoneyt.MenuActivity
Déclaration de la classe MenuActivity.

Paquetages

— package com.lasalle.beehoneyt

9.7.1 Description détaillée

Déclaration de la classe MenuActivity.

Auteur

Ethan VILLESSECHE

Définition dans le fichier MenuActivity.java.

9.8 MenuActivity.java

```
00001 package com.lasalle.beehoneyt;
00002
00009 import androidx.annotation.Nullable;
00010 import androidx.appcompat.app.AppCompatActivity;
00011
00012 import android.os.Bundle;
00013 import android.util.Log;
00014
00021 public class MenuActivity extends AppCompatActivity {
00022
00023     private static final String TAG = "MenuActivity";
00024
00025     @Override
00026     protected void onCreate(@Nullable Bundle savedInstanceState) {
00027         super.onCreate(savedInstanceState);
00028         setContentView(R.layout.activity_menu);
00029         Log.d(TAG, "onCreate: started");
00030
00031     }
00032
00033 }
```

9.9 Référence du fichier README.md

9.10 README.md

```
00001 \mainpage Le projet
00002
00003 \tableofcontents
00004
00005 Système autonome permettant de connaître à distance certains paramètres d'une ruche afin d'assurer son
       suivi et d'évaluer la santé des abeilles.
00006
00007 Version : PC (*mobile*)
00008
00009 \section section_tdm Table des matières
00010 - \ref page_README
00011 - \ref page_changelog
00012 - \ref page_about
00013 - \ref page_licence
00014
00015 \section section_infos Informations
00016
00017 \author Ethan Villesseche <villesseche.ethan@gmail.com>
00018 \date 2020
00019 \version 0.2
00020 \see https://svn.riouxsvn.com/bee-honey-t
00021
00022
00023 \page page_README README
00024
00025 [TOC]
00026
00027 # Projet {#projet}
00028
00029 ## Présentation {#presentation}
00030
00031 Système autonome permettant de connaître à distance certains paramètres d'une ruche afin d'assurer son
       suivi et d'évaluer la santé des abeilles.
00032
00033 Version : PC (*mobile*)
00034
00035 ## Base de données {#bdd}
00036
00037 ~~~ {.sql}
00038
00039 ~~~
00040
00041 ## Recette de l'application PC (mobile) {#recette}
00042
00043 * Consulter les données d'une ruche
00044 * Recevoir les données actuelles des ruches (MQTT/Json)
00045
00046 ## Informations {#informations}
00047
00048 \author Ethan Villesseche <villesseche.ethan@gmail.com>
00049 \date 2020
00050 \version 0.2
00051 \see https://svn.riouxsvn.com/bee-honey-t
00052
00053
```

```
00054 \page page_about A propos
00055
00056 \author Ethan Villesseche <villesseche.ethan@gmail.com>
00057 \date 2020
00058 \version 0.2
00059 \see https://svn.riouxsvn.com/bee-honey-t
00060
00061
00062 \page page_licence Licence GPL
00063
00064 This program is free software; you can redistribute it and/or modify
00065 it under the terms of the GNU General Public License as published by
00066 the Free Software Foundation; either version 2 of the License, or
00067 (at your option) any later version.
00068
00069 This program is distributed in the hope that it will be useful,
00070 but WITHOUT ANY WARRANTY; without even the implied warranty of
00071 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00072 GNU General Public License for more details.
00073
00074 You should have received a copy of the GNU General Public License
00075 along with this program; if not, write to the Free Software
00076 Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```

9.11 Référence du fichier Ruche.java

Déclaration de la classe Ruche.

Classes

— class com.lasalle.beehoneyt.Ruche
Déclaration de la classe Ruche.

Paquetages

— package com.lasalle.beehoneyt

9.11.1 Description détaillée

Déclaration de la classe Ruche.

Auteur

Ethan VILLESSECHE

Définition dans le fichier Ruche.java.

9.12 Ruche.java

```
00001 package com.lasalle.beehoneyt;
00002
00009 import android.util.Log;
00010 import java.io.Serializable;
00011
00016 public class Ruche implements Serializable
00017 {
00021     private static final String TAG = "Ruche";
00025     private String nom;
00026     private String deviceID;
00027     private String infos;
00028     private int poids;
00029     boolean abonne;
00030
00038     public Ruche(String nom, String deviceID)
00039     {
00040         Log.d(TAG, "Ruche() nom = " + nom + " deviceID = " + deviceID + " instance = " + this);
```

