GrOOm

version 0.2

BTS SNIR LaSalle Avignon 2020

Table des matières

1 Le projet

Le portier connecté « grOOm » permettra à l'occupant du bureau de communiquer sa disponibilité avec des personnes extérieures (visiteurs). Pour cela, il utilise une application soit en version PC soit mobile.

1.1 Table des matières

— README
— Changelog
— A propos
— Licence GPL

1.2 Informations

Auteur

Grégory Eyraud <gregory.eyraud@gmail.com>

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/groom/

2 Changelog

r56 | tvaira | 2020-03-27 16 :51 :01 +0100 (ven. 27 mars 2020) | 1 ligne

Validation émission à la connexion

r55 | tvaira | 2020-03-27 16 :15 :31 +0100 (ven. 27 mars 2020) | 1 ligne

Validation réception de trames

r54 | tvaira | 2020-03-27 15 :58 :46 +0100 (ven. 27 mars 2020) | 1 ligne

Validation émission de trames

r53 | geyraud | 2020-03-27 13 :25 :39 +0100 (ven. 27 mars 2020) | 1 ligne

envoie et reception de la trame + notification si quelqu'un a sonné ou si quelqu'un est présent

r52 | tvaira | 2020-03-26 09 :47 :26 +0100 (jeu. 26 mars 2020) | 1 ligne

r51 | geyraud | 2020-03-25 16 :00 :45 +0100 (mer. 25 mars 2020) | 1 ligne

Début communication Bluetooth

r50 | tvaira | 2020-03-18 08 :02 :03 +0100 (mer. 18 mars 2020) | 1 ligne

fichier à ignorer

r49 | tvaira | 2020-03-18 08 :00 :23 +0100 (mer. 18 mars 2020) | 1 ligne

fichier à ignorer

r47 | geyraud | 2020-03-13 16 :44 :35 +0100 (ven. 13 mars 2020) | 1 ligne

Affichage de la liste des appareils bluetooth appairés

r42 | geyraud | 2020-03-12 12 :28 :21 +0100 (jeu. 12 mars 2020) | 1 ligne

Ajout de la classe Communication.java

r41 | geyraud | 2020-03-12 12 :25 :06 +0100 (jeu. 12 mars 2020) | 1 ligne

Changement du nom de la classe Connexion.java en Communication.java

r38 | geyraud | 2020-03-11 15 :27 :34 +0100 (mer. 11 mars 2020) | 1 ligne

Sauvegarde des changements des attributs de l'objet groom de IHMGroom dans l'obejt groom de IHMConnexion après l'appel de finish()

r36 | geyraud | 2020-03-10 16 :51 :36 +0100 (mar. 10 mars 2020) | 1 ligne

Documentation du code avec doxygen

r35 | tvaira | 2020-03-08 11 :15 :51 +0100 (dim. 08 mars 2020) | 1 ligne

Révision de code

r34 | geyraud | 2020-03-06 17 :05 :38 +0100 (ven. 06 mars 2020) | 1 ligne

Ajout de la classe Groom et affichage de la dernière disponibilité choisie

r32 | geyraud | 2020-03-06 15 :58 :37 +0100 (ven. 06 mars 2020) | 1 ligne

Suppression de l'activité MainActivity

r31 | geyraud | 2020-03-06 14 :54 :59 +0100 (ven. 06 mars 2020) | 1 ligne

Ajout de la classe Communication, des nouvelles activités qui n'étaient pas envoyées, et des fichiers design des activités correspondantes

r27 | geyraud | 2020-03-05 16 :20 :43 +0100 (jeu. 05 mars 2020) | 1 ligne

Création d'une classe IHMConnexion

r23 | geyraud | 2020-03-04 16 :12 :56 +0100 (mer. 04 mars 2020) | 1 ligne

Création de l'application Groom

3 README

3.1 Projet

3.1.1 Présentation

Le portier connecté « grOOm » permettra à l'occupant du bureau de communiquer sa disponibilité avec des personnes extérieures (visiteurs). Pour cela, il utilise une application soit en version PC soit mobile.

Tout en s'intégrant facilement à l'environnement, il résout le manque d'interface entre les utilisateurs et les bureaux permettant de travailler plus efficacement.

A partir de l'application, l'occupant informe le visiteur de son état : "Libre", "Occupé" ou "Absent". Il aura la possibilité d'ajouter un message "libre" qui s'affichera alors sur l'écran du portier. S'il le souhaite, il informera le visiteur que celui-ci peut "Entrer" (ou "Entrez"). L'occupant dispose d'un mode "Sonnette" qu'il peut activer sur le portier connecté. Dans ce mode, le visiteur pourra sonner via l'écran tactile et prévenir l'occupant de sa présence.

Version : Mobile Android

3.1.2 Informations

Auteur

Grégory Eyraud gregory.eyraud@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/groom/

4 A propos

Auteur

Grégory Eyraud gregory.eyraud@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/groom/

5 Licence GPL

6 Documentation des espaces de nommage

6.1 Paquetage com

Paquetages

— package example

6.2 Paquetage com.example

Paquetages

— package groom

6.3 Paquetage com.example.groom

Classes

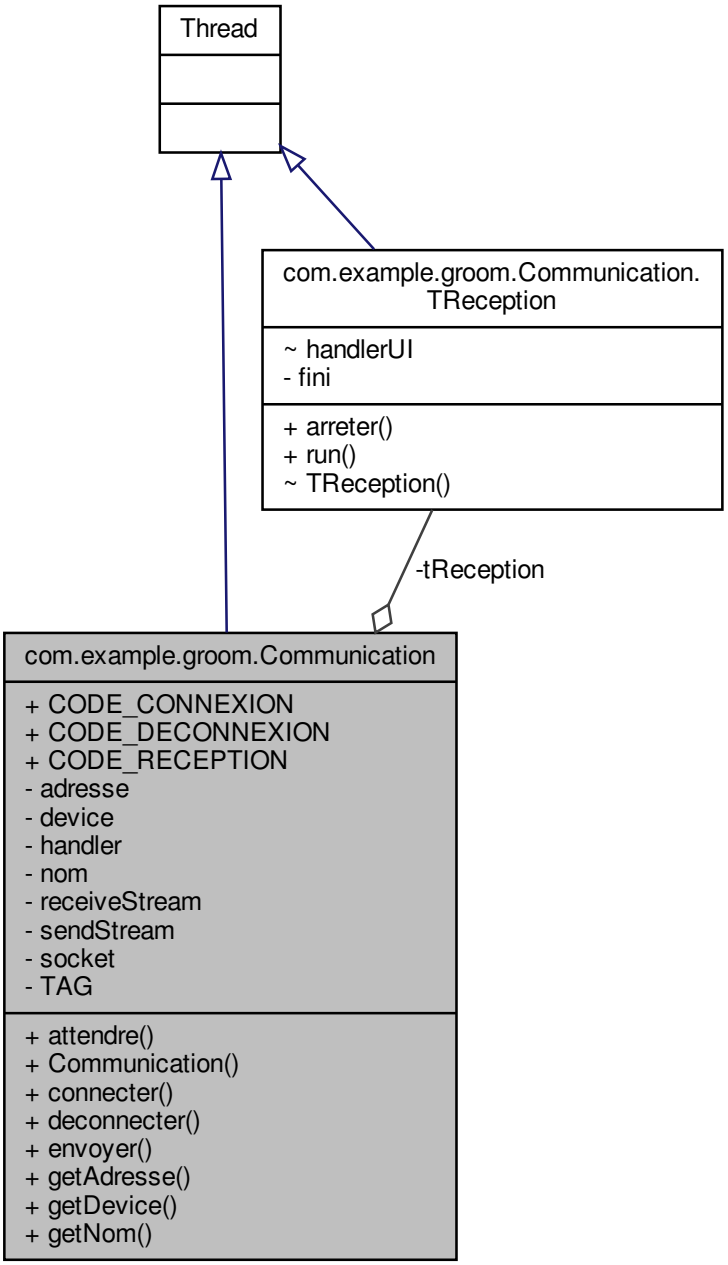
— class Communication
Déclaration de la classe Communication.
— class Groom
Déclaration de la classe Groom.
— class GroomBDD
— class GroomSQLite
— class IHMConnexion
Déclaration de la classe IHMConnexion.
— class IHMGroom
Déclaration de la classe IHMGroom.
— class Occupant
— class Occupants
— class Preference
— class Preferences

7 Documentation des classes

7.1 Référence de la classe com.example.groom.Communication

Déclaration de la classe Communication.

Graphe de collaboration de com.example.groom.Communication :



Classes

— class TReception
Déclaration de la classe TReception.

Fonctions membres publiques

- void attendre (int millis)
  Méthode pour attendre avant de passer à la prochaine ligne d'exécution.
- Communication (BluetoothDevice device, Handler handler)
  Constructeur de la classe Communication.
- void connecter ()
  Méthode pour se connecter à l'appareil.
- boolean deconnecter ()
  Méthode pour se déconnecter de l'appareil.
- void envoyer (String data)
  Méthode pour envoyer une trame.
- String getAdresse ()
  Accesseur get de l'adresse.
- BluetoothDevice getDevice ()
  Accesseur get de l'appareil.
- String getNom ()
  Accesseur get du nom.

Attributs publics statiques

- static final int CODE_CONNEXION = 0
  Code de connexion.
- static final int CODE_DECONNEXION = 2
  Code de déconnexion.
- static final int CODE_RECEPTION = 1
  Code de réception d'une trame.

Attributs privés

- String adresse
  L'adresse.
- BluetoothDevice device
  L'objet device.
- Handler handler
  L'objet handler.
- String nom
  Le nom.
- InputStream receiveStream = null
  L'objet receiveStream.
- OutputStream sendStream = null
  L'objet sendStream.
- BluetoothSocket socket = null
  L'objet socket.
- TReception tReception
  L'objet tReception.

Attributs privés statiques

- static final String TAG = "Communication"
  TAG pour les logs.

7.1.1 Description détaillée

Déclaration de la classe Communication.

Définition à la ligne 25 du fichier Communication.java.

7.1.2 Documentation des constructeurs et destructeur

7.1.2.1 Communication()

```
com.example.groom.Communication.Communication (
            BluetoothDevice device,
            Handler handler )
```

Constructeur de la classe Communication.

Paramètres

device	L'appareil où on va se connecter
handler	L'handler

Définition à la ligne 53 du fichier Communication.java.

Références com.example.groom.Communication.device, com.example.groom.Communication.getNom(), et com.example.groom.↩ Communication.handler.

```
00054     {
00055         this.handler = handler;
00056         if (device != null)
00057         {
00058             this.device = device;
00059             this.nom = device.getName();
00060             this.adresse = device.getAddress();
00061 
00062             try
00063             {
00064                 socket = device.createRfcommSocketToServiceRecord(UUID.fromString("
      00001101-0000-1000-8000-00805F9B34FB"));
00065                 receiveStream = socket.getInputStream();
00066                 sendStream = socket.getOutputStream();
00067             }
00068             catch (IOException e)
00069             {
00070                 e.printStackTrace();
00071             }
00072         }
00073         else
00074         {
00075             this.device = device;
00076             this.nom = "Aucun";
00077             this.adresse = "";
00078         }
00079 
00080         if(socket != null)
00081             tReception = new TReception(handler);
00082         Log.v(TAG, "Communication() : Nom = " + getNom());
00083     }
```

7.1.3 Documentation des fonctions membres

7.1.3.1 attendre()

```
void com.example.groom.Communication.attendre (
            int millis )
```

Méthode pour attendre avant de passer à la prochaine ligne d'exécution.

Paramètres

millis	le temps d'attent en ms

Définition à la ligne 217 du fichier Communication.java.

```
00218     {
00219         try
00220         {
00221             Thread.sleep(millis);
00222         }
00223         catch (InterruptedException e)
00224         {
00225             e.printStackTrace();
00226         }
00227     }
```

7.1.3.2 connecter()

`com.example.groom.Communication.connecter ( )`

Méthode pour se connecter à l'appareil.

Définition à la ligne 123 du fichier Communication.java.

Références com.example.groom.Communication.CODE_CONNEXION, et com.example.groom.Communication.getNom().

Référencé par com.example.groom.IHMGroom.onCreate().

```
00124     {
00125         new Thread()
00126         {
00127             @Override public void run()
00128             {
00129                 try
00130                 {
00131                     socket.connect();
00132 
00133                     Message msg = Message.obtain();
00134                     msg.what = CODE_CONNEXION;
00135                     handler.sendMessage(msg);
00136 
00137                     if(tReception != null)
00138                         tReception.start();
00139                     Log.v(TAG, "connecter() : Nom = " + getNom());
00140                 }
00141                 catch (IOException e)
00142                 {
00143                     Log.d(TAG, "Erreur connect()");
00144                     e.printStackTrace();
00145                 }
00146             }
00147         }.start();
00148     }
```

7.1.3.3 deconnecter()

`com.example.groom.Communication.deconnecter ( )`

Méthode pour se déconnecter de l'appareil.

Renvoie

boolean true si déconnecté, false si une erreur a été rencontrée

Définition à la ligne 156 du fichier Communication.java.

Références com.example.groom.Communication.TReception.arreter(), et com.example.groom.Communication.CODE_DECONNEXION.

Référencé par com.example.groom.IHMGroom.finish(), et com.example.groom.IHMGroom.onClick().

```
00157     {
00158         try
00159         {
00160             tReception.arreter();
00161 
00162             socket.close();
00163 
00164             Message msg = Message.obtain();
00165             msg.what = CODE_DECONNEXION;
00166             handler.sendMessage(msg);
00167 
00168             return true;
00169         }
00170         catch (IOException e)
00171         {
00172             Log.d(TAG, "Erreur close()");
00173             e.printStackTrace();
00174             return false;
00175         }
00176     }
```

7.1.3.4 envoyer()

```
com.example.groom.Communication.envoyer (
             String data )
```

Méthode pour envoyer une trame.

Paramètres

data	les données à envoyer

Définition à la ligne 184 du fichier Communication.java.

Référencé par com.example.groom.IHMGroom.initialiserSaisieMessagePerso(), et com.example.groom.IHMGroom.onClick().

```
00185     {
00186         final String trame = data;
00187 
00188         if(socket == null)
00189             return;
00190 
00191         new Thread()
00192         {
00193             @Override public void run()
00194             {
00195                 try
00196                 {
00197                     if(socket.isConnected())
00198                     {
00199                         sendStream.write(trame.getBytes());
00200                         sendStream.flush();
00201                         Log.d(TAG, "envoyer() trame : " + trame);
00202                     }
00203                 }
00204                 catch (IOException e)
00205                 {
```

```
00206                     Log.d(TAG, "Erreur write()");
00207                     e.printStackTrace();
00208                 }
00209             }
00210         }.start();
00211     }
```

7.1.3.5 getAdresse()

`com.example.groom.Communication.getAdresse ( )`

Accesseur get de l'adresse.

Renvoie

String l'adresse

Définition à la ligne 102 du fichier Communication.java.

Références com.example.groom.Communication.adresse.

```
00103     {
00104         return adresse;
00105     }
```

7.1.3.6 getDevice()

`com.example.groom.Communication.getDevice ( )`

Accesseur get de l'appareil.

Renvoie

BluetoothDevice l'appareil

Définition à la ligne 113 du fichier Communication.java.

Références com.example.groom.Communication.device.

```
00114     {
00115         return device;
00116     }
```

7.1.3.7 getNom()

`com.example.groom.Communication.getNom ( )`

Accesseur get du nom.

Renvoie

String le nom

Définition à la ligne 91 du fichier Communication.java.

Références com.example.groom.Communication.nom.

Référencé par com.example.groom.Communication.Communication(), et com.example.groom.Communication.connecter().

```
00092     {
00093         return nom;
00094     }
```

7.1.4 Documentation des données membres

7.1.4.1 adresse

```
String com.example.groom.Communication.adresse  [private]
```

L'adresse.

Définition à la ligne 40 du fichier Communication.java.

Référencé par com.example.groom.Communication.getAdresse().

7.1.4.2 CODE_CONNEXION

```
final int com.example.groom.Communication.CODE_CONNEXION = 0  [static]
```

Code de connexion.

Définition à la ligne 31 du fichier Communication.java.

Référencé par com.example.groom.Communication.connecter().

7.1.4.3 CODE_DECONNEXION

```
final int com.example.groom.Communication.CODE_DECONNEXION = 2  [static]
```

Code de déconnexion.

Définition à la ligne 33 du fichier Communication.java.

Référencé par com.example.groom.Communication.deconnecter().

7.1.4.4 CODE_RECEPTION

```
final int com.example.groom.Communication.CODE_RECEPTION = 1  [static]
```

Code de réception d'une trame.

Définition à la ligne 32 du fichier Communication.java.

Référencé par com.example.groom.Communication.TReception.run().

7.1.4.5 device

```
BluetoothDevice com.example.groom.Communication.device  [private]
```

L'objet device.

Attributs

Définition à la ligne 38 du fichier Communication.java.

Référencé par com.example.groom.Communication.Communication(), et com.example.groom.Communication.getDevice().

7.1.4.6 handler

```
Handler com.example.groom.Communication.handler  [private]
```

L'objet handler.

Définition à la ligne 41 du fichier Communication.java.

Référencé par com.example.groom.Communication.Communication().

7.1.4.7 nom

```
String com.example.groom.Communication.nom  [private]
```

Le nom.

Définition à la ligne 39 du fichier Communication.java.

Référencé par com.example.groom.Communication.getNom().

7.1.4.8 receiveStream

```
InputStream com.example.groom.Communication.receiveStream = null  [private]
```

L'objet receiveStream.

Définition à la ligne 43 du fichier Communication.java.

7.1.4.9 sendStream

```
OutputStream com.example.groom.Communication.sendStream = null  [private]
```

L'objet sendStream.

Définition à la ligne 44 du fichier Communication.java.

7.1.4.10 socket

```
BluetoothSocket com.example.groom.Communication.socket = null  [private]
```

L'objet socket.

Définition à la ligne 42 du fichier Communication.java.

7.1.4.11 TAG

```
final String com.example.groom.Communication.TAG = "Communication"  [static], [private]
```

TAG pour les logs.

Constantes

Définition à la ligne 30 du fichier Communication.java.

7.1.4.12 tReception

```
TReception com.example.groom.Communication.tReception  [private]
```

L'objet tReception.

Définition à la ligne 45 du fichier Communication.java.

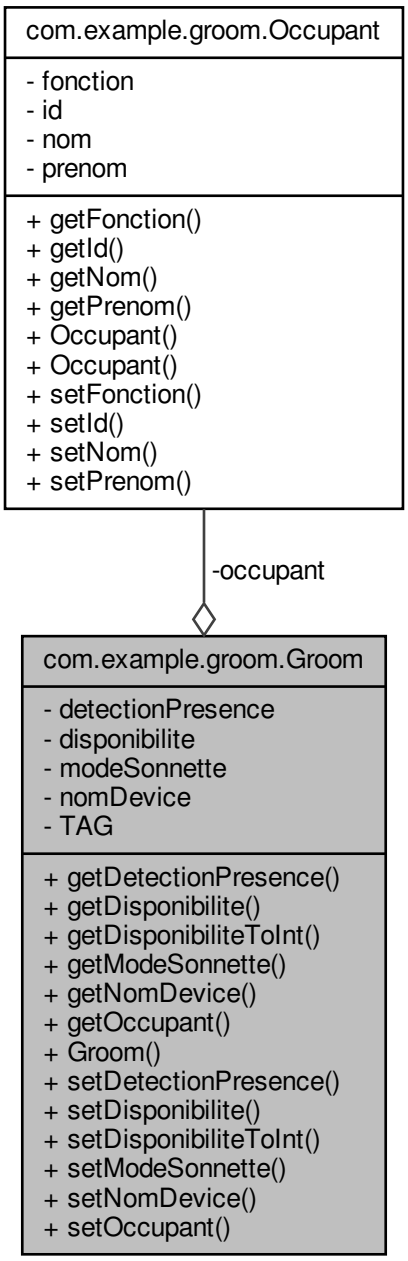
La documentation de cette classe a été générée à partir du fichier suivant :

— Communication.java

7.2 Référence de la classe com.example.groom.Groom

Déclaration de la classe Groom.

Graphe de collaboration de com.example.groom.Groom :



Fonctions membres publiques

- boolean getDetectionPresence ()
  Accesseur get de la détection de présence.
- String getDisponibilite ()
  Accesseur get de la disponibilité
- int getDisponibiliteToInt ()
  Retourne la disponibilité sous forme entière.
- boolean getModeSonnette ()
  Accesseur get du mode de la sonnette.
- String getNomDevice ()

Accesseur get du nomDevice.
- Occupant getOccupant ()
- Groom (String nom, String prenom, String fonction, String disponibilite, boolean modeSonnette, boolean detectionPresence)
  Constructeur de la classe Groom.
- void setDetectionPresence (boolean detectionPresence)
  Accesseur set de la détection de présence.
- void setDisponibilite (String disponibilite)
  Accesseur set de la disponibilité
- void setDisponibiliteToInt (int dispo)
  Fixe la disponibilité à partir de ss forme entière.
- void setModeSonnette (boolean modeSonnette)
  Accesseur set du mode de la sonnette.
- void setNomDevice (String nomDevice)
  Accesseur set du nomDevice.
- void setOccupant (Occupant occupant)

Attributs privés

- boolean detectionPresence
  Le mode détection de présence.
- String disponibilite
  La disponibilité
- boolean modeSonnette
  Le mode sonette.
- String nomDevice
  Le nom du périphérique Bluetooth.
- Occupant occupant

Attributs privés statiques

- static final String TAG = "Groom"
  TAG pour les logs.

