Aix-Marseille

Académie
Session 2021

Dossier technique - Revue finale

Projet groom

version 1.0

MOTA Yuri *ROUGIER Alexander*

BTS SNIR LaSalle Avignon 2021

1. Introduction

groom est un système de gestion automatisé d'un portier connecté qui :

- permettra d'accéder en temps réel aux informations de l'espace concerné.

- permettra à l'occupant du bureau de communiquer sa disponibilité avec des personnes extérieures (visiteurs).
  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.

Le projet est composé de différents modules et matériels qui sont :

Côté électronique :

- Un écran LCD Shield (tactile)
- Une carte électronique type Arduino UNO
- Une led rouge et verte (CMS)
- Un module PIR avec capteur infrarouge Murata.

Côté informatique :

- Un programme fonctionnel (sur pc ou terminal mobile) permettant à l'occupant :

  de se connecter au portier via une liaison bluetooth

  de gérer son profil (Nom, prénom, fonction).

  de donner son état de présence et d'occupation (libre, occupé, absent).

  d'informer le visiteur de la possibilité d'entrer dans la pièce.

  d'ajouter un message libre qui s'affiche alors sur l'écran du portier.

  d'activer ou désactiver la possibilité au visiteur de notifier de sa présence via l'écran tactile ("sonnette").

  d'être informé de façon temporaire (notification) de la présence d'un visiteur (qui a "sonné" ou détecté par une présence prolongée).

  de gérer un calendrier (format iCalendar).

2. Présentation générale

2.1 Expression de besoin

Les interactions dans les bureaux entre le visiteur et l'occupant du bureau ne sont pas toujours faciles. Ce projet a donc pour but de faciliter l'interaction entre les deux acteurs du projet :

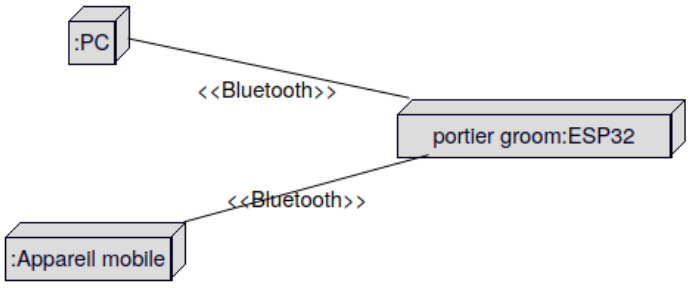
- L'occupant qui pourra communiquer sa présence ou non dans le bureau et indiquer au visiteur s'il faut entrer ou non.
- Le visiteur qui pourra connaître la disponibilité de l'occupant et signaler sa présence.

2.2 Présentation du projet

L'objectif principal est de proposer une solution simple, alliant flexibilité, ergonomie et économie. Les “afficheurs connectés” répondent à ces problématiques en donnant de la visibilité sur la disponibilité d’accès à un bureau.