```
00041         this.nom = nom;
00042         this.deviceID = deviceID;
00043         this.abonne = false;
00044     }
00045
00052     public void setNom(String nom)
00053     {
00054         this.nom = nom;
00055     }
00056
00062     public void setDeviceID(String deviceID)
00063     {
00064         this.deviceID = deviceID;
00065     }
00066
00072     public void setInfos(String Infos)
00073     {
00074         this.infos = Infos;
00075     }
00076
00082     public void setPoids(int Poids)
00083     {
00084         this.poids = Poids;
00085     }
00086
00087
00094     public String getNom()
00095     {
00096         return nom;
00097     }
00098
00105     public String getDeviceID()
00106     {
00107         return deviceID;
00108     }
00109
00116     public String getInfos()
00117     {
00118         return infos;
00119     }
00120
00127     public int getPoids()
00128     {
00129         return poids;
00130     }
00131
00137     public boolean estAbonne()
00138     {
00139         return abonne;
00140     }
00141
00146     public void souscrireTopic()
00147     {
00148         if(!abonne)
00149         {
00150             CommunicationMQTT.souscrireTopic(deviceID);
00151             abonne = true;
00152             Log.d(TAG, "souscrireTopic() topic = " + deviceID);
00153         }
00154     }
00155 }
```

9.13 Référence du fichier RucheActivity.java

Déclaration de la classe RucheActivity.

Classes

— class com.lasalle.beehoneyt.RucheActivity
Déclaration de la classe RucheActivity.

Paquetages

— package com.lasalle.beehoneyt

9.13.1 Description détaillée

Déclaration de la classe RucheActivity.

Auteur

Ethan VILLESSECHE

Définition dans le fichier RucheActivity.java.

9.14 RucheActivity.java

```
00001 package com.lasalle.beehoneyt;
00002
00009 import androidx.appcompat.app.AppCompatActivity;
00010
00011 import android.content.Intent;
00012 import android.os.Bundle;
00013 import android.util.Log;
00014 import android.widget.TextView;
00015
00016 import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
00017 import org.eclipse.paho.client.mqttv3.MqttCallbackExtended;
00018 import org.eclipse.paho.client.mqttv3.MqttMessage;
00019 import org.json.JSONException;
00020 import org.json.JSONObject;
00021
00022 import java.text.ParseException;
00023 import java.text.SimpleDateFormat;
00024 import java.util.Date;
00025 import java.util.Iterator;
00026 import java.util.Locale;
00027 import java.util.TimeZone;
00028
00034 public class RucheActivity extends AppCompatActivity
00035 {
00039     private static final String TAG = "RucheActivity";
00040
00044     private Ruche ruche = null;
00045     private TextView nom_ruche;
00046     private TextView topic_ruche;
00047     private TextView message_ruche;
00048
00049     static private TextView poids_ruche;
00050     static private TextView temperature_interieure_ruche;
00051     static private TextView temperature_exterieure_ruche;
00052     static private TextView humidite_interieure_ruche;
00053     static private TextView humidite_exterieure_ruche;
00054     static private TextView pression_ruche;
00055     static private TextView ensoleillement_ruche;
00056
00057     @Override
00058     protected void onCreate(Bundle savedInstanceState)
00059     {
00060         super.onCreate(savedInstanceState);
00061         setContentView(R.layout.activity_ruche);
00062
00063         nom_ruche = (TextView) this.findViewById(R.id.nom_ruche);
00064         topic_ruche = (TextView) this.findViewById(R.id.topic_ruche);
00065         message_ruche = (TextView) this.findViewById(R.id.message_ruche);
00066         poids_ruche = (TextView) this.findViewById(R.id.poids_ruche);
00067         temperature_interieure_ruche = (TextView) this.findViewById(R.id.temperature_interieure_ruche);
00068         temperature_exterieure_ruche = (TextView) this.findViewById(R.id.temperature_exterieure_ruche);
00069         humidite_interieure_ruche = (TextView) this.findViewById(R.id.humidite_interieure_ruche);
00070         humidite_exterieure_ruche = (TextView) this.findViewById(R.id.humidite_exterieure_ruche);
00071         pression_ruche = (TextView) this.findViewById(R.id.pression_ruche);
00072         ensoleillement_ruche = (TextView) this.findViewById(R.id.ensoleillement_ruche);
00073
00074
00075
00076         Intent intent = getIntent();
00077         // Attention : un nouvel objet Ruche est créé
00078         ruche = (Ruche)intent.getSerializableExtra("Ruche");
00079         if(ruche != null)
00080         {
00081             Log.d(TAG, "Ruche : " + ruche.getNom() + " " + ruche);
00082             afficherInfo();
00083             communiquerTTN(ruche.getDeviceID());
00084         }
00085     }
00086
```