7.2.1 Description détaillée

Déclaration de la classe Groom.

Définition à la ligne 15 du fichier Groom.java.

7.2.2 Documentation des constructeurs et destructeur

7.2.2.1 Groom()

```
com.example.groom.Groom.Groom (
            String nom,
            String prenom,
            String fonction,
            String disponibilite,
            boolean modeSonnette,
            boolean detectionPresence )
```

Constructeur de la classe Groom.

Paramètres

nom	Le nom de l'occupant
prenom	Le prénom de l'occupant
fonction	La fonction de l'occupant
disponibilite	La disponibilité de l'occupant
modeSonnette	Le mode sonnette
detectionPresence	Le mode détection de présence

Définition à la ligne 42 du fichier Groom.java.

Références com.example.groom.Groom.detectionPresence, com.example.groom.Groom.disponibilite, et com.example.groom.↩ Groom.modeSonnette.

```
00043     {
00044         occupant = new Occupant(nom, prenom , fonction);
00045         this.disponibilite = disponibilite;
00046         this.modeSonnette = modeSonnette;
00047         this.detectionPresence = detectionPresence;
00048     }
```

7.2.3 Documentation des fonctions membres

7.2.3.1 getDetectionPresence()

`com.example.groom.Groom.getDetectionPresence ( )`

Accesseur get de la détection de présence.

Renvoie

boolean la détection de présence : true = activée , false = désactivée

Définition à la ligne 146 du fichier Groom.java.

Références com.example.groom.Groom.detectionPresence.

Référencé par com.example.groom.IHMGroom.onClick().

```
00147     {
00148         return this.detectionPresence;
00149     }
```

7.2.3.2 getDisponibilite()

`com.example.groom.Groom.getDisponibilite ( )`

Accesseur get de la disponibilité

Renvoie

disponibilite la disponibilité

Définition à la ligne 66 du fichier Groom.java.

Références com.example.groom.Groom.disponibilite.

Référencé par com.example.groom.IHMGroom.decoderTrameRecue(), com.example.groom.IHMConnexion.onActivityResult(), com.example.groom.IHMGroom.onClick(), com.example.groom.IHMGroom.onCreate(), et com.example.groom.IHMGroom.onStart().

```
00067     {
00068         return this.disponibilite;
00069     }
```

7.2.3.3 getDisponibiliteToInt()

```
com.example.groom.Groom.getDisponibiliteToInt ( )
```

Retourne la disponibilité sous forme entière.

Renvoie

disponibilite la disponibilité sous forme de int

Définition à la ligne 77 du fichier Groom.java.

Référencé par com.example.groom.IHMGroom.onClick().

```
00078     {
00079         int dispo = 0;
00080 
00081         if(this.disponibilite.equals("Libre"))
00082             dispo = 0;
00083         else if(this.disponibilite.equals("Absent"))
00084             dispo = 1;
00085         else if(this.disponibilite.equals("Occupé"))
00086             dispo = 2;
00087 
00088         return dispo;
00089     }
```

7.2.3.4 getModeSonnette()

```
com.example.groom.Groom.getModeSonnette ( )
```

Accesseur get du mode de la sonnette.

Renvoie

boolean le mode de la sonnette : true = activée , false = désactivée

Définition à la ligne 124 du fichier Groom.java.

Références com.example.groom.Groom.modeSonnette.

Référencé par com.example.groom.IHMConnexion.onActivityResult(), com.example.groom.IHMGroom.onClick(), et com.example.groom.IHMGroom.verifierModeSonnette().

```
00125     {
00126         return this.modeSonnette;
00127     }
```

7.2.3.5 getNomDevice()

```
com.example.groom.Groom.getNomDevice ( )
```

Accesseur get du nomDevice.

Renvoie

String le nom du périphérique Bluetooth

Définition à la ligne 168 du fichier Groom.java.

Références com.example.groom.Groom.nomDevice.

Référencé par com.example.groom.IHMGroom.onCreate().

```
00169     {
00170         return this.nomDevice;
00171     }
```

7.2.3.6 getOccupant()

```
Occupant com.example.groom.Groom.getOccupant ( )
```

Définition à la ligne 50 du fichier Groom.java.

Références com.example.groom.Groom.occupant.

Référencé par com.example.groom.IHMConnexion.initialiserRetraitOccupant(), et com.example.groom.IHMGroom.onCreate().

```
00051     {
00052         return occupant;
00053     }
```

7.2.3.7 setDetectionPresence()

```
com.example.groom.Groom.setDetectionPresence (
            boolean detectionPresence )
```

Accesseur set de la détection de présence.

Paramètres

detectionPresence	

Définition à la ligne 157 du fichier Groom.java.

Références com.example.groom.Groom.detectionPresence.

```
00158     {
00159         this.detectionPresence = detectionPresence;
00160     }
```

7.2.3.8 setDisponibilite()

```
com.example.groom.Groom.setDisponibilite (
            String disponibilite )
```

Accesseur set de la disponibilité

Paramètres

disponibilite	la disponibilité

Définition à la ligne 113 du fichier Groom.java.

Références com.example.groom.Groom.disponibilite.

Référencé par com.example.groom.IHMGroom.onClick().

```
00114     {
00115         this.disponibilite = disponibilite;
00116     }
```

7.2.3.9 setDisponibiliteToInt()

```
com.example.groom.Groom.setDisponibiliteToInt (
             int dispo )
```

Fixe la disponibilité à partir de ss forme entière.

Paramètres

dispo	la disponibilité sous forme de int

Définition à la ligne 97 du fichier Groom.java.

Référencé par com.example.groom.IHMGroom.decoderTrameRecue().

```
00098     {
00099         if(dispo == 0)
00100             this.disponibilite = "Libre";
00101         else if(dispo == 1)
00102             this.disponibilite = "Absent";
00103         else if(dispo == 2)
00104             this.disponibilite = "Occupé";
00105     }
```

7.2.3.10 setModeSonnette()

```
com.example.groom.Groom.setModeSonnette (
            boolean modeSonnette )
```

Accesseur set du mode de la sonnette.

Paramètres

modeSonnette	

Définition à la ligne 135 du fichier Groom.java.

Références com.example.groom.Groom.modeSonnette.

Référencé par com.example.groom.IHMGroom.onClick().

```
00136     {
00137         this.modeSonnette = modeSonnette;
00138     }
```

7.2.3.11 setNomDevice()

```
com.example.groom.Groom.setNomDevice (
            String nomDevice )
```

Accesseur set du nomDevice.

Paramètres

nomDevice	le nom du périphérique Bluetooth

Définition à la ligne 179 du fichier Groom.java.

Références com.example.groom.Groom.nomDevice.

Référencé par com.example.groom.IHMConnexion.selectionnerGroom().

```
00180     {
00181         this.nomDevice = nomDevice;
00182     }
```

7.2.3.12 setOccupant()

```
void com.example.groom.Groom.setOccupant (
            Occupant occupant )
```

Définition à la ligne 55 du fichier Groom.java.

Références com.example.groom.Groom.occupant.

Référencé par com.example.groom.IHMConnexion.selectionnerOccupant().

```
00056     {
00057         this.occupant = occupant;
00058     }
```

7.2.4 Documentation des données membres

7.2.4.1 detectionPresence

```
boolean com.example.groom.Groom.detectionPresence  [private]
```

Le mode détection de présence.

Définition à la ligne 28 du fichier Groom.java.

Référencé par com.example.groom.Groom.getDetectionPresence(), com.example.groom.Groom.Groom(), et com.example.groom.↩ Groom.setDetectionPresence().

7.2.4.2 disponibilite

```
String com.example.groom.Groom.disponibilite  [private]
```

La disponibilité

Définition à la ligne 26 du fichier Groom.java.

Référencé par com.example.groom.Groom.getDisponibilite(), com.example.groom.Groom.Groom(), et com.example.groom.Groom.↩ setDisponibilite().

7.2.4.3 modeSonnette

```
boolean com.example.groom.Groom.modeSonnette  [private]
```

Le mode sonette.

Définition à la ligne 27 du fichier Groom.java.

Référencé par com.example.groom.Groom.getModeSonnette(), com.example.groom.Groom.Groom(), et com.example.groom.↩ Groom.setModeSonnette().

7.2.4.4 nomDevice

```
String com.example.groom.Groom.nomDevice  [private]
```

Le nom du périphérique Bluetooth.

Définition à la ligne 29 du fichier Groom.java.

Référencé par com.example.groom.Groom.getNomDevice(), et com.example.groom.Groom.setNomDevice().

7.2.4.5 occupant

```
Occupant com.example.groom.Groom.occupant  [private]
```

Attributs

Définition à la ligne 25 du fichier Groom.java.

Référencé par com.example.groom.Groom.getOccupant(), et com.example.groom.Groom.setOccupant().

7.2.4.6 TAG

```
final String com.example.groom.Groom.TAG = "Groom"  [static], [private]
```

TAG pour les logs.

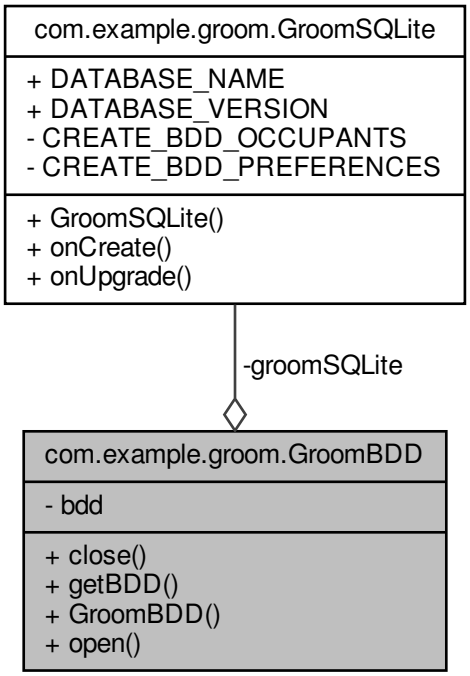
Constantes

Définition à la ligne 20 du fichier Groom.java.

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

7.3 Référence de la classe com.example.groom.GroomBDD

Graphe de collaboration de com.example.groom.GroomBDD :



Fonctions membres publiques

- void close ()
- SQLiteDatabase getBDD ()
- GroomBDD (Context context)
- void open ()

Attributs privés

- SQLiteDatabase bdd = null
- GroomSQLite groomSQLite = null

7.3.1 Description détaillée

Définition à la ligne 6 du fichier GroomBDD.java.

7.3.2 Documentation des constructeurs et destructeur

7.3.2.1 GroomBDD()

```
com.example.groom.GroomBDD.GroomBDD (
            Context context )
```

Définition à la ligne 11 du fichier GroomBDD.java.

```
00012     {
00013         groomSQLite = new GroomSQLite(context);
00014     }
```

7.3.3 Documentation des fonctions membres

7.3.3.1 close()

```
void com.example.groom.GroomBDD.close ( )
```

Définition à la ligne 22 du fichier GroomBDD.java.

```
00023     {
00024         if (bdd != null)
00025             if (bdd.isOpen())
00026                 bdd.close();
00027     }
```

7.3.3.2 getBDD()

```
SQLiteDatabase com.example.groom.GroomBDD.getBDD ( )
```

Définition à la ligne 29 du fichier GroomBDD.java.

Références com.example.groom.GroomBDD.bdd, et com.example.groom.GroomBDD.open().

Référencé par com.example.groom.Occupants.Occupants(), et com.example.groom.Preferences.Preferences().

```
00030     {
00031         if (bdd == null)
00032             open();
00033         return bdd;
00034     }
```

7.3.3.3 open()

```
void com.example.groom.GroomBDD.open ( )
```

Définition à la ligne 16 du fichier GroomBDD.java.

Référencé par com.example.groom.GroomBDD.getBDD(), com.example.groom.Occupants.Occupants(), et com.example.groom.↩ Preferences.Preferences().

```
00017     {
00018         if (bdd == null)
00019             bdd = groomSQLite.getWritableDatabase();
00020     }
```

7.3.4 Documentation des données membres

7.3.4.1 bdd

```
SQLiteDatabase com.example.groom.GroomBDD.bdd = null  [private]
```

Définition à la ligne 8 du fichier GroomBDD.java.

Référencé par com.example.groom.GroomBDD.getBDD().

7.3.4.2 groomSQLite

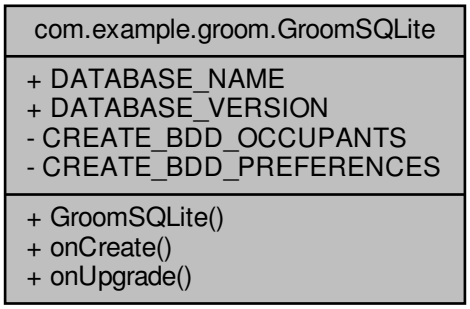
```
GroomSQLite com.example.groom.GroomBDD.groomSQLite = null  [private]
```

Définition à la ligne 9 du fichier GroomBDD.java.

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

7.4 Référence de la classe com.example.groom.GroomSQLite

Graphe de collaboration de com.example.groom.GroomSQLite :



Fonctions membres publiques

- GroomSQLite (Context context)
- void onCreate (SQLiteDatabase db)
- void onUpgrade (SQLiteDatabase db, int oldVersion, int newVersion)

Attributs publics statiques

- static final String DATABASE_NAME = "groom.db"
- static final int DATABASE_VERSION = 1

Attributs privés statiques

- static final String CREATE_BDD_OCCUPANTS
- static final String CREATE_BDD_PREFERENCES

7.4.1 Description détaillée

Définition à la ligne 7 du fichier GroomSQLite.java.

7.4.2 Documentation des constructeurs et destructeur

7.4.2.1 GroomSQLite()

```
com.example.groom.GroomSQLite.GroomSQLite (
            Context context )
```

Définition à la ligne 24 du fichier GroomSQLite.java.

```
00025     {
00026         super(context, DATABASE_NAME, null, DATABASE_VERSION);
00027     }
```

7.4.3 Documentation des fonctions membres

7.4.3.1 onCreate()

```
void com.example.groom.GroomSQLite.onCreate (
            SQLiteDatabase db )
```

Définition à la ligne 30 du fichier GroomSQLite.java.

```
00031     {
00032         db.execSQL(CREATE_BDD_OCCUPANTS);
00033         db.execSQL(CREATE_BDD_PREFERENCES);
00034     }
```

7.4.3.2 onUpgrade()

```
void com.example.groom.GroomSQLite.onUpgrade (
            SQLiteDatabase db,
            int oldVersion,
            int newVersion )
```

Définition à la ligne 37 du fichier GroomSQLite.java.

```
00038     {
00039         db.execSQL("DROP TABLE occupants;");
00040         db.execSQL("DROP TABLE preferences");
00041     }
```

7.4.4 Documentation des données membres

7.4.4.1 CREATE_BDD_OCCUPANTS

```
final String com.example.groom.GroomSQLite.CREATE_BDD_OCCUPANTS  [static], [private]
```

Valeur initiale :

```
=
            "CREATE TABLE occupants (" +
                    "idOccupant INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL," +
                    "nom VARCHAR(45) NULL," +
                    "prenom VARCHAR(45) NULL," +
                    "fonction VARCHAR(45) NULL);"
```

Définition à la ligne 11 du fichier GroomSQLite.java.

7.4.4.2 CREATE_BDD_PREFERENCES

```
final String com.example.groom.GroomSQLite.CREATE_BDD_PREFERENCES  [static], [private]
```

Valeur initiale :

```
=
            "CREATE TABLE preferences (" +
                    "idPreferences INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL," +
                    "groom VARCHAR(45) NULL," +
                    "idPrecedentOccupant INTEGER NOT NULL, " +
                    "FOREIGN KEY (idPrecedentOccupant) REFERENCES occupants (idOccupant);"
```

Définition à la ligne 17 du fichier GroomSQLite.java.

7.4.4.3 DATABASE_NAME

```
final String com.example.groom.GroomSQLite.DATABASE_NAME = "groom.db"  [static]
```

Définition à la ligne 9 du fichier GroomSQLite.java.

7.4.4.4 DATABASE_VERSION

```
final int com.example.groom.GroomSQLite.DATABASE_VERSION = 1  [static]
```

Définition à la ligne 10 du fichier GroomSQLite.java.

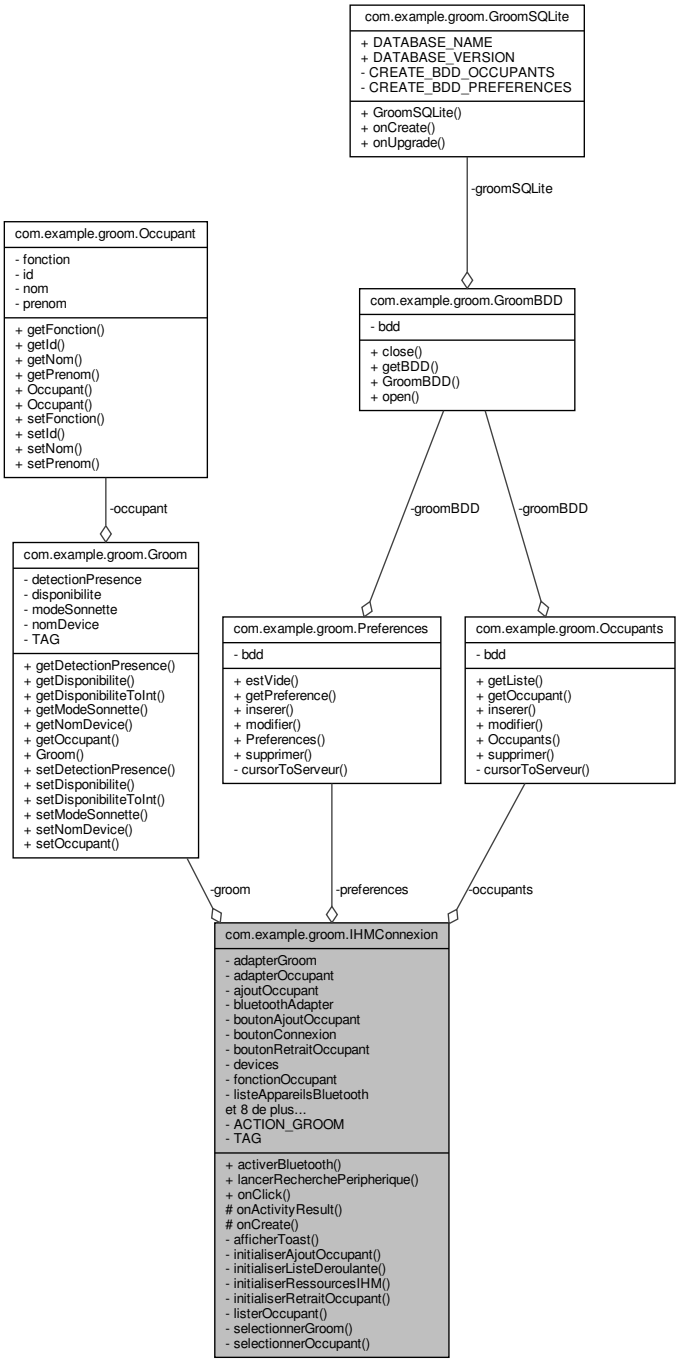
La documentation de cette classe a été générée à partir du fichier suivant :

— GroomSQLite.java

7.5 Référence de la classe com.example.groom.IHMConnexion

Déclaration de la classe IHMConnexion.

Graphe de collaboration de com.example.groom.IHMConnexion :



Fonctions membres publiques

- void activerBluetooth ()
  Méthode qui active le Bluetooth de l'appareil.
- void lancerRecherchePeripherique ()
  Méthode qui lance la recherche de périphérique.
- void onClick (View element)

Fonctions membres protégées

- void onActivityResult (int requestCode, int resultCode, Intent intent)

Méthode appelée à la fin de l'activité lancée et récupère l'objet groom envoyé

— void onCreate (Bundle savedInstanceState)

Méthode appelée à la création de l'activité IHMConnexion.

Fonctions membres privées

— void afficherToast (String message)

Méthode appelée pour donner des informations supplémentaires à l'utilisateur.

— void initialiserAjoutOccupant ()

Méthode appelée pour initialiser la boite de dialogue personnalisé pour l'ajout d'un occupant.

— void initialiserListeDeroulante ()

Méthode qui initialise la liste déroulante d'appareils Bluetooth appairés.