2.3 Architecture du système

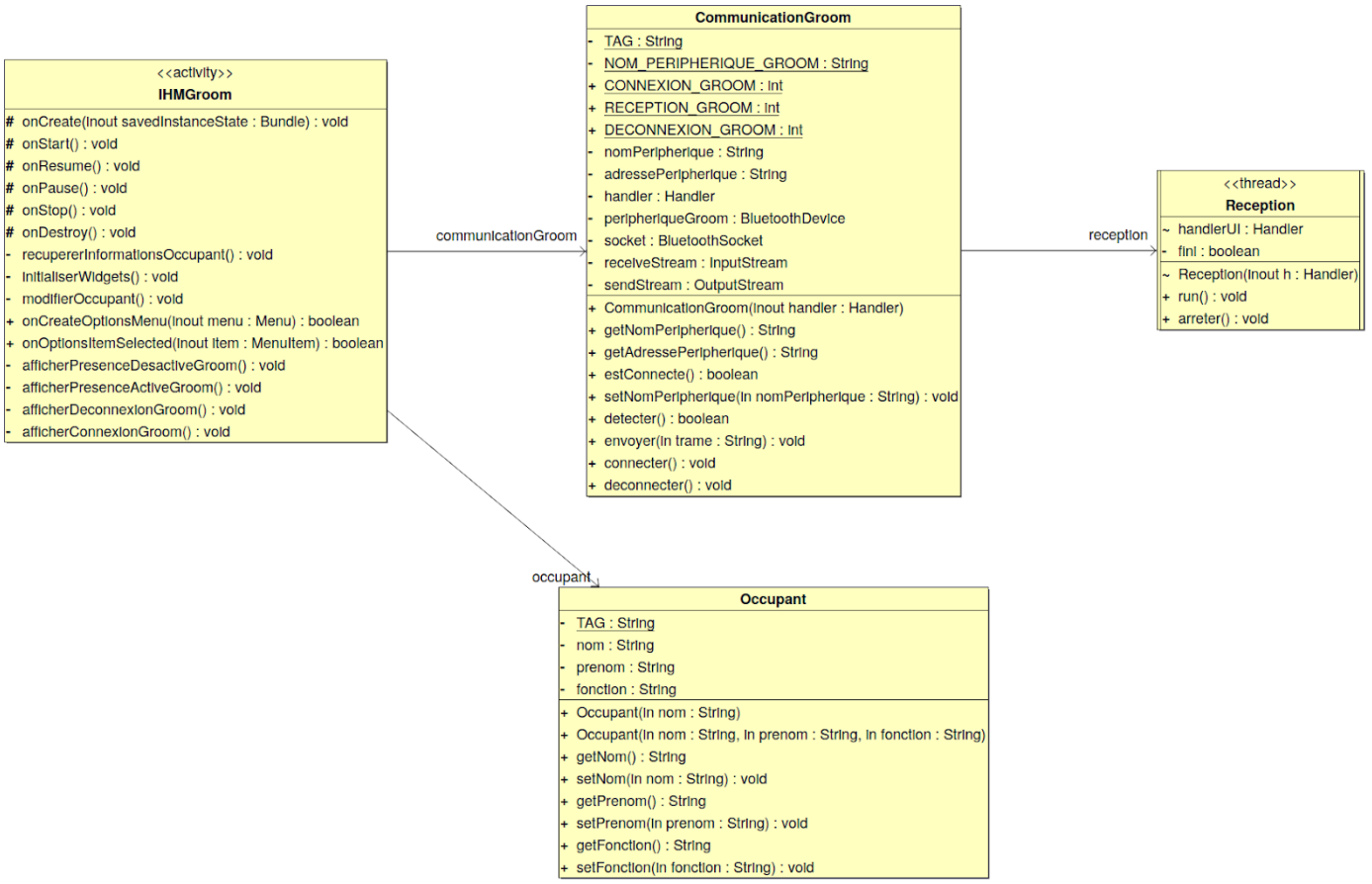
2.4 Diagramme de déploiement



2.5 Diagramme de classes du projet

2.5.1 Diagramme de l'application PC

2.5.2 diagramme de classe de l'application mobile



2.6 Choix du type de communication

Liaison	ESP32
Distance	10 mètres
Débit max	115200 Bauds
Nombre récepteurs	12

Le protocole de communication GROOM est basé sur des trames requêtes/réponses via une liaison Bluetooth.

La fréquence d'émission est de 2,4 - 2.5 GHz.

La puissance d'émission est de 2,5 mW (-97dBm) pour une portée de 10m d'après les exigences de l'utilisateur / occupant qui doit
pouvoir contrôler le portier dans la pièce.

La version (norme) utilisée est la v4,2.

2.7 Répartition des tâches en IR

Etudiant IR 1 : Yuri MOTA
- Programmation du logiciel sur PC (Qt).

Etudiant IR 2 : Alexander ROUGIER
- Programmation de l'application mobile (Android).

2.8 Objectifs attendues

- Pouvoir se connecter en Bluetooth a la carte ESP32 pour communiquer avec l'écran
- Réaliser deux IHM permettant à un ou plusieurs occupants de la pièce de gérer un portier connecté via une liaison bluetooth.
- Gestion de l'état de l'occupant
- Gestion de la sonnette (accès à l'écran pour le visiteur)
- Gestion de l'affichage des informations de l'occupant choisi sur l'écran.
- Possibilité d'afficher la mention "entrer"
- Gestion d'un iCalendar (Calendrier) pour chaque occupant enregistré avec possibilités d'afficher des tâches et autres évènements et pouvoir gérer l'état de l'occupant automatiquement en fonction de son calendrier.

2.9 Protocole de communication

Protocole de communication Groom (version 0.9c)

Trame Groom

La trame est composée de caractères ASCII. Le délimiteur de champ est le '**;**'. Le délimiteur de début est le '**$**' suivi de **GROOM** qui indique le nom du protocole. Le délimiteur de fin de trame est "**\r\n**".