```
00091     private void afficherInfo()
00092     {
00093         afficherNom(ruche.getNom());
00094         afficherTopic(ruche.getDeviceID());
00095     }
00096
00101     public void afficherNom(String message)
00102     {
00103         nom_ruche.setText("Nom : " + message);
00104     }
00105
00113     public void afficherTopic(String message)
00114     {
00115         topic_ruche.setText("Topic : " + message);
00116     }
00117
00122     public void afficherJSON(String message)
00123     {
00124         message_ruche.setText("Message : " + message);
00125     }
00126
00132     private void communiquerTTN(final String deviceID)
00133     {
00134         Log.d(TAG, "deviceID : " + deviceID);
00135         CommunicationMQTT.setCallback(new MqttCallbackExtended()
00136         {
00137             @Override
00138             public void connectComplete(boolean b, String s) {
00139
00140             }
00141             @Override
00142             public void connectionLost(Throwable throwable) {
00143
00144             }
00145             @Override
00146             public void messageArrived(String topic, MqttMessage mqttMessage) throws Exception
00147             {
00148                 Log.d(TAG, "Topic : " + topic);
00149                 Log.d(TAG, "Message : " + mqttMessage.toString());
00150                 if(estBonneRuche(deviceID, mqttMessage.toString()))
00151                 {
00152                     afficherJSON(mqttMessage.toString());
00153                     CommunicationMQTT.decoderMessage(mqttMessage.toString(),
      ruche);
00154                     String horodatage = CommunicationMQTT.
      extraireHorodatage(mqttMessage.toString());
00155                     String horodatageFormate = formaterHorodatge(horodatage);
00156                     Log.d(TAG, "Horodatage formaté : " + horodatageFormate);
00157                 }
00158
00159             }
00160
00161             @Override
00162             public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken)
00163             {
00164             }
00165         });
00166     }
00167
00174     private boolean estBonneRuche(String deviceID,String message)
00175     {
00176         try
00177         {
00178             JSONObject json = null;
00179             Iterator<String> it = null;
00180
00181             json = new JSONObject(message);
00182             int port = json.getInt("port");
00183
00184             it = json.keys();
00185             while (it.hasNext())
00186             {
00187                 String cle = it.next();
00188                 Log.i(TAG, "clé = " + cle);
00189                 Log.i(TAG, "valeur = " + json.getString(cle));
00190                 //Log.i(TAG, "type = " + json.get(cle).getClass());
00191
00192                 if (cle.equals("dev_id"))
00193                 {
00194                     if(json.getString(cle).equals(deviceID)) {
00195                         return true;
00196                     }
00197                 }
00198             }
00199             return false;
00200         }
00201         catch (JSONException e)
00202         {
00203             e.printStackTrace();
00204             return false;
00205         }
```

```
00206     }
00207
00213     private String formaterHorodatge(String horodatage)
00214     {
00218         SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss", Locale.FRENCH);
00219         df.setTimeZone(TimeZone.getTimeZone("UTC"));
00220         Date date = null;
00221         String horodatageFormate = horodatage;
00222         try
00223         {
00224             date = df.parse(horodatage);
00225             df.setTimeZone(TimeZone.getDefault());
00226             horodatageFormate = df.format(date);
00227         }
00228         catch (ParseException e)
00229         {
00230             e.printStackTrace();
00231         }
00232
00233         return horodatageFormate;
00234     }
00235
00240     static public void afficherPoids(int message)
00241     {
00242         poids_ruche.setText("Poids : " + message + "kg");
00243     }
00244     public void setPoids(int poids)
00245     {
00246         ruche.setPoids(poids);
00247     }
00252     static public void afficherTemperatureInterieure(String message)
00253     {
00254         temperature_interieure_ruche.setText("Temperature interieure : " + message + " °C");
00255     }
00260     static public void afficherTemperatureExterieure(String message)
00261     {
00262         temperature_exterieure_ruche.setText("Temperature exterieure : " + message + " °C");
00263     }
00268     static public void afficherHumiditeInterieure(String message)
00269     {
00270         humidite_interieure_ruche.setText("Humidité interieure : " + message + " %");
00271     }
00276     static public void afficherHumiditerExterieure(String message)
00277     {
00278         humidite_exterieure_ruche.setText("Humidité exterieure : " + message+ " %");
00279     }
00284     static public void afficherPression(String message)
00285     {
00286         pression_ruche.setText("Pression atmosphérique : " + message + " hPa");
00287     }
00292     static public void afficherEnsoleillement(String message)
00293     {
00294         ensoleillement_ruche.setText("Ensoleillement : " + message + " lux");
00295     }
00296
00297 }
00298
```

9.15 Référence du fichier RucheAdapter.java

Déclaration de la classe RucheAdapter.

Classes

— class com.lasalle.beehoneyt.RucheAdapter
Déclaration de la classe RucheAdapter.

Paquetages

— package com.lasalle.beehoneyt

9.15.1 Description détaillée

Déclaration de la classe RucheAdapter.

Auteur

Ethan VILLESSECHE

Définition dans le fichier RucheAdapter.java.

9.16 RucheAdapter.java