— void initialiserRessourcesIHM ()

Méthode qui initialise les ressources du layout de l'activité IHMConnexion.

— void initialiserRetraitOccupant ()

— void listerOccupant ()

— void selectionnerGroom ()

Méthode permet de sélectionner un portier Groom Bluetooth dans la liste.

— void selectionnerOccupant ()

Méthode permet de sélectionner un occupant dans la liste.

Attributs privés

— ArrayAdapter< String > adapterGroom

Adaptateur pour mettre la liste de noms des appareils dans la liste déroulante listeGroom.

— ArrayAdapter< String > adapterOccupant

Adaptateur pour mettre la liste des occupants dnas la liste déroulante listeOccupant.

— AlertDialog.Builder ajoutOccupant

Le builder qui permet de créer une fenêtre de dialogue d'ajout d'occupant.

— BluetoothAdapter bluetoothAdapter

L'objet BluetoothAdapter.

— Button boutonAjoutOccupant

Le bouton pour ajouter un occupant.

— Button boutonConnexion

Le bouton pour se connecter à l'appareil Groom.

— Button boutonRetraitOccupant

Le bouton pour retirer un occupant.

— Set< BluetoothDevice > devices

Conteneur qui liste les appareils bluetooth disponibles sans doublons.

— EditText fonctionOccupant

Le champ de texte pour écrire la fonction de l'occupnt.

— Groom groom = null

L'objet groom.

— List< BluetoothDevice > listeAppareilsBluetooth

Conteneur qui liste les appareils bluetooth disponibles.

— Spinner listeGroom

Liste déroulante des appareils bluetooth.

— List< String > listeNomsAppareilsBluetooth

Conteneur qui liste les noms des appareils bluetooth disponibles.

— Spinner listeOccupant

Liste déroulante des occupants.

— List< String > listeOccupants

Conteneur qui liste les occupants.

— EditText nomOccupant

Le champ de texte pour écrire le nom de l'occupant.

— Occupants occupants = null

— Preferences preferences = null

— EditText prenomOccupant

Le champ de texte pour écrire le prenom de l'occupant.

— AlertDialog.Builder retraitOccupant

Le builder qui permet de créer une fenêtre de dialogue de retrait d'occupant.

— Toast toast

Le toast qui permet d'afficher des informations à l'utilisateur.

Attributs privés statiques

— static final int ACTION_GROOM = 1

Code de requête lors du début et de la fin de l'activité IHMGroom.

— static final String TAG = "IHMConnexion"

TAG pour les logs.

7.5.1 Description détaillée

Déclaration de la classe IHMConnexion.

Définition à la ligne 34 du fichier IHMConnexion.java.

7.5.2 Documentation des fonctions membres

7.5.2.1 activerBluetooth()

```
com.example.groom.IHMConnexion.activerBluetooth ( )
```

Méthode qui active le Bluetooth de l'appareil.

Définition à la ligne 145 du fichier IHMConnexion.java.

Références com.example.groom.IHMConnexion.lancerRecherchePeripherique().

Référencé par com.example.groom.IHMConnexion.onCreate().

```
00146     {
00147         bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
00148         if (bluetoothAdapter == null)
00149         {
00150             Toast.makeText(getApplicationContext(), "Bluetooth non activé !", Toast.LENGTH_SHORT).show();
00151         }
00152         else
00153             {
00154             if (!bluetoothAdapter.isEnabled())
00155             {
00156                 Toast.makeText(getApplicationContext(), "Bluetooth non activé !", Toast.LENGTH_SHORT).show(
      );
00157                 bluetoothAdapter.enable();
00158             }
00159             else
00160                 {
00161                 Toast.makeText(getApplicationContext(), "Bluetooth activé", Toast.LENGTH_SHORT).show();
00162             }
00163         }
00164
00165         lancerRecherchePeripherique();
00166     }
```

7.5.2.2 afficherToast()

```
com.example.groom.IHMConnexion.afficherToast (
             String message )  [private]
```

Méthode appelée pour donner des informations supplémentaires à l'utilisateur.

Paramètres

message	le message à afficher

Définition à la ligne 296 du fichier IHMConnexion.java.

Référencé par com.example.groom.IHMConnexion.onClick().

```
00297     {
00298         toast = Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT);
00299         toast.show();
00300     }
```

7.5.2.3 initialiserAjoutOccupant()

`com.example.groom.IHMConnexion.initialiserAjoutOccupant ( )  [private]`

Méthode appelée pour initialiser la boite de dialogue personnalisé pour l'ajout d'un occupant.

Méthode appelée pour initialiser la boite de dialogue personnalisé pour le retrait d'un occupant.

Définition à la ligne 306 du fichier IHMConnexion.java.

Références com.example.groom.IHMConnexion.ajoutOccupant, com.example.groom.Occupants.inserer(), et com.example.groom.IHMConnexion.onClick().

Référencé par com.example.groom.IHMConnexion.initialiserRessourcesIHM().

```
00307     {
00308         ajoutOccupant = new AlertDialog.Builder(this);
00309
00310         ajoutOccupant.setMessage("Veuillez saisir vos informations : ");
00311         ajoutOccupant.setView(R.layout.ajout_occupant);
00312         ajoutOccupant.setPositiveButton("Ajouter", new DialogInterface.OnClickListener()
00313         {
00314             @Override
00315             public void onClick(DialogInterface dialog, int which)
00316             {
00317                 nomOccupant = (EditText) ((AlertDialog) dialog).findViewById(R.id.nomOccupant);
00318                 prenomOccupant = (EditText) ((AlertDialog) dialog).findViewById(R.id.
      prenomOccupant);
00319                 fonctionOccupant = (EditText) ((AlertDialog) dialog).findViewById(R.id.
      fonctionOccupant);
00320                 Log.v(TAG, "Occupant ajouté :" + "Nom = " + nomOccupant.getText().toString()
      + " "
00321                         + "Prenom = " + prenomOccupant.getText().toString() + " "
00322                         + "Fonction = " + fonctionOccupant.getText().toString());
00323
00324                 adapterOccupant.add(nomOccupant.getText().toString() + " " +
      prenomOccupant.getText().toString() + " " + fonctionOccupant.getText().
      toString());
00325
00326                 occupants.inserer(nomOccupant.getText().toString(),
      prenomOccupant.getText().toString(), fonctionOccupant.getText().toString());
00327             }
00328         });
00329         ajoutOccupant.setNegativeButton("Annuler", new DialogInterface.OnClickListener()
00330         {
00331             @Override
00332             public void onClick(DialogInterface dialog, int which)
00333             {
00334                 Log.v(TAG, "Ajout annulé");
00335             }
00336         });
00337     }
```

7.5.2.4 initialiserListeDeroulante()

`com.example.groom.IHMConnexion.initialiserListeDeroulante ( )  [private]`

Méthode qui initialise la liste déroulante d'appareils Bluetooth appairés.

Définition à la ligne 233 du fichier IHMConnexion.java.

Références com.example.groom.IHMConnexion.listeNomsAppareilsBluetooth, et com.example.groom.IHMConnexion.listeOccupants.

Référencé par com.example.groom.IHMConnexion.listerOccupant().

```
00234     {
00235         adapterGroom = new ArrayAdapter<String>(this, R.layout.
      support_simple_spinner_dropdown_item, listeNomsAppareilsBluetooth);
00236         adapterGroom.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
00237         listeGroom.setAdapter(adapterGroom);
00238 
00239         adapterOccupant = new ArrayAdapter<String>(this, R.layout.
      support_simple_spinner_dropdown_item, listeOccupants);
00240         adapterOccupant.setDropDownViewResource(android.R.layout.
      simple_spinner_dropdown_item);
00241         listeOccupant.setAdapter(adapterOccupant);
00242     }
```

7.5.2.5 initialiserRessourcesIHM()

`com.example.groom.IHMConnexion.initialiserRessourcesIHM ( )  [private]`

Méthode qui initialise les ressources du layout de l'activité IHMConnexion.

Définition à la ligne 206 du fichier IHMConnexion.java.

Références com.example.groom.IHMConnexion.initialiserAjoutOccupant(), et com.example.groom.IHMConnexion.initialiserRetrait↩Occupant().

Référencé par com.example.groom.IHMConnexion.onCreate().

```
00207     {
00208         Log.d(TAG, "initialiserRessourcesIHM()");
00209 
00210         boutonConnexion = findViewById(R.id.bouttonConnexion);
00211         boutonAjoutOccupant = findViewById(R.id.bouttonAjoutOccupant);
00212         boutonRetraitOccupant = findViewById(R.id.bouttonRetraitOccupant);
00213 
00214         boutonConnexion.setOnClickListener(this);
00215         boutonAjoutOccupant.setOnClickListener(this);
00216         boutonRetraitOccupant.setOnClickListener(this);
00217 
00218         listeGroom = findViewById(R.id.listeGroom);
00219         listeOccupant = findViewById(R.id.listeOccupant);
00220 
00221         listeNomsAppareilsBluetooth = new ArrayList<String>();
00222         listeAppareilsBluetooth = new ArrayList<BluetoothDevice>();
00223         listeOccupants = new ArrayList<String>();
00224 
00225         initialiserAjoutOccupant();
00226         initialiserRetraitOccupant();
00227     }
```

7.5.2.6 initialiserRetraitOccupant()

```
void com.example.groom.IHMConnexion.initialiserRetraitOccupant ( )  [private]
```

Définition à la ligne 343 du fichier IHMConnexion.java.

Références com.example.groom.Occupant.getFonction(), com.example.groom.Occupant.getId(), com.example.groom.Occupant.getNom(), com.example.groom.Groom.getOccupant(), com.example.groom.Occupant.getPrenom(), com.example.groom.IHMConnexion.onClick(), com.example.groom.IHMConnexion.retraitOccupant, et com.example.groom.Occupants.supprimer().

Référencé par com.example.groom.IHMConnexion.initialiserRessourcesIHM().

```
00344     {
00345         retraitOccupant = new AlertDialog.Builder(this);
00346         retraitOccupant.setMessage("Vous êtes sur le point de supprimer l'occupant choisit.
      \r\n\r\n" + "Êtes-vous sûr de vouloir le supprimer ?");
00347         retraitOccupant.setPositiveButton("Supprimer", new DialogInterface.OnClickListener()
00348         {
00349             @Override
00350             public void onClick(DialogInterface dialog, int which)
00351             {
00352                 Log.v(TAG, "Supression Occupant");
00353                 occupants.supprimer(groom.getOccupant().
      getId());
00354                 adapterOccupant.remove(groom.getOccupant().
      getNom() + " " + groom.getOccupant().getPrenom() + " " +
      groom.getOccupant().getFonction());
00355                 listeOccupant.setAdapter(adapterOccupant);
00356             }
00357         });
00358         retraitOccupant.setNegativeButton("Annuler", new DialogInterface.OnClickListener()
00359         {
00360             @Override
00361             public void onClick(DialogInterface dialog, int which)
00362             {
00363                 Log.v(TAG, "Suppression annulée");
00364             }
00365         });
00366     }
```

7.5.2.7 lancerRecherchePeripherique()

```
com.example.groom.IHMConnexion.lancerRecherchePeripherique ( )
```

Méthode qui lance la recherche de périphérique.

Définition à la ligne 172 du fichier IHMConnexion.java.

Références com.example.groom.Preferences.getPreference(), et com.example.groom.IHMConnexion.listerOccupant().

Référencé par com.example.groom.IHMConnexion.activerBluetooth().

```
00173     {
00174         devices = bluetoothAdapter.getBondedDevices();
00175
00176         //if(!(preferences.getPreference().getAppareilGroom().equals("")))
00177         //{
00178             //listeNomsAppareilsBluetooth.add(preferences.getPreference().getAppareilGroom());
00179         //}
00180
00181         Log.v(TAG, preferences.getPreference().toString());
00182
00183         for (BluetoothDevice bluetoothDevice : devices)
00184         {
00185             listeAppareilsBluetooth.add(bluetoothDevice);
00186             listeNomsAppareilsBluetooth.add(bluetoothDevice.getName());
00187         }
00188
00189         listerOccupant();
00190     }
```

7.5.2.8 listerOccupant()

```
void com.example.groom.IHMConnexion.listerOccupant ( )  [private]
```

Définition à la ligne 192 du fichier IHMConnexion.java.

Références com.example.groom.Occupants.getListe(), et com.example.groom.IHMConnexion.initialiserListeDeroulante().

Référencé par com.example.groom.IHMConnexion.lancerRecherchePeripherique().

```
00193     {
00194         List<Occupant> liste = occupants.getListe();
00195         for(int i = 0; i<liste.size(); i++)
00196         {
00197             listeOccupants.add(liste.get(i).getNom() + " " + liste.get(i).getPrenom() + " " +
       liste.get(i).getFonction());
00198         }
00199         initialiserListeDeroulante();
00200     }
```

7.5.2.9 onActivityResult()

```
com.example.groom.IHMConnexion.onActivityResult (
             int requestCode,
             int resultCode,
             Intent intent )  [protected]
```

Méthode appelée à la fin de l'activité lancée et récupère l'objet groom envoyé

Paramètres

requestCode	le code de requête
resultCode	le code de résultat
intent	l'objet Intent utilisé pour envoyer l'objet Groom

Définition à la ligne 252 du fichier IHMConnexion.java.

Références com.example.groom.Groom.getDisponibilite(), et com.example.groom.Groom.getModeSonnette().

```
00253     {
00254         if (requestCode == ACTION_GROOM && resultCode == RESULT_OK)
00255         {
00256             groom = (Groom) intent.getSerializableExtra("Groom");
00257             Log.v(TAG, "Disponibilité = " + groom.getDisponibilite());
00258             Log.v(TAG, "Sonnette = " + groom.getModeSonnette());
00259         }
00260     }
```

7.5.2.10 onClick()

```
void com.example.groom.IHMConnexion.onClick (
             View element )
```

Définition à la ligne 263 du fichier IHMConnexion.java.

Références com.example.groom.IHMConnexion.afficherToast(), com.example.groom.IHMConnexion.ajoutOccupant, et com.example.groom.IHMConnexion.retraitOccupant.

Référencé par com.example.groom.IHMConnexion.initialiserAjoutOccupant(), et com.example.groom.IHMConnexion.initialiserRetraitOccupant().

```
00264     {
00265         switch (element.getId())
00266         {
00267             case R.id.bouttonAjoutOccupant:
00268                 afficherToast("Ajout Occupant");
00269                 ajoutOccupant.show();
00270                 break;
00271
00272             case R.id.bouttonRetraitOccupant:
00273                 afficherToast("Retrait Occupant");
00274                 retraitOccupant.show();
00275                 break;
00276
00277             case R.id.bouttonConnexion:
00278                 afficherToast("Connexion");
00279
00280                 Intent intent = new Intent(IHMConnexion.this, IHMGroom.class);
00281                 intent.putExtra("Groom", (Serializable) groom);
00282                 startActivityForResult(intent, ACTION_GROOM);
00283                 /*if(preferences.getPreference() == null)
00284                     preferences.inserer(groom.getNomDevice(), listeOccupant.getId());
00285                 else
00286                     preferences.modifier(0, groom.getNomDevice(), listeOccupant.getId());*/
00287                 break;
00288         }
00289     }
```

7.5.2.11 onCreate()

```
com.example.groom.IHMConnexion.onCreate (
            Bundle savedInstanceState )  [protected]
```

Méthode appelée à la création de l'activité IHMConnexion.

Paramètres

savedInstanceState	

Définition à la ligne 77 du fichier IHMConnexion.java.

Références com.example.groom.IHMConnexion.activerBluetooth(), com.example.groom.IHMConnexion.initialiserRessourcesIHM(), com.example.groom.IHMConnexion.selectionnerGroom(), et com.example.groom.IHMConnexion.selectionnerOccupant().

```
00078     {
00079         super.onCreate(savedInstanceState);
00080         setContentView(R.layout.activity_communication);
00081
00082         groom = new Groom("Eyraud", "Grégory", "étudiant", "Libre", true, false);
00083         occupants = new Occupants(this);
00084         preferences = new Preferences(this);
00085
00086         initialiserRessourcesIHM();
00087         activerBluetooth();
00088
00089         selectionnerGroom();
00090         selectionnerOccupant();
00091     }
```

7.5.2.12 selectionnerGroom()

```
void com.example.groom.IHMConnexion.selectionnerGroom ( )  [private]
```

Méthode permet de sélectionner un portier Groom Bluetooth dans la liste.

Définition à la ligne 96 du fichier IHMConnexion.java.

Références com.example.groom.Groom.setNomDevice().

Référencé par com.example.groom.IHMConnexion.onCreate().

```
00096                                       {
00097         listeGroom.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()
00098         {
00099             @Override
00100             public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
00101             {
00102                 Log.v(TAG, "onItemSelected()");
00103                 for (int i = 0; i < listeAppareilsBluetooth.size(); i++)
00104                 {
00105                     if (listeAppareilsBluetooth.get(i).getName().equals(
      listeNomsAppareilsBluetooth.get(position)))
00106                     {
00107                         Log.v(TAG, "onItemSelected() " +
      listeAppareilsBluetooth.get(i).getName());
00108                         groom.setNomDevice(
      listeAppareilsBluetooth.get(i).getName());
00109                         break;
00110                     }
00111                 }
00112             }
00113
00114             @Override
00115             public void onNothingSelected(AdapterView<?> parent)
00116             {
00117
00118             }
00119         });
00120     }
```

7.5.2.13 selectionnerOccupant()

```
void com.example.groom.IHMConnexion.selectionnerOccupant ( )  [private]
```

Méthode permet de sélectionner un occupant dans la liste.

Définition à la ligne 125 du fichier IHMConnexion.java.

Références com.example.groom.Occupants.getOccupant(), et com.example.groom.Groom.setOccupant().

Référencé par com.example.groom.IHMConnexion.onCreate().

```
00125                                       {
00126         listeOccupant.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
00127             @Override
00128             public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
00129             {
00130                 Log.v(TAG, "onItemSelected()" + "Occupant choisi : " +
      listeOccupants.get(position));
00131                 String identite[] = listeOccupants.get(position).split(" ");
00132                 groom.setOccupant(occupants.getOccupant(identite[0]));
00133             }
00134
00135             @Override
00136             public void onNothingSelected(AdapterView<?> parent) {
00137
00138             }
00139         });
00140     }
```

7.5.3 Documentation des données membres

7.5.3.1 ACTION_GROOM

```
final int com.example.groom.IHMConnexion.ACTION_GROOM = 1  [static], [private]
```

Code de requête lors du début et de la fin de l'activité IHMGroom.

Définition à la ligne 40 du fichier IHMConnexion.java.

7.5.3.2 adapterGroom

```
ArrayAdapter<String> com.example.groom.IHMConnexion.adapterGroom  [private]
```

Adaptateur pour mettre la liste de noms des appareils dans la liste déroulante listeGroom.

Définition à la ligne 64 du fichier IHMConnexion.java.

7.5.3.3 adapterOccupant

```
ArrayAdapter<String> com.example.groom.IHMConnexion.adapterOccupant  [private]
```

Adaptateur pour mettre la liste des occupants dnas la liste déroulante listeOccupant.

Définition à la ligne 65 du fichier IHMConnexion.java.

7.5.3.4 ajoutOccupant

```
AlertDialog.Builder com.example.groom.IHMConnexion.ajoutOccupant  [private]
```

Le builder qui permet de créer une fenêtre de dialogue d'ajout d'occupant.

Définition à la ligne 50 du fichier IHMConnexion.java.

Référencé par com.example.groom.IHMConnexion.initialiserAjoutOccupant(), et com.example.groom.IHMConnexion.onClick().

7.5.3.5 bluetoothAdapter

```
BluetoothAdapter com.example.groom.IHMConnexion.bluetoothAdapter  [private]
```

L'objet BluetoothAdapter.

Définition à la ligne 60 du fichier IHMConnexion.java.

7.5.3.6 boutonAjoutOccupant

```
Button com.example.groom.IHMConnexion.boutonAjoutOccupant  [private]
```

Le bouton pour ajouter un occupant.

Définition à la ligne 46 du fichier IHMConnexion.java.

7.5.3.7 boutonConnexion

```
Button com.example.groom.IHMConnexion.boutonConnexion  [private]
```

Le bouton pour se connecter à l'appareil Groom.

Ressources IHM

Définition à la ligne 45 du fichier IHMConnexion.java.

7.5.3.8 boutonRetraitOccupant

```
Button com.example.groom.IHMConnexion.boutonRetraitOccupant  [private]
```

Le bouton pour retirer un occupant.

Définition à la ligne 47 du fichier IHMConnexion.java.

7.5.3.9 devices

```
Set<BluetoothDevice> com.example.groom.IHMConnexion.devices  [private]
```

Conteneur qui liste les appareils bluetooth disponibles sans doublons.

Définition à la ligne 61 du fichier IHMConnexion.java.

7.5.3.10 fonctionOccupant

```
EditText com.example.groom.IHMConnexion.fonctionOccupant  [private]
```

Le champ de texte pour écrire la fonction de l'occupnt.