```
00001 package com.lasalle.beehoneyt;
00002
00009 import android.content.Context;
00010 import android.content.Intent;
00011 import android.util.Log;
00012 import android.view.ContextThemeWrapper;
00013 import android.view.LayoutInflater;
00014 import android.view.View;
00015 import android.view.ViewGroup;
00016 import android.widget.Toast;
00017
00018 import androidx.annotation.NonNull;
00019 import androidx.recyclerview.widget.RecyclerView;
00020
00021 import java.io.Serializable;
00022 import java.lang.reflect.Array;
00023 import java.util.List;
00024
00030 public class RucheAdapter extends RecyclerView.Adapter<RuchesViewHolder>
00031 {
00032     private static final String TAG = "RucheAdapter";
00033     private  Context mContext;
00034     private List<Ruche> ruches = null;
00035
00036     public RucheAdapter(Context context, List<Ruche> ruches)
00037     {
00038         if(ruches != null)
00039         {
00040             this.ruches = ruches;
00041             mContext = context;
00042         }
00043     }
00044
00045     @Override
00046     public RuchesViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
      int viewType)
00047     {
00048         LayoutInflater inflater = LayoutInflater.from(parent.getContext());
00049         View view = inflater.inflate(R.layout.ruche, parent, false);
00050         return new RuchesViewHolder(view);
00051     }
00052
00053     @Override
00054     public void onBindViewHolder(@NonNull RuchesViewHolder holder, final
     int position)
00055     {
00056         Log.d(TAG, "onBindViewHolder: appel.");
00057
00058         final Ruche ruche = ruches.get(position);
00059         holder.afficher(ruche);
00060
00061         holder.cardview.setOnClickListener(new View.OnClickListener(){
00062             @Override
00063             public  void onClick(View view) {
00064                 Log.d(TAG, "onClick: click sur : " + ruche);
00065                 Toast.makeText(mContext, ruche.getNom(), Toast.LENGTH_SHORT).show();
00066
00067                 ruche.souscrireTopic();
00068                 Intent intent = new Intent(mContext, RucheActivity.class);
00069                 intent.putExtra("Ruche", (Serializable) ruche);
00070                 mContext.startActivity(intent);
00071             }
00072         });
00073     }
00074
00075     @Override
00076     public int getItemCount()
00077     {
00078         if(ruches != null)
00079             return ruches.size();
00080         return 0;
00081     }
00082 }
```

9.17 Référence du fichier RuchesViewHolder.java

Déclaration de la classe RuchesViewHolder.

Classes

— class com.lasalle.beehoneyt.RuchesViewHolder
Déclaration de la classe RuchesViewHolder.

Paquetages

— package com.lasalle.beehoneyt

9.17.1 Description détaillée

Déclaration de la classe RuchesViewHolder.

Auteur

Ethan VILLESSECHE

Définition dans le fichier RuchesViewHolder.java.

9.18 RuchesViewHolder.java

```
00001 package com.lasalle.beehoneyt;
00002
00009 import android.util.Log;
00010 import android.view.View;
00011 import android.widget.TextView;
00012 import androidx.cardview.widget.CardView;
00013 import androidx.recyclerview.widget.RecyclerView;
00014 import de.hdodenhof.circleimageview.CircleImageView;
00015
00022 public class RuchesViewHolder extends RecyclerView.ViewHolder
00023 {
00024     private static final String TAG = "RuchesViewHolder";
00025
00026     final TextView nom;
00027     final TextView info;
00028     //final TextView poids;
00029     CircleImageView image;
00030     CardView cardview;
00031     private Ruche ruche;
00032
00033     public RuchesViewHolder(final View itemView)
00034     {
00035         super(itemView);
00036
00037         image = itemView.findViewById(R.id.circle_image);
00038         nom = ((TextView) itemView.findViewById(R.id.nom));
00039         info = ((TextView) itemView.findViewById(R.id.info));
00040        // poids = ((TextView) itemView.findViewById(R.id.poids));
00041         cardview = itemView.findViewById(R.id.card_view);
00042     }
00043
00044     public void afficher(Ruche ruche)
00045     {
00046         this.ruche = ruche;
00047         nom.setText(ruche.getNom());
00048         info.setText(ruche.getInfos());
00049         //poids.setText(ruche.getPoids() + " Kg");
00050     }
00051 }
```

Index