Définition à la ligne 54 du fichier IHMConnexion.java.

7.5.3.11 groom

```
Groom com.example.groom.IHMConnexion.groom = null  [private]
```

L'objet groom.

Attributs

Définition à la ligne 59 du fichier IHMConnexion.java.

7.5.3.12 listeAppareilsBluetooth

```
List<BluetoothDevice> com.example.groom.IHMConnexion.listeAppareilsBluetooth  [private]
```

Conteneur qui liste les appareils bluetooth disponibles.

Définition à la ligne 66 du fichier IHMConnexion.java.

7.5.3.13 listeGroom

```
Spinner com.example.groom.IHMConnexion.listeGroom  [private]
```

Liste déroulante des appareils bluetooth.

Définition à la ligne 48 du fichier IHMConnexion.java.

7.5.3.14 listeNomsAppareilsBluetooth

```
List<String> com.example.groom.IHMConnexion.listeNomsAppareilsBluetooth  [private]
```

Conteneur qui liste les noms des appareils bluetooth disponibles.

Définition à la ligne 62 du fichier IHMConnexion.java.

Référencé par com.example.groom.IHMConnexion.initialiserListeDeroulante().

7.5.3.15 listeOccupant

```
Spinner com.example.groom.IHMConnexion.listeOccupant  [private]
```

Liste déroulante des occupants.

Définition à la ligne 49 du fichier IHMConnexion.java.

7.5.3.16 listeOccupants

```
List<String> com.example.groom.IHMConnexion.listeOccupants  [private]
```

Conteneur qui liste les occupants.

Définition à la ligne 63 du fichier IHMConnexion.java.

Référencé par com.example.groom.IHMConnexion.initialiserListeDeroulante().

7.5.3.17 nomOccupant

```
EditText com.example.groom.IHMConnexion.nomOccupant  [private]
```

Le champ de texte pour écrire le nom de l'occupant.

Définition à la ligne 52 du fichier IHMConnexion.java.

7.5.3.18 occupants

```
Occupants com.example.groom.IHMConnexion.occupants = null  [private]
```

Définition à la ligne 68 du fichier IHMConnexion.java.

7.5.3.19 preferences

```
Preferences com.example.groom.IHMConnexion.preferences = null  [private]
```

Définition à la ligne 69 du fichier IHMConnexion.java.

7.5.3.20 prenomOccupant

```
EditText com.example.groom.IHMConnexion.prenomOccupant  [private]
```

Le champ de texte pour écrire le prenom de l'occupant.

Définition à la ligne 53 du fichier IHMConnexion.java.

7.5.3.21 retraitOccupant

```
AlertDialog.Builder com.example.groom.IHMConnexion.retraitOccupant  [private]
```

Le builder qui permet de créer une fenêtre de dialogue de retrait d'occupant.

Définition à la ligne 51 du fichier IHMConnexion.java.

Référencé par com.example.groom.IHMConnexion.initialiserRetraitOccupant(), et com.example.groom.IHMConnexion.onClick().

7.5.3.22 TAG

```
final String com.example.groom.IHMConnexion.TAG = "IHMConnexion"  [static], [private]
```

TAG pour les logs.

Constantes

Définition à la ligne 39 du fichier IHMConnexion.java.

7.5.3.23 toast

```
Toast com.example.groom.IHMConnexion.toast  [private]
```

Le toast qui permet d'afficher des informations à l'utilisateur.

Définition à la ligne 67 du fichier IHMConnexion.java.

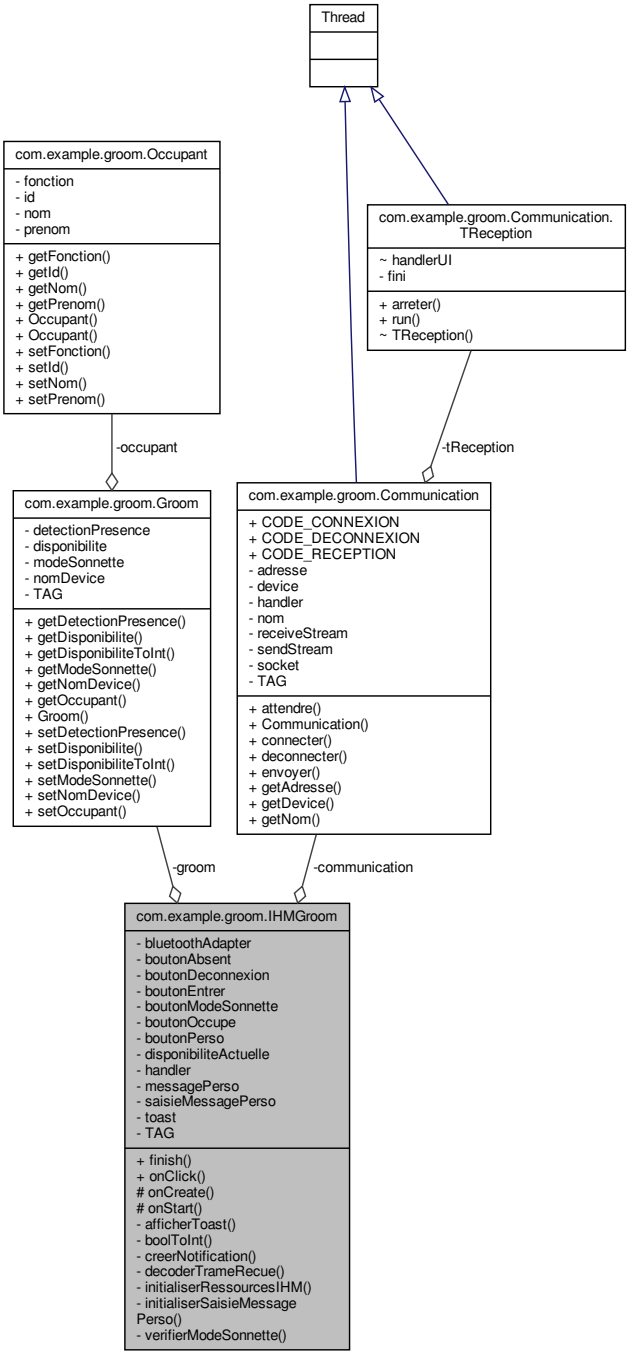
La documentation de cette classe a été générée à partir du fichier suivant :

— IHMConnexion.java

7.6 Référence de la classe com.example.groom.IHMGroom

Déclaration de la classe IHMGroom.

Graphe de collaboration de com.example.groom.IHMGroom :



Fonctions membres publiques

- void finish ()
  Méthode appelée à la fin de l'activité
- void onClick (View element)
  Méthode appelé au click d'un bouton et appelle une méthode celon le bouton qui a été cliqué

Fonctions membres protégées

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

Fonctions membres privées

- void afficherToast (String message)
  Méthode appelée pour donner des informations supplémentaires à l'utilisateur.
- int boolToInt (boolean b)
  Méthode qui convertit un bool en un int.
- void creerNotification (String texte, int id)
  Méthode qui permet de créer une notification pour avertir l'utilisateur si quelqu'un à sonné et/ou si quelqu'un est présent devant la porte.
- void decoderTrameRecue (String trame[ ])
  Méthode qui permet de décoder la trame reçue.
- void initialiserRessourcesIHM ()
  Méthode appelée pour initialiser les différentes ressources nécessaire à l'affichage de l'IHM.
- void initialiserSaisieMessagePerso ()
  Méthode appelée pour initialiser la boite de dialogue personnalisé pour la saisie d'un message personnalisé
- void verifierModeSonnette ()
  Méthode appelée pour vérifier le mode de sonnette et changer le texte du boutton.

Attributs privés

- BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter()
  L'objet Bluetooth.
- Button boutonAbsent
  Le bouton pour définir sa disponibilité en Absent.
- Button boutonDeconnexion
  Le bouton pour se déconnecter.
- Button boutonEntrer
  Le bouton pour définir sa disponibilité en Libre.
- Button boutonModeSonnette
  Le bouton pour activer/désactiver la sonnette.
- Button boutonOccupe
  Le bouton pour définir sa disponibilité en Occupé
- Button boutonPerso
  Le bouton pour envoyer un message personnalisé
- Communication communication = null
  l'objet communication pour communiquer avec le portier groom
- TextView disponibiliteActuelle
  Le texte qui affiche la dernière disponiblité définie.
- Groom groom = null
  L'objet groom connecté
- final Handler handler
  objet Handler utiliser pour la reception du code de retour de la communication
- EditText messagePerso
  Le champ de texte pour écrire son message personnalisé
- AlertDialog.Builder saisieMessagePerso
  Le builder qui permet de créer une fenêtre de dialogue de saisie personnalisé
- Toast toast
  Le toast qui permet d'afficher des informations à l'utilisateur.

Attributs privés statiques

— static final String TAG = "IHMGroom"
TAG pour les logs.

7.6.1 Description détaillée

Déclaration de la classe IHMGroom.

Définition à la ligne 35 du fichier IHMGroom.java.

7.6.2 Documentation des fonctions membres

7.6.2.1 afficherToast()

```
com.example.groom.IHMGroom.afficherToast (
            String message )  [private]
```

Méthode appelée pour donner des informations supplémentaires à l'utilisateur.

Paramètres

message	le message à afficher

Définition à la ligne 239 du fichier IHMGroom.java.

Référencé par com.example.groom.IHMGroom.onClick().

```
00240     {
00241         toast = Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT);
00242         toast.show();
00243     }
```

7.6.2.2 boolToInt()

```
com.example.groom.IHMGroom.boolToInt (
            boolean b )  [private]
```

Méthode qui convertit un bool en un int.

Paramètres

b	le booléen à convertir

Renvoie

int 1 pour true et 0 pour false

Définition à la ligne 358 du fichier IHMGroom.java.

Référencé par com.example.groom.IHMGroom.onClick().

```
00359     {
00360         return b ? 1 : 0;
00361     }
```

7.6.2.3 creerNotification()

```
com.example.groom.IHMGroom.creerNotification (
            String texte,
            int id )  [private]
```

Méthode qui permet de créer une notification pour avertir l'utilisateur si quelqu'un à sonné et/ou si quelqu'un est présent devant la porte.

Paramètres

texte	le texte de la notification
id	unique à la notification

Définition à la ligne 338 du fichier IHMGroom.java.

Référencé par com.example.groom.IHMGroom.decoderTrameRecue().

```
00339     {
00340         NotificationCompat.Builder builder = new NotificationCompat.Builder(this, "channelId")
00341                 .setSmallIcon(R.drawable.ic_launcher_background)
00342                 .setContentTitle("GrOOm")
00343                 .setContentText(texte)
00344                 .setPriority(NotificationCompat.PRIORITY_DEFAULT);
00345         NotificationManagerCompat notificationSonnette = NotificationManagerCompat.from(this);
00346         notificationSonnette.notify(id, builder.build());
00347         Log.v(TAG, "Notification Groom : " + texte);
00348     }
```

7.6.2.4 decoderTrameRecue()

```
com.example.groom.IHMGroom.decoderTrameRecue (
            String trame[] )  [private]
```

Méthode qui permet de décoder la trame reçue.

Paramètres

trame	la trame à décoder

Définition à la ligne 313 du fichier IHMGroom.java.

Références com.example.groom.IHMGroom.creerNotification(), com.example.groom.Groom.getDisponibilite(), et com.example.groom.Groom.setDisponibiliteToInt().

```
00314     {
00315         if (trame[0].equals("$GROOM"))
00316         {
00317             groom.setDisponibiliteToInt(Integer.parseInt(trame[1]));
00318             disponibiliteActuelle.setText(groom.
      getDisponibilite());
00319             if(trame[2].equals("1"))
00320             {
00321                 creerNotification("Quelqu'un vient de sonner à votre porte", 1);
00322             }
00323             if(trame[3].equals("1"))
00324             {
00325                 creerNotification("Une personne attend devant votre bureau", 2);
00326             }
00327         }
00328     }
```

7.6.2.5 finish()

`com.example.groom.IHMGroom.finish ( )`

Méthode appelée à la fin de l'activité

Définition à la ligne 251 du fichier IHMGroom.java.

Références com.example.groom.Communication.deconnecter().

Référencé par com.example.groom.IHMGroom.onClick().

```
00252     {
00253         if(communication != null)
00254             communication.deconnecter();
00255         Intent data = new Intent();
00256         data.putExtra("Groom", groom);
00257         setResult(RESULT_OK, data);
00258         super.finish();
00259     }
```

7.6.2.6 initialiserRessourcesIHM()

`com.example.groom.IHMGroom.initialiserRessourcesIHM ( )  [private]`

Méthode appelée pour initialiser les différentes ressources nécessaire à l'affichage de l'IHM.

Définition à la ligne 120 du fichier IHMGroom.java.

Références com.example.groom.IHMGroom.verifierModeSonnette().

Référencé par com.example.groom.IHMGroom.onCreate().

```
00121     {
00122         // Les boutons
00123         boutonPerso = findViewById(R.id.bouttonMessagePersonnalise);
00124         boutonEntrer = findViewById(R.id.boutonLibre);
00125         boutonAbsent = findViewById(R.id.boutonAbsent);
00126         boutonOccupe = findViewById(R.id.boutonOccupe);
00127         boutonModeSonnette = findViewById(R.id.boutonSonnette);
00128         boutonDeconnexion = findViewById(R.id.boutonDeconnexion);
00129
00130         boutonPerso.setOnClickListener(this);
00131         boutonEntrer.setOnClickListener(this);
00132         boutonAbsent.setOnClickListener(this);
00133         boutonOccupe.setOnClickListener(this);
00134         boutonModeSonnette.setOnClickListener(this);
00135         boutonDeconnexion.setOnClickListener(this);
00136
00137         disponibiliteActuelle = (TextView) findViewById(R.id.disponibiliteActuelle);
00138
00139         verifierModeSonnette();
00140     }
```

7.6.2.7 initialiserSaisieMessagePerso()

```
com.example.groom.IHMGroom.initialiserSaisieMessagePerso ( )  [private]
```

Méthode appelée pour initialiser la boite de dialogue personnalisé pour la saisie d'un message personnalisé

Définition à la ligne 147 du fichier IHMGroom.java.

Références com.example.groom.Communication.envoyer(), com.example.groom.IHMGroom.onClick(), et com.example.groom.IHM↩ Groom.saisieMessagePerso.

Référencé par com.example.groom.IHMGroom.onCreate().

```
00148     {
00149         saisieMessagePerso = new AlertDialog.Builder(this);
00150 
00151         saisieMessagePerso.setMessage("Veuillez saisir votre message :");
00152         saisieMessagePerso.setView(R.layout.saisie_message_perso);
00153         saisieMessagePerso.setPositiveButton("Ok", new DialogInterface.OnClickListener()
    {
00154             @Override
00155             public void onClick(DialogInterface dialog, int which) {
00156                 messagePerso = (EditText) ((AlertDialog) dialog).findViewById(R.id.
    saisieMessagePerso);
00157                 Log.v(TAG, "Saisie : " + messagePerso.getText().toString());
00158                 communication.envoyer("$MSGPERSO;" +
    messagePerso.getText() + "\r\n");
00159             }
00160         });
00161         saisieMessagePerso.setNegativeButton("Annuler", new DialogInterface.
    OnClickListener() {
00162             @Override
00163             public void onClick(DialogInterface dialog, int which) {
00164                 Log.v(TAG, "Saisie annulée");
00165             }
00166         });
00167     }
```

7.6.2.8 onClick()

```
com.example.groom.IHMGroom.onClick (
             View element )
```

Méthode appelé au click d'un bouton et appelle une méthode celon le bouton qui a été cliqué

Paramètres

element	l'élément cliqué

Définition à la ligne 176 du fichier IHMGroom.java.

Références com.example.groom.IHMGroom.afficherToast(), com.example.groom.IHMGroom.boolToInt(), com.example.groom.↩ Communication.deconnecter(), com.example.groom.Communication.envoyer(), com.example.groom.IHMGroom.finish(), com.↩ example.groom.Groom.getDetectionPresence(), com.example.groom.Groom.getDisponibilite(), com.example.groom.Groom.get↩ DisponibiliteToInt(), com.example.groom.Groom.getModeSonnette(), com.example.groom.IHMGroom.saisieMessagePerso, com.↩ example.groom.Groom.setDisponibilite(), et com.example.groom.Groom.setModeSonnette().

Référencé par com.example.groom.IHMGroom.initialiserSaisieMessagePerso().

```
00177     {
00178         switch (element.getId()) {
00179             case R.id.bouttonMessagePersonnalise:
```

```
00180                 saisieMessagePerso.show();
00181                 break;
00182
00183             case R.id.boutonLibre:
00184                 communication.envoyer("$GROOM;0;" +
      boolToInt(groom.getModeSonnette()) + ";" +
      boolToInt(groom.getDetectionPresence()) + "\r\n");
00185                 if (groom != null)
00186                     groom.setDisponibilite("Libre");
00187                 break;
00188
00189             case R.id.boutonAbsent:
00190                 communication.envoyer("$GROOM;1;" +
      boolToInt(groom.getModeSonnette()) + ";" +
      boolToInt(groom.getDetectionPresence()) + "\r\n");
00191                 if (groom != null)
00192                     groom.setDisponibilite("Absent");
00193                 break;
00194
00195             case R.id.boutonOccupe:
00196                 communication.envoyer("$GROOM;2;" +
      boolToInt(groom.getModeSonnette()) + ";" +
      boolToInt(groom.getDetectionPresence()) + "\r\n");
00197                 if (groom != null)
00198                     groom.setDisponibilite("Occupé");
00199                 break;
00200
00201             case R.id.boutonSonnette:
00202                 if (boutonModeSonnette.getText().equals(getString(R.string.
      desactiverSonnette)))
00203                 {
00204                     if (groom != null)
00205                     {
00206                         groom.setModeSonnette(false);
00207                         communication.envoyer("$GROOM;" +
      groom.getDisponibiliteToInt() + ";" + boolToInt(
      groom.getModeSonnette()) + ";" + boolToInt(groom.
      getDetectionPresence()) + "\r\n");
00208                         boutonModeSonnette.setText(R.string.activerSonnette);
00209                     }
00210                 }
00211                 else
00212                 {
00213                     if (groom != null)
00214                     {
00215                         groom.setModeSonnette(true);
00216                         communication.envoyer("$GROOM;" +
      groom.getDisponibiliteToInt() + ";" + boolToInt(
      groom.getModeSonnette()) + ";" + boolToInt(groom.
      getDetectionPresence()) + "\r\n");
00217                         boutonModeSonnette.setText(R.string.desactiverSonnette);
00218                     }
00219                 }
00220                 break;
00221
00222             case R.id.boutonDeconnexion:
00223                 afficherToast("Déconnexion");
00224                 communication.deconnecter();
00225                 finish();
00226                 break;
00227         }
00228
00229         if (groom != null)
00230             disponibiliteActuelle.setText(groom.
      getDisponibilite());
00231     }
```

7.6.2.9 onCreate()

```
com.example.groom.IHMGroom.onCreate (
            Bundle savedInstanceState )  [protected]
```

Méthode appelée à la création de l'activité IHMGroom.

Paramètres

savedInstanceState	

Définition à la ligne 69 du fichier IHMGroom.java.

Références com.example.groom.Communication.connecter(), com.example.groom.Groom.getDisponibilite(), com.example.groom.↩ Occupant.getNom(), com.example.groom.Groom.getNomDevice(), com.example.groom.Groom.getOccupant(), com.example.↩ groom.Occupant.getPrenom(), com.example.groom.IHMGroom.handler, com.example.groom.IHMGroom.initialiserRessourcesIHM(), et com.example.groom.IHMGroom.initialiserSaisieMessagePerso().

```
00070     {
00071         super.onCreate(savedInstanceState);
00072         setContentView(R.layout.activity_groom);
00073 
00074         Intent intent = getIntent();
00075         groom = (Groom) intent.getSerializableExtra("Groom");
00076         if (groom != null)
00077         {
00078             Log.d(TAG, "Groom : " + groom.getOccupant().
      getNom() + " " + groom.getOccupant().getPrenom() + " " +
      groom.getDisponibilite() + " " + groom.getNomDevice());
00079         }
00080 
00081         Set<BluetoothDevice> devices = bluetoothAdapter.getBondedDevices();
00082         //Log.d(TAG, "Nb BluetoothDevice " + devices.size());
00083         for (BluetoothDevice bluetoothDevice : devices)
00084         {
00085             //Log.d(TAG, "BluetoothDevice " + bluetoothDevice.getName() + " [" +
       bluetoothDevice.getAddress() + "]");
00086             if(bluetoothDevice.getName().equals(groom.getNomDevice()))
00087             {
00088                 Log.d(TAG, "BluetoothDevice Groom trouvé : " + bluetoothDevice.getName() + " [" +
      bluetoothDevice.getAddress() + "]");
00089                 communication = new Communication(bluetoothDevice,
      handler);
00090             }
00091         }
00092 
00093         if(communication != null)
00094         {
00095             communication.connecter();
00096         }
00097 
00098         initialiserRessourcesIHM();
00099         initialiserSaisieMessagePerso();
00100     }
```

7.6.2.10 onStart()

```
com.example.groom.IHMGroom.onStart ( )  [protected]
```

Méthode appelée au lancement de l'activité IHMGroom.

Définition à la ligne 108 du fichier IHMGroom.java.

Références com.example.groom.Groom.getDisponibilite().

```
00109     {
00110         super.onStart();
00111         if (groom != null)
00112             disponibiliteActuelle.setText(groom.
      getDisponibilite());
00113     }
```

7.6.2.11 verifierModeSonnette()

```
com.example.groom.IHMGroom.verifierModeSonnette ( )  [private]
```

Méthode appelée pour vérifier le mode de sonnette et changer le texte du boutton.

Définition à la ligne 266 du fichier IHMGroom.java.

Références com.example.groom.Groom.getModeSonnette().

Référencé par com.example.groom.IHMGroom.initialiserRessourcesIHM().

```
00267     {
00268         if(groom.getModeSonnette())
00269         {
00270             boutonModeSonnette.setText(R.string.desactiverSonnette);
00271         }
00272         else
00273         {
00274             boutonModeSonnette.setText(R.string.activerSonnette);
00275         }
00276     }
```

7.6.3 Documentation des données membres

7.6.3.1 bluetoothAdapter

```
BluetoothAdapter com.example.groom.IHMGroom.bluetoothAdapter = BluetoothAdapter.getDefaultAdapter()  [private]
```

L'objet Bluetooth.

Définition à la ligne 59 du fichier IHMGroom.java.

7.6.3.2 boutonAbsent

```
Button com.example.groom.IHMGroom.boutonAbsent  [private]
```

Le bouton pour définir sa disponibilité en Absent.

Définition à la ligne 46 du fichier IHMGroom.java.

7.6.3.3 boutonDeconnexion

```
Button com.example.groom.IHMGroom.boutonDeconnexion  [private]
```

Le bouton pour se déconnecter.

Définition à la ligne 50 du fichier IHMGroom.java.

7.6.3.4 boutonEntrer

```
Button com.example.groom.IHMGroom.boutonEntrer  [private]
```

Le bouton pour définir sa disponibilité en Libre.

Définition à la ligne 45 du fichier IHMGroom.java.

7.6.3.5 boutonModeSonnette

```
Button com.example.groom.IHMGroom.boutonModeSonnette  [private]
```

Le bouton pour activer/désactiver la sonnette.

Définition à la ligne 48 du fichier IHMGroom.java.

7.6.3.6 boutonOccupe

```
Button com.example.groom.IHMGroom.boutonOccupe  [private]
```

Le bouton pour définir sa disponibilité en Occupé

Définition à la ligne 47 du fichier IHMGroom.java.

7.6.3.7 boutonPerso

```
Button com.example.groom.IHMGroom.boutonPerso  [private]
```

Le bouton pour envoyer un message personnalisé

Ressources IHM

Définition à la ligne 44 du fichier IHMGroom.java.

7.6.3.8 communication

```
Communication com.example.groom.IHMGroom.communication = null  [private]
```

l'objet communication pour communiquer avec le portier groom

Définition à la ligne 60 du fichier IHMGroom.java.

7.6.3.9 disponibiliteActuelle

```
TextView com.example.groom.IHMGroom.disponibiliteActuelle  [private]
```

Le texte qui affiche la dernière disponiblité définie.

Définition à la ligne 51 du fichier IHMGroom.java.

7.6.3.10 groom

```
Groom com.example.groom.IHMGroom.groom = null  [private]
```

L'objet groom connecté

Attributs

Définition à la ligne 58 du fichier IHMGroom.java.

7.6.3.11 handler

```
final Handler com.example.groom.IHMGroom.handler  [private]
```

Valeur initiale :

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

            switch(msg.what)
            {
                case Communication.CODE_CONNEXION:
                    Log.v(TAG, "Groom connecté");
                    communication.envoyer("$AFFICHAGE;" +
      groom.getOccupant().getNom() + ";" + groom.
      getOccupant().getPrenom() + ";" + groom.getOccupant().
      getFonction() + "\r\n");
                    break;
                case Communication.CODE_RECEPTION:
                    Log.v(TAG, "Trame reçue " + msg.obj);
                    String trame[] = msg.obj.toString().split(";");
                    decoderTrameRecue(trame);
                    break;
                case Communication.CODE_DECONNEXION:
                    Log.v(TAG, "Groom déconnecté");
                    break;
            }
        }
    }
```

objet Handler utiliser pour la reception du code de retour de la communication

Définition à la ligne 281 du fichier IHMGroom.java.

Référencé par com.example.groom.IHMGroom.onCreate().

7.6.3.12 messagePerso

```
EditText com.example.groom.IHMGroom.messagePerso  [private]
```

Le champ de texte pour écrire son message personnalisé

Définition à la ligne 49 du fichier IHMGroom.java.

7.6.3.13 saisieMessagePerso

```
AlertDialog.Builder com.example.groom.IHMGroom.saisieMessagePerso  [private]
```

Le builder qui permet de créer une fenêtre de dialogue de saisie personnalisé

Définition à la ligne 52 du fichier IHMGroom.java.

Référencé par com.example.groom.IHMGroom.initialiserSaisieMessagePerso(), et com.example.groom.IHMGroom.onClick().

7.6.3.14 TAG

```
final String com.example.groom.IHMGroom.TAG = "IHMGroom"  [static], [private]
```

TAG pour les logs.

Constantes

Définition à la ligne 39 du fichier IHMGroom.java.

7.6.3.15 toast

```
Toast com.example.groom.IHMGroom.toast  [private]
```

Le toast qui permet d'afficher des informations à l'utilisateur.

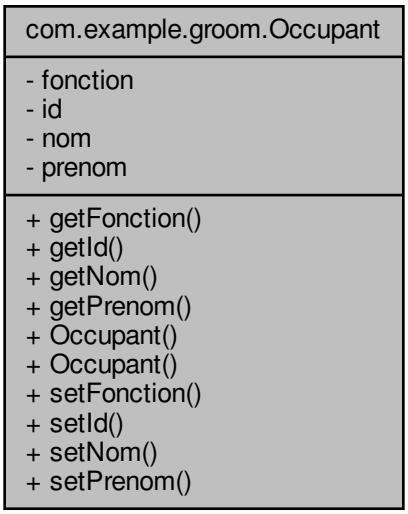
Définition à la ligne 53 du fichier IHMGroom.java.

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

— IHMGroom.java

7.7 Référence de la classe com.example.groom.Occupant

Graphe de collaboration de com.example.groom.Occupant :



Fonctions membres publiques

- String getFonction ()
- int getId ()
- String getNom ()
- String getPrenom ()
- Occupant ()
- Occupant (String nom, String prenom, String fonction)
- void setFonction (String fonction)
- void setId (int id)
- void setNom (String nom)
- void setPrenom (String prenom)

Attributs privés

- String fonction
  La fonction.
- int id
  L'id.
- String nom
  Le nom.
- String prenom
  Le prénom.

7.7.1 Description détaillée

Définition à la ligne 5 du fichier Occupant.java.

7.7.2 Documentation des constructeurs et destructeur

7.7.2.1 Occupant() [1/2]

```
com.example.groom.Occupant.Occupant ( )
```

Définition à la ligne 12 du fichier Occupant.java.

```
00013     {
00014         this.nom = "";
00015         this.prenom = "";
00016         this.fonction ="";
00017     }
```

7.7.2.2 Occupant() [2/2]

```
com.example.groom.Occupant.Occupant (
              String nom,
              String prenom,
              String fonction )
```

Définition à la ligne 19 du fichier Occupant.java.

Références com.example.groom.Occupant.fonction, com.example.groom.Occupant.nom, et com.example.groom.Occupant.prenom.

```
00020     {
00021         this.nom = nom;
00022         this.prenom = prenom;
00023         this.fonction = fonction;
00024     }
```

7.7.3 Documentation des fonctions membres

7.7.3.1 getFonction()

```
String com.example.groom.Occupant.getFonction ( )
```

Définition à la ligne 86 du fichier Occupant.java.

Références com.example.groom.Occupant.fonction.

Référencé par com.example.groom.IHMConnexion.initialiserRetraitOccupant(), et com.example.groom.Occupants.modifier().

```
00087     {
00088         return this.fonction;
00089     }
```

7.7.3.2 getId()

```
int com.example.groom.Occupant.getId ( )
```

Définition à la ligne 26 du fichier Occupant.java.

Références com.example.groom.Occupant.id.

Référencé par com.example.groom.IHMConnexion.initialiserRetraitOccupant().

```
00027     {
00028         return id;
00029     }
```

7.7.3.3 getNom()

```
String com.example.groom.Occupant.getNom ( )
```

Définition à la ligne 42 du fichier Occupant.java.

Références com.example.groom.Occupant.nom.

Référencé par com.example.groom.IHMConnexion.initialiserRetraitOccupant(), com.example.groom.Occupants.modifier(), et com.↩ example.groom.IHMGroom.onCreate().

```
00043     {
00044         return this.nom;
00045     }
```

7.7.3.4 getPrenom()

```
String com.example.groom.Occupant.getPrenom ( )
```

Définition à la ligne 64 du fichier Occupant.java.

Références com.example.groom.Occupant.prenom.

Référencé par com.example.groom.IHMConnexion.initialiserRetraitOccupant(), com.example.groom.Occupants.modifier(), et com.↩ example.groom.IHMGroom.onCreate().

```
00065     {
00066         return this.prenom;
00067     }
```

7.7.3.5 setFonction()

```
void com.example.groom.Occupant.setFonction (
            String fonction )
```

Définition à la ligne 97 du fichier Occupant.java.

Références com.example.groom.Occupant.fonction.

Référencé par com.example.groom.Occupants.cursorToServeur().

```
00098     {
00099         this.fonction = fonction;
00100     }
```

7.7.3.6 setId()

```
void com.example.groom.Occupant.setId (
            int id )
```

Définition à la ligne 31 du fichier Occupant.java.

Références com.example.groom.Occupant.id.

Référencé par com.example.groom.Occupants.cursorToServeur().

```
00032     {
00033         this.id = id;
00034     }
```

7.7.3.7 setNom()

```
void com.example.groom.Occupant.setNom (
            String nom )
```

Définition à la ligne 53 du fichier Occupant.java.

Références com.example.groom.Occupant.nom.

Référencé par com.example.groom.Occupants.cursorToServeur().

```
00054     {
00055         this.nom = nom;
00056     }
```

7.7.3.8 setPrenom()

```
void com.example.groom.Occupant.setPrenom (
            String prenom )
```

Définition à la ligne 75 du fichier Occupant.java.

Références com.example.groom.Occupant.prenom.

Référencé par com.example.groom.Occupants.cursorToServeur().

```
00076     {
00077         this.prenom = prenom;
00078     }
```

7.7.4 Documentation des données membres

7.7.4.1 fonction

```
String com.example.groom.Occupant.fonction  [private]
```

La fonction.

Définition à la ligne 10 du fichier Occupant.java.

Référencé par com.example.groom.Occupant.getFonction(), com.example.groom.Occupant.Occupant(), et com.example.groom.↩ Occupant.setFonction().

7.7.4.2 id

```
int com.example.groom.Occupant.id  [private]
```

L'id.

Définition à la ligne 7 du fichier Occupant.java.

Référencé par com.example.groom.Occupant.getId(), et com.example.groom.Occupant.setId().

7.7.4.3 nom

```
String com.example.groom.Occupant.nom  [private]
```

Le nom.

Définition à la ligne 8 du fichier Occupant.java.

Référencé par com.example.groom.Occupant.getNom(), com.example.groom.Occupant.Occupant(), et com.example.groom.↩ Occupant.setNom().

7.7.4.4 prenom

```
String com.example.groom.Occupant.prenom  [private]
```

Le prénom.

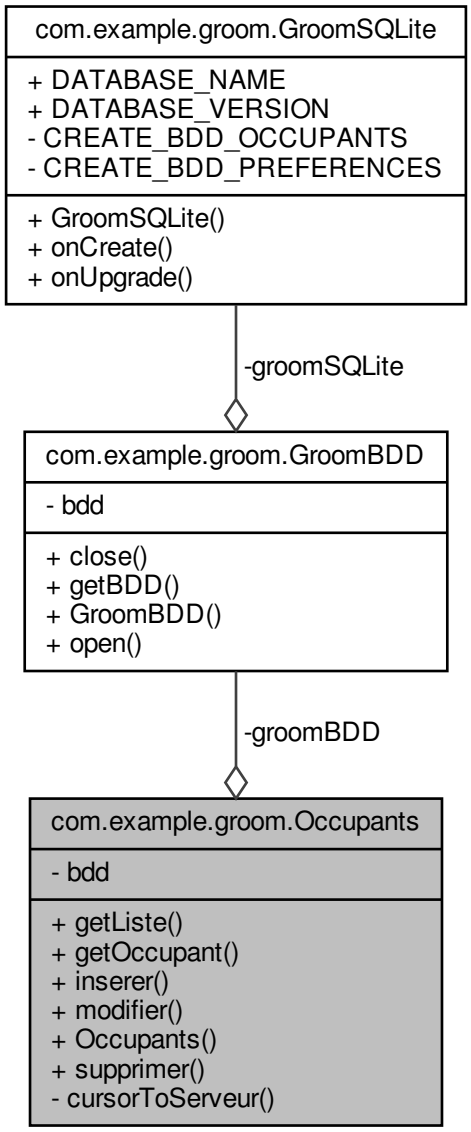
Définition à la ligne 9 du fichier Occupant.java.

Référencé par com.example.groom.Occupant.getPrenom(), com.example.groom.Occupant.Occupant(), et com.example.groom.↩ Occupant.setPrenom().

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

7.8 Référence de la classe com.example.groom.Occupants

Graphe de collaboration de com.example.groom.Occupants :



Fonctions membres publiques

- List< Occupant > getListe ()
- Occupant getOccupant (String nom)
- long inserer (String nom, String prenom, String fonction)
- int modifier (int id, Occupant occupant)
- Occupants (Context context)
- int supprimer (int id)

Fonctions membres privées

- Occupant cursorToServeur (Cursor c, boolean one)

Attributs privés

- SQLiteDatabase bdd
- GroomBDD groomBDD

7.8.1 Description détaillée

Définition à la ligne 11 du fichier Occupants.java.

7.8.2 Documentation des constructeurs et destructeur

7.8.2.1 Occupants()

```
com.example.groom.Occupants.Occupants (
            Context context )
```

Définition à la ligne 16 du fichier Occupants.java.

Références com.example.groom.GroomBDD.getBDD(), et com.example.groom.GroomBDD.open().

```
00017     {
00018         groomBDD = new GroomBDD(context);
00019         groomBDD.open();
00020         bdd = groomBDD.getBDD();
00021     }
```

7.8.3 Documentation des fonctions membres

7.8.3.1 cursorToServeur()

```
Occupant com.example.groom.Occupants.cursorToServeur (
            Cursor c,
            boolean one )  [private]
```

Définition à la ligne 73 du fichier Occupants.java.

Références com.example.groom.Occupant.setFonction(), com.example.groom.Occupant.setId(), com.example.groom.Occupant.setNom(), et com.example.groom.Occupant.setPrenom().

Référencé par com.example.groom.Occupants.getListe(), et com.example.groom.Occupants.getOccupant().

```
00074     {
00075         if (c.getCount() == 0)
00076             return null;
00077 
00078         if(one)
00079             c.moveToFirst();
00080 
00081         Occupant occupant = new Occupant();
00082         occupant.setId(c.getInt(0));
00083         occupant.setNom(c.getString(1));
00084         occupant.setPrenom(c.getString(2));
00085         occupant.setFonction(c.getString(3));
00086 
00087         if (one)
00088             c.close();
00089         return occupant;
00090     }
```

7.8.3.2 getListe()

```
List<Occupant> com.example.groom.Occupants.getListe ( )
```

Définition à la ligne 23 du fichier Occupants.java.

Références com.example.groom.Occupants.cursorToServeur().

Référencé par com.example.groom.IHMConnexion.listerOccupant().

```
00024     {
00025         List<Occupant> occupants = new ArrayList<Occupant>();
00026         Cursor cursor = bdd.query("occupants", new String[] {"idOccupant", "nom", "prenom", "fonction"},
00027                 null, null, null, null, null, null);
00028 
00029         cursor.moveToFirst();
00030         while (!cursor.isAfterLast())
00031         {
00032             Occupant occupant = cursorToServeur(cursor, false);
00033             occupants.add(occupant);
00034             cursor.moveToNext();
00035         }
00036 
00037         cursor.close();
00038         return occupants;
00039     }
```

7.8.3.3 getOccupant()

```
Occupant com.example.groom.Occupants.getOccupant (
            String nom )
```

Définition à la ligne 66 du fichier Occupants.java.

Références com.example.groom.Occupants.cursorToServeur().

Référencé par com.example.groom.IHMConnexion.selectionnerOccupant().

```
00067     {
00068         Cursor c = bdd.rawQuery("SELECT * FROM occupants WHERE nom = ?", new String[] {nom});
00069
00070         return cursorToServeur(c, true);
00071     }
```

7.8.3.4 inserer()

```
long com.example.groom.Occupants.inserer (
            String nom,
            String prenom,
            String fonction )
```

Définition à la ligne 41 du fichier Occupants.java.

Référencé par com.example.groom.IHMConnexion.initialiserAjoutOccupant().

```
00042     {
00043         ContentValues values = new ContentValues();
00044         values.put("nom", nom);
00045         values.put("prenom", prenom);
00046         values.put("fonction", fonction);
00047
00048         return bdd.insert("occupants", null, values);
00049     }
```

7.8.3.5 modifier()

```
int com.example.groom.Occupants.modifier (
            int id,
            Occupant occupant )
```

Définition à la ligne 51 du fichier Occupants.java.

Références com.example.groom.Occupant.getFonction(), com.example.groom.Occupant.getNom(), et com.example.groom.↩ Occupant.getPrenom().

```
00052     {
00053         ContentValues values = new ContentValues();
00054         values.put("nom", occupant.getNom());
00055         values.put("prenom", occupant.getPrenom());
00056         values.put("fonction", occupant.getFonction());
00057
00058         return bdd.update("occupants", values, "idOccupant = " + id, null);
00059     }
```

7.8.3.6 supprimer()

```
int com.example.groom.Occupants.supprimer (
            int id )
```

Définition à la ligne 61 du fichier Occupants.java.

Référencé par com.example.groom.IHMConnexion.initialiserRetraitOccupant().

```
00062     {
00063         return bdd.delete("occupants", "idOccupant = " + id, null);
00064     }
```

7.8.4 Documentation des données membres

7.8.4.1 bdd

```
SQLiteDatabase com.example.groom.Occupants.bdd  [private]
```

Définition à la ligne 14 du fichier Occupants.java.

7.8.4.2 groomBDD

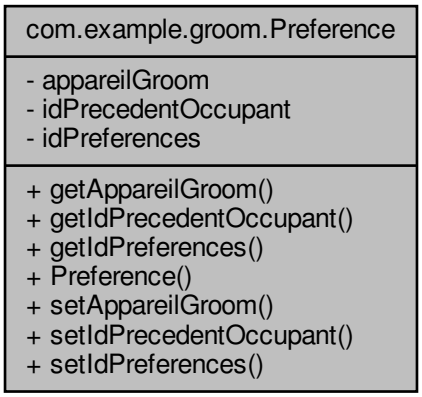
```
GroomBDD com.example.groom.Occupants.groomBDD  [private]
```

Définition à la ligne 13 du fichier Occupants.java.

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

7.9 Référence de la classe com.example.groom.Preference

Graphe de collaboration de com.example.groom.Preference :



Fonctions membres publiques

- String getAppareilGroom ()
- int getIdPrecedentOccupant ()
- int getIdPreferences ()
- Preference ()
- void setAppareilGroom (String appareilGroom)
- void setIdPrecedentOccupant (int idPrecedentOccupant)
- void setIdPreferences (int idPreferences)

Attributs privés

- String appareilGroom
- int idPrecedentOccupant
- int idPreferences

7.9.1 Description détaillée

Définition à la ligne 3 du fichier Preference.java.

7.9.2 Documentation des constructeurs et destructeur

7.9.2.1 Preference()

```
com.example.groom.Preference.Preference ( )
```

Définition à la ligne 9 du fichier Preference.java.

```
00010     {
00011         appareilGroom = "";
00012         idPrecedentOccupant = 0;
00013     }
```

7.9.3 Documentation des fonctions membres

7.9.3.1 getAppareilGroom()

```
String com.example.groom.Preference.getAppareilGroom ( )
```

Définition à la ligne 25 du fichier Preference.java.

Références com.example.groom.Preference.appareilGroom.

```
00026     {
00027         return this.appareilGroom;
00028     }
```

7.9.3.2 getIdPrecedentOccupant()

```
int com.example.groom.Preference.getIdPrecedentOccupant ( )
```

Définition à la ligne 35 du fichier Preference.java.

Références com.example.groom.Preference.idPrecedentOccupant.

```
00036     {
00037         return this.idPrecedentOccupant;
00038     }
```

7.9.3.3 getIdPreferences()

```
int com.example.groom.Preference.getIdPreferences ( )
```

Définition à la ligne 15 du fichier Preference.java.

Références com.example.groom.Preference.idPreferences.

```
00016     {
00017         return this.idPreferences;
00018     }
```

7.9.3.4 setAppareilGroom()

```
void com.example.groom.Preference.setAppareilGroom (
             String appareilGroom )
```

Définition à la ligne 30 du fichier Preference.java.

Références com.example.groom.Preference.appareilGroom.

Référencé par com.example.groom.Preferences.cursorToServeur().

```
00031     {
00032         this.appareilGroom = appareilGroom;
00033     }
```

7.9.3.5 setIdPrecedentOccupant()

```
void com.example.groom.Preference.setIdPrecedentOccupant (
             int idPrecedentOccupant )
```

Définition à la ligne 40 du fichier Preference.java.

Références com.example.groom.Preference.idPrecedentOccupant.

Référencé par com.example.groom.Preferences.cursorToServeur().

```
00041     {
00042         this.idPrecedentOccupant = idPrecedentOccupant;
00043     }
```

7.9.3.6 setIdPreferences()

```
void com.example.groom.Preference.setIdPreferences (
            int idPreferences )
```

Définition à la ligne 20 du fichier Preference.java.

Références com.example.groom.Preference.idPreferences.

Référencé par com.example.groom.Preferences.cursorToServeur().

```
00021     {
00022         this.idPreferences = idPreferences;
00023     }
```

7.9.4 Documentation des données membres

7.9.4.1 appareilGroom

```
String com.example.groom.Preference.appareilGroom  [private]
```

Définition à la ligne 6 du fichier Preference.java.

Référencé par com.example.groom.Preference.getAppareilGroom(), et com.example.groom.Preference.setAppareilGroom().

7.9.4.2 idPrecedentOccupant

```
int com.example.groom.Preference.idPrecedentOccupant  [private]
```

Définition à la ligne 7 du fichier Preference.java.

Référencé par com.example.groom.Preference.getIdPrecedentOccupant(), et com.example.groom.Preference.setIdPrecedent↩Occupant().

7.9.4.3 idPreferences

```
int com.example.groom.Preference.idPreferences  [private]
```

Définition à la ligne 5 du fichier Preference.java.

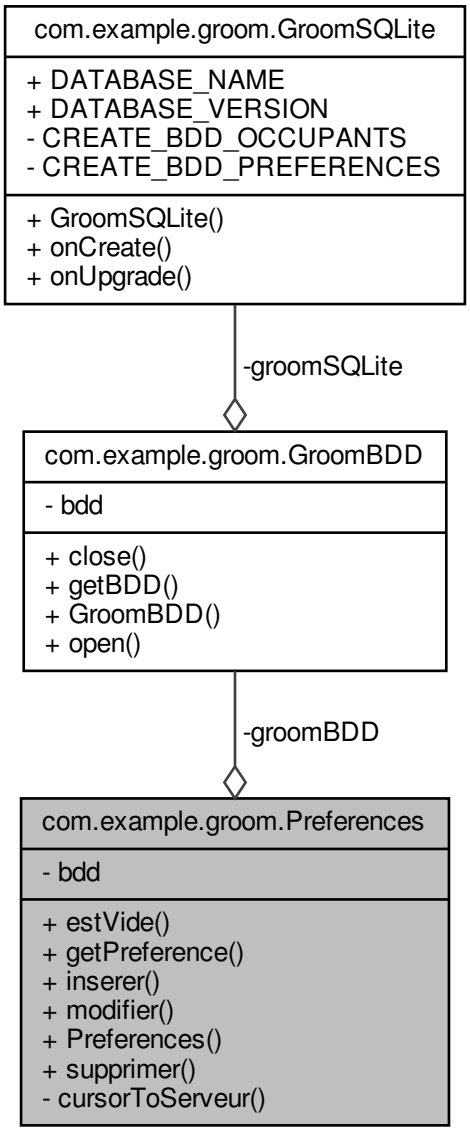
Référencé par com.example.groom.Preference.getIdPreferences(), et com.example.groom.Preference.setIdPreferences().

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

— Preference.java

7.10 Référence de la classe com.example.groom.Preferences

Graphe de collaboration de com.example.groom.Preferences :



Fonctions membres publiques

- boolean estVide ()
- Preference getPreference ()
- long inserer (String appareilGroom, int idPrecedentOccupant)
- int modifier (int id, String appareilGroom, int idPrecedentOccupant)
- Preferences (Context context)
- int supprimer (int id)

Fonctions membres privées

- Preference cursorToServeur (Cursor c, boolean one)

Attributs privés

— SQLiteDatabase bdd
— GroomBDD groomBDD

7.10.1 Description détaillée

Définition à la ligne 8 du fichier Preferences.java.

7.10.2 Documentation des constructeurs et destructeur

7.10.2.1 Preferences()

```
com.example.groom.Preferences.Preferences (
            Context context )
```

Définition à la ligne 13 du fichier Preferences.java.

Références com.example.groom.GroomBDD.getBDD(), et com.example.groom.GroomBDD.open().

```
00014     {
00015         groomBDD = new GroomBDD(context);
00016         groomBDD.open();
00017         bdd = groomBDD.getBDD();
00018     }
```

7.10.3 Documentation des fonctions membres

7.10.3.1 cursorToServeur()

```
Preference com.example.groom.Preferences.cursorToServeur (
            Cursor c,
            boolean one )  [private]
```

Définition à la ligne 70 du fichier Preferences.java.

Références com.example.groom.Preference.setAppareilGroom(), com.example.groom.Preference.setIdPrecedentOccupant(), et com.example.groom.Preference.setIdPreferences().

Référencé par com.example.groom.Preferences.getPreference().

```
00071     {
00072         if (c.getCount() == 0)
00073             return null;
00074 
00075         if(one)
00076             c.moveToFirst();
00077 
00078       Preference preference = new Preference();
00079       preference.setIdPreferences(c.getInt(0));
00080       preference.setAppareilGroom(c.getString(1));
00081       preference.setIdPrecedentOccupant(c.getInt(2));
00082 
00083       if (one)
00084           c.close();
00085       return preference;
00086     }
```

7.10.3.2 estVide()

```
boolean com.example.groom.Preferences.estVide ( )
```

Définition à la ligne 56 du fichier Preferences.java.

```
00057     {
00058         Cursor c = bdd.rawQuery("SELECT * FROM preferences", null);
00059 
00060         if(c.getCount() == 0)
00061         {
00062             return true;
00063         }
00064         else
00065         {
00066             return false;
00067         }
00068     }
```

7.10.3.3 getPreference()

```
Preference com.example.groom.Preferences.getPreference ( )
```

Définition à la ligne 20 du fichier Preferences.java.

Références com.example.groom.Preferences.cursorToServeur().

Référencé par com.example.groom.IHMConnexion.lancerRecherchePeripherique().

```
00021     {
00022        Preference preference;
00023        Cursor cursor = bdd.query("preferences", new String[] {"idPreferences", "groom", "
      idPrecedentOccupant"},
00024                    null, null, null, null, null, null);
00025 
00026         cursor.moveToFirst();
00027         preference = cursorToServeur(cursor, false);
00028 
00029         cursor.close();
00030         return preference;
00031     }
```

7.10.3.4 inserer()

```
long com.example.groom.Preferences.inserer (
             String appareilGroom,
             int idPrecedentOccupant )
```

Définition à la ligne 33 du fichier Preferences.java.

```
00034     {
00035         ContentValues values = new ContentValues();
00036         values.put("groom", appareilGroom);
00037         values.put("idPrecedentOccupant", idPrecedentOccupant);
00038 
00039         return bdd.insert("preferences", null, values);
00040     }
```

7.10.3.5 modifier()

```
int com.example.groom.Preferences.modifier (
            int id,
            String appareilGroom,
            int idPrecedentOccupant )
```

Définition à la ligne 42 du fichier Preferences.java.

```
00043     {
00044         ContentValues values = new ContentValues();
00045         values.put("groom", appareilGroom);
00046         values.put("idPrecedentOccupant", idPrecedentOccupant);
00047 
00048         return bdd.update("preferences", values, "idPreferences = " + id, null);
00049     }
```

7.10.3.6 supprimer()

```
int com.example.groom.Preferences.supprimer (
            int id )
```

Définition à la ligne 51 du fichier Preferences.java.

```
00052     {
00053         return bdd.delete("preferences", "idOccupant = " + id, null);
00054     }
```

7.10.4 Documentation des données membres

7.10.4.1 bdd

```
SQLiteDatabase com.example.groom.Preferences.bdd  [private]
```

Définition à la ligne 11 du fichier Preferences.java.

7.10.4.2 groomBDD

```
GroomBDD com.example.groom.Preferences.groomBDD  [private]
```

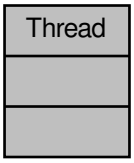
Définition à la ligne 10 du fichier Preferences.java.

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

— Preferences.java

7.11 Référence de la classe Thread

Graphe de collaboration de Thread :

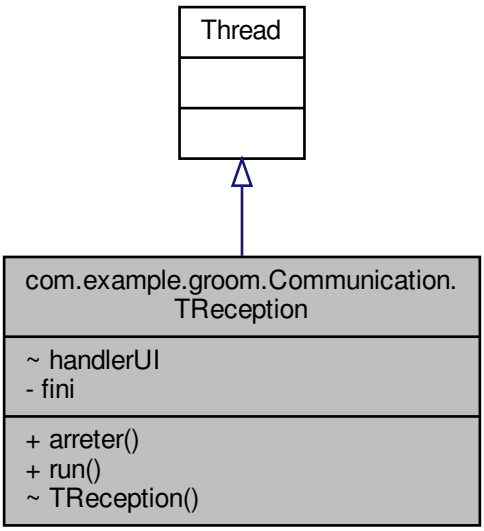


La documentation de cette classe a été générée à partir du fichier suivant :
— Communication.java

7.12 Référence de la classe com.example.groom.Communication.TReception

Déclaration de la classe TReception.

Graphe de collaboration de com.example.groom.Communication.TReception :



Fonctions membres publiques

— void arreter ()
Méthode pour arrêter la réception.
— void run ()
Méthode qui lance le thread.

Attributs privés

— boolean fini
true si la réception est fini, false sinon

7.12.1 Description détaillée

Déclaration de la classe TReception.

Définition à la ligne 233 du fichier Communication.java.

7.12.2 Documentation des fonctions membres

7.12.2.1 arreter()

`com.example.groom.Communication.TReception.arreter ( )`

Méthode pour arrêter la réception.

Définition à la ligne 301 du fichier Communication.java.

Référencé par com.example.groom.Communication.deconnecter().

```
00302         {
00303             if (!fini)
00304             {
00305                 fini = true;
00306             }
00307             try
00308             {
00309                 Thread.sleep(250);
00310             }
00311             catch (InterruptedException e)
00312             {
00313                 e.printStackTrace();
00314             }
00315         }
```

7.12.2.2 run()

`com.example.groom.Communication.TReception.run ( )`

Méthode qui lance le thread.

Définition à la ligne 259 du fichier Communication.java.

Références com.example.groom.Communication.CODE_RECEPTION.

```
00260         {
00261             BufferedReader reception = new BufferedReader(new InputStreamReader(
      receiveStream));
00262             while (!fini)
00263             {
00264                 try
00265                 {
00266                     String trame = "";
00267                     if (reception.ready())
00268                     {
00269                         trame = reception.readLine();
00270                     }
00271                     if (trame.length() > 0)
00272                     {
00273                         Log.d(TAG, "run() trame : " + trame);
00274                         Message msg = Message.obtain();
00275                         msg.what = Communication.CODE_RECEPTION;
00276                         msg.obj = trame;
00277                         handlerUI.sendMessage(msg);
00278                     }
00279                 }
00280                 catch (IOException e)
00281                 {
00282                     Log.d(TAG, "Erreur read()");
00283                     e.printStackTrace();
00284                 }
00285                 try
00286                 {
00287                     Thread.sleep(250);
00288                 }
00289                 catch (InterruptedException e)
00290                 {
00291                     e.printStackTrace();
00292                 }
00293             }
00294         }
```

7.12.3 Documentation des données membres

7.12.3.1 fini

```
boolean com.example.groom.Communication.TReception.fini  [private]
```

true si la réception est fini, false sinon

Définition à la ligne 239 du fichier Communication.java.

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

8 Documentation des fichiers

8.1 Référence du fichier Changelog.md

8.2 Changelog.md

```
00001 \page page_changelog Changelog
00002
00003 r56 | tvaira | 2020-03-27 16:51:01 +0100 (ven. 27 mars 2020) | 1 ligne
00004
00005 Validation émission à la connexion
00006
00007 r55 | tvaira | 2020-03-27 16:15:31 +0100 (ven. 27 mars 2020) | 1 ligne
00008
00009 Validation réception de trames
00010
00011 r54 | tvaira | 2020-03-27 15:58:46 +0100 (ven. 27 mars 2020) | 1 ligne
00012
```

```
00013 Validation émission de trames
00014
00015 r53 | geyraud | 2020-03-27 13:25:39 +0100 (ven. 27 mars 2020) | 1 ligne
00016
00017 envoie et reception de la trame + notification si quelqu'un a sonné ou si quelqu'un est présent
00018
00019 r52 | tvaira | 2020-03-26 09:47:26 +0100 (jeu. 26 mars 2020) | 1 ligne
00020
00021
00022 r51 | geyraud | 2020-03-25 16:00:45 +0100 (mer. 25 mars 2020) | 1 ligne
00023
00024 Début communication Bluetooth
00025
00026 r50 | tvaira | 2020-03-18 08:02:03 +0100 (mer. 18 mars 2020) | 1 ligne
00027
00028 fichier à ignorer
00029
00030 r49 | tvaira | 2020-03-18 08:00:23 +0100 (mer. 18 mars 2020) | 1 ligne
00031
00032 fichier à ignorer
00033
00034 r47 | geyraud | 2020-03-13 16:44:35 +0100 (ven. 13 mars 2020) | 1 ligne
00035
00036 Affichage de la liste des appareils bluetooth appairés
00037
00038 r42 | geyraud | 2020-03-12 12:28:21 +0100 (jeu. 12 mars 2020) | 1 ligne
00039
00040 Ajout de la classe Communication.java
00041
00042 r41 | geyraud | 2020-03-12 12:25:06 +0100 (jeu. 12 mars 2020) | 1 ligne
00043
00044 Changement du nom de la classe Connexion.java en Communication.java
00045
00046 r38 | geyraud | 2020-03-11 15:27:34 +0100 (mer. 11 mars 2020) | 1 ligne
00047
00048 Sauvegarde des changements des attributs de l'objet groom de IHMGroom dans l'obejt groom de
       IHMConnexion après l'appel de finish()
00049
00050 r36 | geyraud | 2020-03-10 16:51:36 +0100 (mar. 10 mars 2020) | 1 ligne
00051
00052 Documentation du code avec doxygen
00053
00054 r35 | tvaira | 2020-03-08 11:15:51 +0100 (dim. 08 mars 2020) | 1 ligne
00055
00056 Révision de code
00057
00058 r34 | geyraud | 2020-03-06 17:05:38 +0100 (ven. 06 mars 2020) | 1 ligne
00059
00060 Ajout de la classe Groom et affichage de la dernière disponibilité choisie
00061
00062 r32 | geyraud | 2020-03-06 15:58:37 +0100 (ven. 06 mars 2020) | 1 ligne
00063
00064 Suppression de l'activité MainActivity
00065
00066 r31 | geyraud | 2020-03-06 14:54:59 +0100 (ven. 06 mars 2020) | 1 ligne
00067
00068 Ajout de la classe Communication, des nouvelles activités qui n'étaient pas envoyées, et des fichiers
       design des activités correspondantes
00069
00070 r27 | geyraud | 2020-03-05 16:20:43 +0100 (jeu. 05 mars 2020) | 1 ligne
00071
00072 Création d'une classe IHMConnexion
00073
00074 r23 | geyraud | 2020-03-04 16:12:56 +0100 (mer. 04 mars 2020) | 1 ligne
00075
00076 Création de l'application Groom
```

8.3 Référence du fichier Communication.java

Déclaration de la classe Communication.

Classes

— class com.example.groom.Communication
Déclaration de la classe Communication.
— class com.example.groom.Communication.TReception
Déclaration de la classe TReception.

Paquetages

— package com.example.groom

8.3.1 Description détaillée

Déclaration de la classe Communication.

Auteur

Grégory Eyraud

Définition dans le fichier Communication.java.

8.4 Communication.java

```
00001 package com.example.groom;
00002
00003 import android.bluetooth.BluetoothDevice;
00004 import android.bluetooth.BluetoothSocket;
00005 import android.os.Handler;
00006 import android.os.Message;
00007 import android.util.Log;
00008 import java.io.BufferedReader;
00009 import java.io.IOException;
00010 import java.io.InputStream;
00011 import java.io.InputStreamReader;
00012 import java.io.OutputStream;
00013 import java.util.UUID;
00014
00025 public class Communication extends Thread
00026 {
00030     private static final String TAG = "Communication";
00031     public final static int CODE_CONNEXION = 0;
00032     public final static int CODE_RECEPTION = 1;
00033     public final static int CODE_DECONNEXION = 2;
00034
00038     private BluetoothDevice device;
00039     private String nom;
00040     private String adresse;
00041     private Handler handler;
00042     private BluetoothSocket socket = null;
00043     private InputStream receiveStream = null;
00044     private OutputStream sendStream = null;
00045     private TReception tReception;
00046
00053     public Communication(BluetoothDevice device, Handler handler)
00054     {
00055         this.handler = handler;
00056         if (device != null)
00057         {
00058             this.device = device;
00059             this.nom = device.getName();
00060             this.adresse = device.getAddress();
00061
00062             try
00063             {
00064                 socket = device.createRfcommSocketToServiceRecord(UUID.fromString("
      00001101-0000-1000-8000-00805F9B34FB"));
00065                 receiveStream = socket.getInputStream();
00066                 sendStream = socket.getOutputStream();
00067             }
00068             catch (IOException e)
00069             {
00070                 e.printStackTrace();
00071             }
00072         }
00073         else
00074         {
00075             this.device = device;
00076             this.nom = "Aucun";
00077             this.adresse = "";
00078         }
00079
00080         if(socket != null)
00081             tReception = new TReception(handler);
00082         Log.v(TAG, "Communication() : Nom = " + getNom());
00083     }
```

```
00084
00091     public String getNom()
00092     {
00093         return nom;
00094     }
00095
00102     public String getAdresse()
00103     {
00104         return adresse;
00105     }
00106
00113     public BluetoothDevice getDevice()
00114     {
00115         return device;
00116     }
00117
00123     public void connecter()
00124     {
00125         new Thread()
00126         {
00127             @Override public void run()
00128             {
00129                 try
00130                 {
00131                     socket.connect();
00132
00133                     Message msg = Message.obtain();
00134                     msg.what = CODE_CONNEXION;
00135                     handler.sendMessage(msg);
00136
00137                     if(tReception != null)
00138                         tReception.start();
00139                     Log.v(TAG, "connecter() : Nom = " + getNom());
00140                 }
00141                 catch (IOException e)
00142                 {
00143                     Log.d(TAG, "Erreur connect()");
00144                     e.printStackTrace();
00145                 }
00146             }
00147         }.start();
00148     }
00149
00156     public boolean deconnecter()
00157     {
00158         try
00159         {
00160             tReception.arreter();
00161
00162             socket.close();
00163
00164             Message msg = Message.obtain();
00165             msg.what = CODE_DECONNEXION;
00166             handler.sendMessage(msg);
00167
00168             return true;
00169         }
00170         catch (IOException e)
00171         {
00172             Log.d(TAG, "Erreur close()");
00173             e.printStackTrace();
00174             return false;
00175         }
00176     }
00177
00184     public void envoyer(String data)
00185     {
00186         final String trame = data;
00187
00188         if(socket == null)
00189             return;
00190
00191         new Thread()
00192         {
00193             @Override public void run()
00194             {
00195                 try
00196                 {
00197                     if(socket.isConnected())
00198                     {
00199                         sendStream.write(trame.getBytes());
00200                         sendStream.flush();
00201                         Log.d(TAG, "envoyer() trame : " + trame);
00202                     }
00203                 }
00204                 catch (IOException e)
00205                 {
00206                     Log.d(TAG, "Erreur write()");
00207                     e.printStackTrace();
00208                 }
00209             }
```

```
00210         }.start();
00211     }
00212
00217     public void attendre(int millis)
00218     {
00219         try
00220         {
00221             Thread.sleep(millis);
00222         }
00223         catch (InterruptedException e)
00224         {
00225             e.printStackTrace();
00226         }
00227     }
00228
00233     private class TReception extends Thread
00234     {
00238         Handler handlerUI;
00239         private boolean fini;
00240
00247         TReception(Handler h)
00248         {
00249             handlerUI = h;
00250             fini = false;
00251         }
00252
00258         @Override
00259         public void run()
00260         {
00261             BufferedReader reception = new BufferedReader(new InputStreamReader(receiveStream));
00262             while (!fini)
00263             {
00264                 try
00265                 {
00266                     String trame = "";
00267                     if (reception.ready())
00268                     {
00269                         trame = reception.readLine();
00270                     }
00271                     if (trame.length() > 0)
00272                     {
00273                         Log.d(TAG, "run() trame : " + trame);
00274                         Message msg = Message.obtain();
00275                         msg.what = Communication.CODE_RECEPTION;
00276                         msg.obj = trame;
00277                         handlerUI.sendMessage(msg);
00278                     }
00279                 }
00280                 catch (IOException e)
00281                 {
00282                     Log.d(TAG, "Erreur read()");
00283                     e.printStackTrace();
00284                 }
00285                 try
00286                 {
00287                     Thread.sleep(250);
00288                 }
00289                 catch (InterruptedException e)
00290                 {
00291                     e.printStackTrace();
00292                 }
00293             }
00294         }
00295
00301         public void arreter()
00302         {
00303             if (!fini)
00304             {
00305                 fini = true;
00306             }
00307             try
00308             {
00309                 Thread.sleep(250);
00310             }
00311             catch (InterruptedException e)
00312             {
00313                 e.printStackTrace();
00314             }
00315         }
00316     }
00317 }
00318
00319
```

8.5 Référence du fichier Groom.java

Déclaration de la classe Groom.

Classes

— class com.example.groom.Groom
Déclaration de la classe Groom.

Paquetages

— package com.example.groom

8.5.1 Description détaillée

Déclaration de la classe Groom.

Auteur

Grégory Eyraud

Définition dans le fichier Groom.java.

8.6 Groom.java

```
00001 package com.example.groom;
00002
00009 import java.io.Serializable;
00010
00015 public class Groom implements Serializable
00016 {
00020     private static final String TAG = "Groom";
00021
00025     private Occupant occupant;
00026     private String disponibilite;
00027     private boolean modeSonnette;
00028     private boolean detectionPresence;
00029     private String nomDevice;
00030
00042     public Groom(String nom, String prenom, String fonction, String disponibilite, boolean
      modeSonnette, boolean detectionPresence)
00043     {
00044         occupant = new Occupant(nom, prenom , fonction);
00045         this.disponibilite = disponibilite;
00046         this.modeSonnette = modeSonnette;
00047         this.detectionPresence = detectionPresence;
00048     }
00049
00050     public Occupant getOccupant()
00051     {
00052         return occupant;
00053     }
00054
00055     public void setOccupant(Occupant occupant)
00056     {
00057         this.occupant = occupant;
00058     }
00059
00066     public String getDisponibilite()
00067     {
00068         return this.disponibilite;
00069     }
00070
00077     public int getDisponibiliteToInt()
00078     {
00079         int dispo = 0;
00080
00081         if(this.disponibilite.equals("Libre"))
00082             dispo = 0;
00083         else if(this.disponibilite.equals("Absent"))
00084             dispo = 1;
00085         else if(this.disponibilite.equals("Occupé"))
00086             dispo = 2;
00087
00088         return dispo;
00089     }
00090
00097     public void setDisponibiliteToInt(int dispo)
```

```
00098     {
00099         if(dispo == 0)
00100             this.disponibilite = "Libre";
00101         else if(dispo == 1)
00102             this.disponibilite = "Absent";
00103         else if(dispo == 2)
00104             this.disponibilite = "Occupé";
00105     }
00106
00113     public void setDisponibilite(String disponibilite)
00114     {
00115         this.disponibilite = disponibilite;
00116     }
00117
00124     public boolean getModeSonnette()
00125     {
00126         return this.modeSonnette;
00127     }
00128
00135     public void setModeSonnette(boolean modeSonnette)
00136     {
00137         this.modeSonnette = modeSonnette;
00138     }
00139
00146     public boolean getDetectionPresence()
00147     {
00148         return this.detectionPresence;
00149     }
00150
00157     public void setDetectionPresence(boolean detectionPresence)
00158     {
00159         this.detectionPresence = detectionPresence;
00160     }
00161
00168     public String getNomDevice()
00169     {
00170         return this.nomDevice;
00171     }
00172
00179     public void setNomDevice(String nomDevice)
00180     {
00181         this.nomDevice = nomDevice;
00182     }
00183 }
```

8.7 Référence du fichier GroomBDD.java

Classes

— class com.example.groom.GroomBDD

Paquetages

— package com.example.groom

8.8 GroomBDD.java

```
00001 package com.example.groom;
00002
00003 import android.content.Context;
00004 import android.database.sqlite.SQLiteDatabase;
00005
00006 public class GroomBDD
00007 {
00008     private SQLiteDatabase bdd = null;
00009     private GroomSQLite groomSQLite = null;
00010
00011     public GroomBDD(Context context)
00012     {
00013         groomSQLite = new GroomSQLite(context);
00014     }
00015
00016     public void open()
00017     {
00018         if (bdd == null)
00019             bdd = groomSQLite.getWritableDatabase();
00020     }
```

```
00021
00022     public void close()
00023     {
00024         if (bdd != null)
00025             if (bdd.isOpen())
00026                 bdd.close();
00027     }
00028
00029     public SQLiteDatabase getBDD()
00030     {
00031         if (bdd == null)
00032             open();
00033         return bdd;
00034     }
00035 }
```

8.9 Référence du fichier GroomSQLite.java

Classes

— class com.example.groom.GroomSQLite

Paquetages

— package com.example.groom

8.10 GroomSQLite.java

```
00001 package com.example.groom;
00002
00003 import android.content.Context;
00004 import android.database.sqlite.SQLiteDatabase;
00005 import android.database.sqlite.SQLiteOpenHelper;
00006
00007 public class GroomSQLite extends SQLiteOpenHelper
00008 {
00009     public static final String DATABASE_NAME = "groom.db";
00010     public static final int DATABASE_VERSION = 1;
00011     private static final String CREATE_BDD_OCCUPANTS =
00012             "CREATE TABLE occupants (" +
00013                     "idOccupant INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL," +
00014                     "nom VARCHAR(45) NULL," +
00015                     "prenom VARCHAR(45) NULL," +
00016                     "fonction VARCHAR(45) NULL);";
00017     private static final String CREATE_BDD_PREFERENCES =
00018             "CREATE TABLE preferences (" +
00019                     "idPreferences INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL," +
00020                     "groom VARCHAR(45) NULL," +
00021                     "idPrecedentOccupant INTEGER NOT NULL, " +
00022                     "FOREIGN KEY (idPrecedentOccupant) REFERENCES occupants (idOccupant);";
00023
00024     public GroomSQLite(Context context)
00025     {
00026         super(context, DATABASE_NAME, null, DATABASE_VERSION);
00027     }
00028
00029     @Override
00030     public void onCreate(SQLiteDatabase db)
00031     {
00032         db.execSQL(CREATE_BDD_OCCUPANTS);
00033         db.execSQL(CREATE_BDD_PREFERENCES);
00034     }
00035
00036     @Override
00037     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
00038     {
00039         db.execSQL("DROP TABLE occupants;");
00040         db.execSQL("DROP TABLE preferences");
00041     }
00042 }
```

8.11 Référence du fichier IHMConnexion.java

Déclaration de la classe IHMConnexion.

Classes

— class com.example.groom.IHMConnexion
Déclaration de la classe IHMConnexion.

Paquetages

— package com.example.groom

8.11.1 Description détaillée

Déclaration de la classe IHMConnexion.

Auteur

Grégory Eyraud

Définition dans le fichier IHMConnexion.java.

8.12 IHMConnexion.java

```
00001 package com.example.groom;
00002
00003 import android.app.AlertDialog;
00004 import android.bluetooth.BluetoothAdapter;
00005 import android.bluetooth.BluetoothDevice;
00006 import android.content.DialogInterface;
00007 import android.content.Intent;
00008 import android.os.Bundle;
00009 import android.util.Log;
00010 import android.view.View;
00011 import android.widget.AdapterView;
00012 import android.widget.ArrayAdapter;
00013 import android.widget.Button;
00014 import android.widget.EditText;
00015 import android.widget.Spinner;
00016 import android.widget.Toast;
00017 import java.io.Serializable;
00018 import java.util.ArrayList;
00019 import java.util.List;
00020 import java.util.Set;
00021
00022 import androidx.appcompat.app.AppCompatActivity;
00023
00034 public class IHMConnexion extends AppCompatActivity implements View.OnClickListener {
00039     private static final String TAG = "IHMConnexion";
00040     private static final int ACTION_GROOM = 1;
00041
00045     private Button boutonConnexion;
00046     private Button boutonAjoutOccupant;
00047     private Button boutonRetraitOccupant;
00048     private Spinner listeGroom;
00049     private Spinner listeOccupant;
00050     private AlertDialog.Builder ajoutOccupant;
00051     private AlertDialog.Builder retraitOccupant;
00052     private EditText nomOccupant;
00053     private EditText prenomOccupant;
00054     private EditText fonctionOccupant;
00055
00059     private Groom groom = null;
00060     private BluetoothAdapter bluetoothAdapter;
00061     private Set<BluetoothDevice> devices;
00062     private List<String> listeNomsAppareilsBluetooth;
00063     private List<String> listeOccupants;
00064     private ArrayAdapter<String> adapterGroom;
00065     private ArrayAdapter<String> adapterOccupant;
00066     private List<BluetoothDevice> listeAppareilsBluetooth;
00067     private Toast toast;
00068     private Occupants occupants = null;
00069     private Preferences preferences = null;
00070
00076     @Override
00077     protected void onCreate(Bundle savedInstanceState)
00078     {
```

```
00079         super.onCreate(savedInstanceState);
00080         setContentView(R.layout.activity_communication);
00081 
00082         groom = new Groom("Eyraud", "Grégory", "étudiant", "Libre", true, false);
00083         occupants = new Occupants(this);
00084         preferences = new Preferences(this);
00085 
00086         initialiserRessourcesIHM();
00087         activerBluetooth();
00088 
00089         selectionnerGroom();
00090         selectionnerOccupant();
00091     }
00092 
00096     private void selectionnerGroom() {
00097         listeGroom.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()
00098         {
00099             @Override
00100             public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
00101             {
00102                 Log.v(TAG, "onItemSelected()");
00103                 for (int i = 0; i < listeAppareilsBluetooth.size(); i++)
00104                 {
00105                     if (listeAppareilsBluetooth.get(i).getName().equals(listeNomsAppareilsBluetooth.get(
      position)))
00106                     {
00107                         Log.v(TAG, "onItemSelected() " + listeAppareilsBluetooth.get(i).getName());
00108                         groom.setNomDevice(listeAppareilsBluetooth.get(i).getName());
00109                         break;
00110                     }
00111                 }
00112             }
00113 
00114             @Override
00115             public void onNothingSelected(AdapterView<?> parent)
00116             {
00117 
00118             }
00119         });
00120     }
00121 
00125     private void selectionnerOccupant() {
00126         listeOccupant.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
00127             @Override
00128             public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
00129             {
00130                 Log.v(TAG, "onItemSelected()" + "Occupant choisi : " + listeOccupants.get(position));
00131                 String identite[] = listeOccupants.get(position).split(" ");
00132                 groom.setOccupant(occupants.getOccupant(identite[0]));
00133             }
00134 
00135             @Override
00136             public void onNothingSelected(AdapterView<?> parent) {
00137 
00138             }
00139         });
00140     }
00145     public void activerBluetooth()
00146     {
00147         bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
00148         if (bluetoothAdapter == null)
00149         {
00150             Toast.makeText(getApplicationContext(), "Bluetooth non activé !", Toast.LENGTH_SHORT).show();
00151         }
00152         else
00153             {
00154             if (!bluetoothAdapter.isEnabled())
00155             {
00156                 Toast.makeText(getApplicationContext(), "Bluetooth non activé !", Toast.LENGTH_SHORT).show(
      );
00157                 bluetoothAdapter.enable();
00158             }
00159             else
00160                 {
00161                 Toast.makeText(getApplicationContext(), "Bluetooth activé", Toast.LENGTH_SHORT).show();
00162             }
00163         }
00164 
00165         lancerRecherchePeripherique();
00166     }
00167 
00172     public void lancerRecherchePeripherique()
00173     {
00174         devices = bluetoothAdapter.getBondedDevices();
00175 
00176         //if(!(preferences.getPreference().getAppareilGroom().equals("")))
00177         //{
00178             //listeNomsAppareilsBluetooth.add(preferences.getPreference().getAppareilGroom());
00179         //}
00180 
00181         Log.v(TAG, preferences.getPreference().toString());
```

```
00182
00183         for (BluetoothDevice bluetoothDevice : devices)
00184         {
00185             listeAppareilsBluetooth.add(bluetoothDevice);
00186             listeNomsAppareilsBluetooth.add(bluetoothDevice.getName());
00187         }
00188
00189         listerOccupant();
00190     }
00191
00192     private void listerOccupant()
00193     {
00194         List<Occupant> liste = occupants.getListe();
00195         for(int i = 0; i<liste.size(); i++)
00196         {
00197             listeOccupants.add(liste.get(i).getNom() + " " + liste.get(i).getPrenom() + " " + liste.get(i).
      getFonction());
00198         }
00199         initialiserListeDeroulante();
00200     }
00201
00206     private void initialiserRessourcesIHM()
00207     {
00208         Log.d(TAG, "initialiserRessourcesIHM()");
00209
00210         boutonConnexion = findViewById(R.id.bouttonConnexion);
00211         boutonAjoutOccupant = findViewById(R.id.bouttonAjoutOccupant);
00212         boutonRetraitOccupant = findViewById(R.id.bouttonRetraitOccupant);
00213
00214         boutonConnexion.setOnClickListener(this);
00215         boutonAjoutOccupant.setOnClickListener(this);
00216         boutonRetraitOccupant.setOnClickListener(this);
00217
00218         listeGroom = findViewById(R.id.listeGroom);
00219         listeOccupant = findViewById(R.id.listeOccupant);
00220
00221         listeNomsAppareilsBluetooth = new ArrayList<String>();
00222         listeAppareilsBluetooth = new ArrayList<BluetoothDevice>();
00223         listeOccupants = new ArrayList<String>();
00224
00225         initialiserAjoutOccupant();
00226         initialiserRetraitOccupant();
00227     }
00228
00233     private void initialiserListeDeroulante()
00234     {
00235         adapterGroom = new ArrayAdapter<String>(this, R.layout.support_simple_spinner_dropdown_item,
      listeNomsAppareilsBluetooth);
00236         adapterGroom.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
00237         listeGroom.setAdapter(adapterGroom);
00238
00239         adapterOccupant = new ArrayAdapter<String>(this, R.layout.support_simple_spinner_dropdown_item,
      listeOccupants);
00240         adapterOccupant.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
00241         listeOccupant.setAdapter(adapterOccupant);
00242     }
00243
00251     @Override
00252     protected void onActivityResult(int requestCode, int resultCode, Intent intent)
00253     {
00254         if (requestCode == ACTION_GROOM && resultCode == RESULT_OK)
00255         {
00256             groom = (Groom) intent.getSerializableExtra("Groom");
00257             Log.v(TAG, "Disponibilité = " + groom.getDisponibilite());
00258             Log.v(TAG, "Sonnette = " + groom.getModeSonnette());
00259         }
00260     }
00261
00262     @Override
00263     public void onClick(View element)
00264     {
00265         switch (element.getId())
00266         {
00267             case R.id.bouttonAjoutOccupant:
00268                 afficherToast("Ajout Occupant");
00269                 ajoutOccupant.show();
00270                 break;
00271
00272             case R.id.bouttonRetraitOccupant:
00273                 afficherToast("Retrait Occupant");
00274                 retraitOccupant.show();
00275                 break;
00276
00277             case R.id.bouttonConnexion:
00278                 afficherToast("Connexion");
00279
00280                 Intent intent = new Intent(IHMConnexion.this,
      IHMGroom.class);
00281                 intent.putExtra("Groom", (Serializable) groom);
00282                 startActivityForResult(intent, ACTION_GROOM);
00283                 /*if(preferences.getPreference() == null)
```

```
00284                   preferences.inserer(groom.getNomDevice(), listeOccupant.getId());
00285               else
00286                   preferences.modifier(0, groom.getNomDevice(), listeOccupant.getId());*/
00287               break;
00288           }
00289       }
00290
00296       private void afficherToast(String message)
00297       {
00298           toast = Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT);
00299           toast.show();
00300       }
00301
00306       private void initialiserAjoutOccupant()
00307       {
00308           ajoutOccupant = new AlertDialog.Builder(this);
00309
00310           ajoutOccupant.setMessage("Veuillez saisir vos informations : ");
00311           ajoutOccupant.setView(R.layout.ajout_occupant);
00312           ajoutOccupant.setPositiveButton("Ajouter", new DialogInterface.OnClickListener()
00313           {
00314               @Override
00315               public void onClick(DialogInterface dialog, int which)
00316               {
00317                   nomOccupant = (EditText) ((AlertDialog) dialog).findViewById(R.id.nomOccupant);
00318                   prenomOccupant = (EditText) ((AlertDialog) dialog).findViewById(R.id.prenomOccupant);
00319                   fonctionOccupant = (EditText) ((AlertDialog) dialog).findViewById(R.id.fonctionOccupant);
00320                   Log.v(TAG, "Occupant ajouté :" + "Nom = " + nomOccupant.getText().toString() + " "
00321                           + "Prenom = " + prenomOccupant.getText().toString() + " "
00322                           + "Fonction = " + fonctionOccupant.getText().toString());
00323
00324                   adapterOccupant.add(nomOccupant.getText().toString() + " " + prenomOccupant.getText().
      toString() + " " + fonctionOccupant.getText().toString());
00325
00326                   occupants.inserer(nomOccupant.getText().toString(), prenomOccupant.getText().
      toString(), fonctionOccupant.getText().toString());
00327               }
00328           });
00329           ajoutOccupant.setNegativeButton("Annuler", new DialogInterface.OnClickListener()
00330           {
00331               @Override
00332               public void onClick(DialogInterface dialog, int which)
00333               {
00334                   Log.v(TAG, "Ajout annulé");
00335               }
00336           });
00337       }
00338
00343       private void initialiserRetraitOccupant()
00344       {
00345           retraitOccupant = new AlertDialog.Builder(this);
00346           retraitOccupant.setMessage("Vous êtes sur le point de supprimer l'occupant choisit.
      \r\n\r\n" + "Êtes-vous sûr de vouloir le supprimer ?");
00347           retraitOccupant.setPositiveButton("Supprimer", new DialogInterface.OnClickListener()
00348           {
00349               @Override
00350               public void onClick(DialogInterface dialog, int which)
00351               {
00352                   Log.v(TAG, "Supression Occupant");
00353                   occupants.supprimer(groom.getOccupant().
      getId());
00354                   adapterOccupant.remove(groom.getOccupant().getNom() + " " + groom.
      getOccupant().getPrenom() + " " + groom.getOccupant().
      getFonction());
00355                   listeOccupant.setAdapter(adapterOccupant);
00356               }
00357           });
00358           retraitOccupant.setNegativeButton("Annuler", new DialogInterface.OnClickListener()
00359           {
00360               @Override
00361               public void onClick(DialogInterface dialog, int which)
00362               {
00363                   Log.v(TAG, "Suppression annulée");
00364               }
00365           });
00366       }
00367 }
```

8.13 Référence du fichier IHMGroom.java

Déclaration de la classe IHMGroom.

Classes

— class com.example.groom.IHMGroom

Déclaration de la classe IHMGroom.

Paquetages

— package com.example.groom

8.13.1 Description détaillée

Déclaration de la classe IHMGroom.

Auteur

Grégory Eyraud

Définition dans le fichier IHMGroom.java.

8.14 IHMGroom.java

```
00001 package com.example.groom;
00002
00003
00004 import android.app.AlertDialog;
00005 import android.bluetooth.BluetoothAdapter;
00006 import android.bluetooth.BluetoothDevice;
00007 import android.content.DialogInterface;
00008 import android.content.Intent;
00009 import android.os.Bundle;
00010 import android.os.Handler;
00011 import android.os.Message;
00012 import android.util.Log;
00013 import android.view.View;
00014 import android.widget.Button;
00015 import android.widget.EditText;
00016 import android.widget.TextView;
00017 import android.widget.Toast;
00018
00019 import androidx.appcompat.app.AppCompatActivity;
00020 import androidx.core.app.NotificationCompat;
00021 import androidx.core.app.NotificationManagerCompat;
00022
00023 import java.util.Set;
00024
00035 public class IHMGroom extends AppCompatActivity implements View.OnClickListener {
00039     private static final String TAG = "IHMGroom";
00040
00044     private Button boutonPerso;
00045     private Button boutonEntrer;
00046     private Button boutonAbsent;
00047     private Button boutonOccupe;
00048     private Button boutonModeSonnette;
00049     private EditText messagePerso;
00050     private Button boutonDeconnexion;
00051     private TextView disponibiliteActuelle;
00052     private AlertDialog.Builder saisieMessagePerso;
00053     private Toast toast;
00054
00058     private Groom groom = null;
00059     private BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
00060     private Communication communication = null;
00061
00068     @Override
00069     protected void onCreate(Bundle savedInstanceState)
00070     {
00071         super.onCreate(savedInstanceState);
00072         setContentView(R.layout.activity_groom);
00073
00074         Intent intent = getIntent();
00075         groom = (Groom) intent.getSerializableExtra("Groom");
00076         if (groom != null)
00077         {
00078             Log.d(TAG, "Groom : " + groom.getOccupant().getNom() + " " + groom.
      getOccupant().getPrenom() + " " + groom.getDisponibilite() + " " +
      groom.getNomDevice());
00079         }
00080
```

```
00081         Set<BluetoothDevice> devices = bluetoothAdapter.getBondedDevices();
00082         //Log.d(TAG, "Nb BluetoothDevice " + devices.size());
00083         for (BluetoothDevice bluetoothDevice : devices)
00084         {
00085             //Log.d(TAG, "BluetoothDevice " + bluetoothDevice.getName() + " [" +
       bluetoothDevice.getAddress() + "]");
00086             if(bluetoothDevice.getName().equals(groom.getNomDevice()))
00087             {
00088                 Log.d(TAG, "BluetoothDevice Groom trouvé : " + bluetoothDevice.getName() + " [" +
      bluetoothDevice.getAddress() + "]");
00089                 communication = new Communication(bluetoothDevice,
      handler);
00090             }
00091         }
00092
00093         if(communication != null)
00094         {
00095             communication.connecter();
00096         }
00097
00098         initialiserRessourcesIHM();
00099         initialiserSaisieMessagePerso();
00100     }
00101
00107     @Override
00108     protected void onStart()
00109     {
00110         super.onStart();
00111         if (groom != null)
00112             disponibiliteActuelle.setText(groom.getDisponibilite());
00113     }
00114
00120     private void initialiserRessourcesIHM()
00121     {
00122         // Les boutons
00123         boutonPerso = findViewById(R.id.bouttonMessagePersonnalise);
00124         boutonEntrer = findViewById(R.id.boutonLibre);
00125         boutonAbsent = findViewById(R.id.boutonAbsent);
00126         boutonOccupe = findViewById(R.id.boutonOccupe);
00127         boutonModeSonnette = findViewById(R.id.boutonSonnette);
00128         boutonDeconnexion = findViewById(R.id.boutonDeconnexion);
00129
00130         boutonPerso.setOnClickListener(this);
00131         boutonEntrer.setOnClickListener(this);
00132         boutonAbsent.setOnClickListener(this);
00133         boutonOccupe.setOnClickListener(this);
00134         boutonModeSonnette.setOnClickListener(this);
00135         boutonDeconnexion.setOnClickListener(this);
00136
00137         disponibiliteActuelle = (TextView) findViewById(R.id.disponibiliteActuelle);
00138
00139         verifierModeSonnette();
00140     }
00141
00147     private void initialiserSaisieMessagePerso()
00148     {
00149         saisieMessagePerso = new AlertDialog.Builder(this);
00150
00151         saisieMessagePerso.setMessage("Veuillez saisir votre message :");
00152         saisieMessagePerso.setView(R.layout.saisie_message_perso);
00153         saisieMessagePerso.setPositiveButton("Ok", new DialogInterface.OnClickListener()
      {
00154             @Override
00155             public void onClick(DialogInterface dialog, int which) {
00156                 messagePerso = (EditText) ((AlertDialog) dialog).findViewById(R.id.saisieMessagePerso);
00157                 Log.v(TAG, "Saisie : " + messagePerso.getText().toString());
00158                 communication.envoyer("$MSGPERSO;" + messagePerso.getText() + "\r\n");
00159             }
00160         });
00161         saisieMessagePerso.setNegativeButton("Annuler", new DialogInterface.
      OnClickListener() {
00162             @Override
00163             public void onClick(DialogInterface dialog, int which) {
00164                 Log.v(TAG, "Saisie annulée");
00165             }
00166         });
00167     }
00168
00175     @Override
00176     public void onClick(View element)
00177     {
00178         switch (element.getId()) {
00179             case R.id.bouttonMessagePersonnalise:
00180                 saisieMessagePerso.show();
00181                 break;
00182
00183             case R.id.boutonLibre:
00184                 communication.envoyer("$GROOM;0;" + boolToInt(groom.
      getModeSonnette()) + ";" + boolToInt(groom.
      getDetectionPresence()) + "\r\n");
00185                 if (groom != null)
```

```
00186                   groom.setDisponibilite("Libre");
00187               break;
00188
00189           case R.id.boutonAbsent:
00190               communication.envoyer("$GROOM;1;" + boolToInt(groom.
      getModeSonnette()) + ";" + boolToInt(groom.
      getDetectionPresence()) + "\r\n");
00191               if (groom != null)
00192                   groom.setDisponibilite("Absent");
00193               break;
00194
00195           case R.id.boutonOccupe:
00196               communication.envoyer("$GROOM;2;" + boolToInt(groom.
      getModeSonnette()) + ";" + boolToInt(groom.
      getDetectionPresence()) + "\r\n");
00197               if (groom != null)
00198                   groom.setDisponibilite("Occupé");
00199               break;
00200
00201           case R.id.boutonSonnette:
00202               if (boutonModeSonnette.getText().equals(getString(R.string.desactiverSonnette)))
00203               {
00204                   if (groom != null)
00205                   {
00206                       groom.setModeSonnette(false);
00207                       communication.envoyer("$GROOM;" + groom.
      getDisponibiliteToInt() + ";" + boolToInt(groom.
      getModeSonnette()) + ";" + boolToInt(groom.
      getDetectionPresence()) + "\r\n");
00208                       boutonModeSonnette.setText(R.string.activerSonnette);
00209                   }
00210               }
00211               else
00212               {
00213                   if (groom != null)
00214                   {
00215                       groom.setModeSonnette(true);
00216                       communication.envoyer("$GROOM;" + groom.
      getDisponibiliteToInt() + ";" + boolToInt(groom.
      getModeSonnette()) + ";" + boolToInt(groom.
      getDetectionPresence()) + "\r\n");
00217                       boutonModeSonnette.setText(R.string.desactiverSonnette);
00218                   }
00219               }
00220               break;
00221
00222           case R.id.boutonDeconnexion:
00223               afficherToast("Déconnexion");
00224               communication.deconnecter();
00225               finish();
00226               break;
00227       }
00228
00229       if (groom != null)
00230           disponibiliteActuelle.setText(groom.getDisponibilite());
00231   }
00232
00239   private void afficherToast(String message)
00240   {
00241       toast = Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT);
00242       toast.show();
00243   }
00244
00250   @Override
00251   public void finish()
00252   {
00253       if(communication != null)
00254           communication.deconnecter();
00255       Intent data = new Intent();
00256       data.putExtra("Groom", groom);
00257       setResult(RESULT_OK, data);
00258       super.finish();
00259   }
00260
00266   private void verifierModeSonnette()
00267   {
00268       if(groom.getModeSonnette())
00269       {
00270           boutonModeSonnette.setText(R.string.desactiverSonnette);
00271       }
00272       else
00273       {
00274           boutonModeSonnette.setText(R.string.activerSonnette);
00275       }
00276   }
00277
00281   final private Handler handler = new Handler()
00282   {
00283       @Override
00284       public void handleMessage(Message msg)
00285       {
```

```
00286             super.handleMessage(msg);
00287
00288             switch(msg.what)
00289             {
00290                 case Communication.CODE_CONNEXION:
00291                     Log.v(TAG, "Groom connecté");
00292                     communication.envoyer("$AFFICHAGE;" + groom.
      getOccupant().getNom() + ";" + groom.getOccupant().
      getPrenom() + ";" + groom.getOccupant().getFonction() + "\r\n");
00293                     break;
00294                 case Communication.CODE_RECEPTION:
00295                     Log.v(TAG, "Trame reçue " + msg.obj);
00296                     String trame[] = msg.obj.toString().split(";");
00297                     decoderTrameRecue(trame);
00298                     break;
00299                 case Communication.CODE_DECONNEXION:
00300                     Log.v(TAG, "Groom déconnecté");
00301                     break;
00302             }
00303         }
00304     };
00305
00313     private void decoderTrameRecue(String trame[])
00314     {
00315         if (trame[0].equals("$GROOM"))
00316         {
00317             groom.setDisponibiliteToInt(Integer.parseInt(trame[1]));
00318             disponibiliteActuelle.setText(groom.getDisponibilite());
00319             if(trame[2].equals("1"))
00320             {
00321                 creerNotification("Quelqu'un vient de sonner à votre porte", 1);
00322             }
00323             if(trame[3].equals("1"))
00324             {
00325                 creerNotification("Une personne attend devant votre bureau", 2);
00326             }
00327         }
00328     }
00329
00338     private void creerNotification(String texte, int id)
00339     {
00340         NotificationCompat.Builder builder = new NotificationCompat.Builder(this, "channelId")
00341                 .setSmallIcon(R.drawable.ic_launcher_background)
00342                 .setContentTitle("GrOOm")
00343                 .setContentText(texte)
00344                 .setPriority(NotificationCompat.PRIORITY_DEFAULT);
00345         NotificationManagerCompat notificationSonnette = NotificationManagerCompat.from(this);
00346         notificationSonnette.notify(id, builder.build());
00347         Log.v(TAG, "Notification Groom : " + texte);
00348     }
00349
00358     private int boolToInt(boolean b)
00359     {
00360         return b ? 1 : 0;
00361     }
00362 }
```

8.15 Référence du fichier Occupant.java

Classes

— class com.example.groom.Occupant

Paquetages

— package com.example.groom

8.16 Occupant.java

```
00001 package com.example.groom;
00002
00003 import java.io.Serializable;
00004
00005 public class Occupant implements Serializable
00006 {
00007     private int id;
00008     private String nom;
```

```
00009     private String prenom;
00010     private String fonction;
00011
00012     public Occupant()
00013     {
00014         this.nom = "";
00015         this.prenom = "";
00016         this.fonction ="";
00017     }
00018
00019     public Occupant(String nom, String prenom, String fonction)
00020     {
00021         this.nom = nom;
00022         this.prenom = prenom;
00023         this.fonction = fonction;
00024     }
00025
00026     public int getId()
00027     {
00028         return id;
00029     }
00030
00031     public void setId(int id)
00032     {
00033         this.id = id;
00034     }
00035
00042     public String getNom()
00043     {
00044         return this.nom;
00045     }
00046
00053     public void setNom(String nom)
00054     {
00055         this.nom = nom;
00056     }
00057
00064     public String getPrenom()
00065     {
00066         return this.prenom;
00067     }
00068
00075     public void setPrenom(String prenom)
00076     {
00077         this.prenom = prenom;
00078     }
00079
00086     public String getFonction()
00087     {
00088         return this.fonction;
00089     }
00090
00097     public void setFonction(String fonction)
00098     {
00099         this.fonction = fonction;
00100     }
00101
00102 }
```

8.17 Référence du fichier Occupants.java

Classes

— class com.example.groom.Occupants

Paquetages

— package com.example.groom

8.18 Occupants.java

```
00001 package com.example.groom;
00002
00003 import android.content.ContentValues;
00004 import android.content.Context;
00005 import android.database.Cursor;
00006 import android.database.sqlite.SQLiteDatabase;
```

```
00007
00008 import java.util.ArrayList;
00009 import java.util.List;
00010
00011 public class Occupants
00012 {
00013     private GroomBDD groomBDD;
00014     private SQLiteDatabase bdd;
00015
00016     public Occupants(Context context)
00017     {
00018         groomBDD = new GroomBDD(context);
00019         groomBDD.open();
00020         bdd = groomBDD.getBDD();
00021     }
00022
00023     public List<Occupant> getListe()
00024     {
00025         List<Occupant> occupants = new ArrayList<Occupant>();
00026         Cursor cursor = bdd.query("occupants", new String[] {"idOccupant", "nom", "prenom", "fonction"},
00027                 null, null, null, null, null, null);
00028
00029         cursor.moveToFirst();
00030         while (!cursor.isAfterLast())
00031         {
00032             Occupant occupant = cursorToServeur(cursor, false);
00033             occupants.add(occupant);
00034             cursor.moveToNext();
00035         }
00036
00037         cursor.close();
00038         return occupants;
00039     }
00040
00041     public long inserer(String nom, String prenom, String fonction)
00042     {
00043         ContentValues values = new ContentValues();
00044         values.put("nom", nom);
00045         values.put("prenom", prenom);
00046         values.put("fonction", fonction);
00047
00048         return bdd.insert("occupants", null, values);
00049     }
00050
00051     public int modifier(int id, Occupant occupant)
00052     {
00053         ContentValues values = new ContentValues();
00054         values.put("nom", occupant.getNom());
00055         values.put("prenom", occupant.getPrenom());
00056         values.put("fonction", occupant.getFonction());
00057
00058         return bdd.update("occupants", values, "idOccupant = " + id, null);
00059     }
00060
00061     public int supprimer(int id)
00062     {
00063         return bdd.delete("occupants", "idOccupant = " + id, null);
00064     }
00065
00066     public Occupant getOccupant(String nom)
00067     {
00068         Cursor c = bdd.rawQuery("SELECT * FROM occupants WHERE nom = ?", new String[] {nom});
00069
00070         return cursorToServeur(c, true);
00071     }
00072
00073     private Occupant cursorToServeur(Cursor c, boolean one)
00074     {
00075         if (c.getCount() == 0)
00076             return null;
00077
00078         if(one)
00079             c.moveToFirst();
00080
00081         Occupant occupant = new Occupant();
00082         occupant.setId(c.getInt(0));
00083         occupant.setNom(c.getString(1));
00084         occupant.setPrenom(c.getString(2));
00085         occupant.setFonction(c.getString(3));
00086
00087         if (one)
00088             c.close();
00089         return occupant;
00090     }
00091 }
```

8.19 Référence du fichier Preference.java

Classes

— class com.example.groom.Preference

Paquetages

— package com.example.groom

8.20 Preference.java

```
00001 package com.example.groom;
00002
00003 public class Preference
00004 {
00005     private int idPreferences;
00006     private String appareilGroom;
00007     private int idPrecedentOccupant;
00008
00009     public Preference()
00010     {
00011         appareilGroom = "";
00012         idPrecedentOccupant = 0;
00013     }
00014
00015     public int getIdPreferences()
00016     {
00017         return this.idPreferences;
00018     }
00019
00020     public void setIdPreferences(int idPreferences)
00021     {
00022         this.idPreferences = idPreferences;
00023     }
00024
00025     public String getAppareilGroom()
00026     {
00027         return this.appareilGroom;
00028     }
00029
00030     public void setAppareilGroom(String appareilGroom)
00031     {
00032         this.appareilGroom = appareilGroom;
00033     }
00034
00035     public int getIdPrecedentOccupant()
00036     {
00037         return this.idPrecedentOccupant;
00038     }
00039
00040     public void setIdPrecedentOccupant(int idPrecedentOccupant)
00041     {
00042         this.idPrecedentOccupant = idPrecedentOccupant;
00043     }
00044 }
```

8.21 Référence du fichier Preferences.java

Classes

— class com.example.groom.Preferences

Paquetages

— package com.example.groom

8.22 Preferences.java

```
00001 package com.example.groom;
00002
00003 import android.content.ContentValues;
00004 import android.content.Context;
00005 import android.database.Cursor;
00006 import android.database.sqlite.SQLiteDatabase;
00007
00008 public class Preferences
00009 {
00010     private GroomBDD groomBDD;
00011     private SQLiteDatabase bdd;
00012
00013     public Preferences(Context context)
00014     {
00015         groomBDD = new GroomBDD(context);
00016         groomBDD.open();
00017         bdd = groomBDD.getBDD();
00018     }
00019
00020     public Preference getPreference()
00021     {
00022        Preference preference;
00023        Cursor cursor = bdd.query("preferences", new String[] {"idPreferences", "groom", "
    idPrecedentOccupant"},
00024                 null, null, null, null, null, null);
00025
00026         cursor.moveToFirst();
00027         preference = cursorToServeur(cursor, false);
00028
00029         cursor.close();
00030         return preference;
00031     }
00032
00033     public long inserer(String appareilGroom, int idPrecedentOccupant)
00034     {
00035         ContentValues values = new ContentValues();
00036         values.put("groom", appareilGroom);
00037         values.put("idPrecedentOccupant", idPrecedentOccupant);
00038
00039         return bdd.insert("preferences", null, values);
00040     }
00041
00042     public int modifier(int id, String appareilGroom, int idPrecedentOccupant)
00043     {
00044         ContentValues values = new ContentValues();
00045         values.put("groom", appareilGroom);
00046         values.put("idPrecedentOccupant", idPrecedentOccupant);
00047
00048         return bdd.update("preferences", values, "idPreferences = " + id, null);
00049     }
00050
00051     public int supprimer(int id)
00052     {
00053         return bdd.delete("preferences", "idOccupant = " + id, null);
00054     }
00055
00056     public boolean estVide()
00057     {
00058         Cursor c = bdd.rawQuery("SELECT * FROM preferences", null);
00059
00060         if(c.getCount() == 0)
00061         {
00062             return true;
00063         }
00064         else
00065         {
00066             return false;
00067         }
00068     }
00069
00070     private Preference cursorToServeur(Cursor c, boolean one)
00071     {
00072         if (c.getCount() == 0)
00073             return null;
00074
00075         if(one)
00076             c.moveToFirst();
00077
00078       Preference preference = new Preference();
00079       preference.setIdPreferences(c.getInt(0));
00080       preference.setAppareilGroom(c.getString(1));
00081       preference.setIdPrecedentOccupant(c.getInt(2));
00082
00083       if (one)
00084           c.close();
00085       return preference;
00086     }
00087 }
```

8.23 Référence du fichier README.md

8.24 README.md

```
00001 \mainpage Le projet
00002
00003 \tableofcontents
00004
00005 Le portier connecté « grOOm » permettra à l'occupant du bureau de communiquer sa disponibilité avec
       des personnes extérieures (visiteurs). Pour cela, il utilise une application soit en version PC soit mobile.
00006
00007 \section section_tdm Table des matières
00008 - \ref page_README
00009 - \ref page_changelog
00010 - \ref page_about
00011 - \ref page_licence
00012
00013 \section section_infos Informations
00014
00015 \author Grégory Eyraud <<gregory.eyraud@gmail.com>>
00016 \date 2020
00017 \version 0.2
00018 \see https://svn.riouxsvn.com/groom/
00019
00020
00021 \page page_README README
00022
00023 [TOC]
00024
00025 # Projet {#projet}
00026
00027 ## Présentation {#presentation}
00028
00029 Le portier connecté « grOOm » permettra à l'occupant du bureau de communiquer sa disponibilité avec
       des personnes extérieures (visiteurs). Pour cela, il utilise une application soit en version PC soit mobile.
00030
00031 Tout en s'intégrant facilement à l'environnement, il résout le manque d'interface entre les
       utilisateurs et les bureaux permettant de travailler plus efficacement.
00032
00033 A partir de l'application, l'occupant informe le visiteur de son état : "Libre", "Occupé" ou "Absent".
       Il aura la possibilité d'ajouter un message "libre" qui s'affichera alors sur l'écran du portier. S'il le
       souhaite, il informera le visiteur que celui-ci peut "Entrer" (ou "Entrez"). L'occupant dispose d'un mode
       "Sonnette" qu'il peut activer sur le portier connecté. Dans ce mode, le visiteur pourra sonner via l'écran
       tactile et prévenir l'occupant de sa présence.
00034
00035 Version : Mobile Android
00036
00037 ## Informations {#informations}
00038
00039 \author Grégory Eyraud <gregory.eyraud@gmail.com>
00040 \date 2020
00041 \version 0.2
00042 \see https://svn.riouxsvn.com/groom/
00043
00044
00045 \page page_about A propos
00046
00047 \author Grégory Eyraud <gregory.eyraud@gmail.com>
00048 \date 2020
00049 \version 0.2
00050 \see https://svn.riouxsvn.com/groom/
00051
00052
00053 \page page_licence Licence GPL
00054
00055 This program is free software; you can redistribute it and/or modify
00056 it under the terms of the GNU General Public License as published by
00057 the Free Software Foundation; either version 2 of the License, or
00058 (at your option) any later version.
00059
00060 This program is distributed in the hope that it will be useful,
00061 but WITHOUT ANY WARRANTY; without even the implied warranty of
00062 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00063 GNU General Public License for more details.
00064
00065 You should have received a copy of the GNU General Public License
00066 along with this program; if not, write to the Free Software
00067 Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```