Sens : Groom → PC/Tablette

Format :

$GROOM;ETAT;SONNETTE;PRESENCE;MODE_SONNETTE;MODE_PRESENCE\r\n

Le champ **ETAT** peut prendre 3 valeurs différentes :

- Libre **0**
- Absent **1**
- Occupé **2**

Le champ **SONNETTE** précise si un appel a été réalisé.

Le champ **PRESENCE** aura la valeur 1 dans le cas où une personne a été détectée devant le groom.

Le champ **MODE_SONNETTE** précise l'état d'activation **1** (activé) ou **0** (désactivé) de la sonnette.

Le champ **MODE_PRESENCE** précise l'état d'activation **1** (activé) ou **0** (désactivé) du détecteur de présence.

Cette trame est envoyée par le Groom lorsque celui-ci a détecté un changement d'état (**ETAT**, **SONNETTE** et/ou **PRESENCE**) ou de mode (**MODE_SONNETTE/MODE_PRESENCE**).

Exemple :

$GROOM;0;1;1;1;1\r\n

L'accès au bureau pour les visiteurs est possible (**ETAT=0** donc LIBRE). Le visiteur a "sonné" (**SONNETTE=1**) et il a été détecté (**PRESENCE=1**).

Trame Commande

Sens : PC/Tablette → Groom

Format :

`$CMD;ORDRE;SONNETTE;PRESENCE\r\n`

Le champ **ORDRE** peut prendre 4 valeurs différentes :

- Libre **0**
- Absent **1**
- Occupé **2**
- Entrez **3**

Le champ **SONNETTE** précise l'état d'activation **1** ou **0** (désactivé) de la sonnette.

Le champ **PRESENCE** précise l'état d'activation **1** ou **0** (désactivé) du détecteur de présence. Ceci est utile lorsque le Groom est placé dans une zone où de nombreux passages existent (toilette, couloir, …)

Exemple :

`$CMD;1;0;0\r\n`

L'accès au bureau pour les visiteurs est possible (**`ORDRE=1`** donc ABSENT). Le visiteur ne sera pas détecté (**`PRESENCE=0`** donc le détecteur de présence est désactivé) et ne pourra pas "sonner" (**`SONNETTE=0`** donc désactivée).

Trame Etat

Sens : PC/Tablette → Groom

Format :

$ETAT\r\n

La trame **ETAT** est une trame de requête permettant d'obtenir l'état courant du GROOM. Celui-ci renvoie alors une trame **GROOM**.

Trame de service (application vers système) :

L'application envoie périodiquement (toutes les secondes) une trame ALIVE pour maintenir la connexion ouverte.

Décomposition d'une trame :

$GROOM ; **A** \r\n

$GROOM ; **A**live \r\n

Remarque : la trame ne contient aucun espace

Le système répondra par une trame d'acquittement.

Décomposition d'une trame :

$GROOM ; **A** \r\n

$iotruck ; **A**ck \r\n

Remarque : la trame ne contient aucun espace

Trame Affichage

La trame est composée de caractères ASCII. Le délimiteur de champ est le '**;**'. Le délimiteur de fin de trame est "**\r\n**".

Sens : PC/Tablette ➔ Groom

Format :

$AFFICHAGE;NOM;PRENOM;FONCTION\r\n

La trame **AFFICHAGE** transportera le **NOM**, le **PRENOM** et la **FONCTION** de la personne qui occupe le bureau.

Ces différents champs contiennent des chaînes de caractères qui seront utilisées directement pour l'affichage sur le groom.

Exemple :

$AFFICHAGE;COPIN;Olivier;DDFPT\r\n

La personne qui occupe le bureau est **Olivier COPIN** dont la fonction est **DDFPT** (Directeur Délégué aux Formations Professionnelles et Technologiques).

Trame MsgPerso

La trame est composée de caractères ASCII. Le délimiteur de champ est le '**;**'. Le délimiteur de fin de trame est "**\r\n**".

Sens : PC/Tablette ➔ Groom

Format :

$MSGPERSO;messageperso\r\n

Cette trame sera envoyé que lorsque l'utilisateur du bureau voudra afficher un message personnalisé court.

Exemple :

$MSGPERSO;Je reviens dans 10 minutes\r\n

Le message **Je reviens dans 10 minutes** sera affiché sur l'écran du groom.

2.10 Informations

Auteur

Yuri Mota <motayuri13@gmail.com>
Alexander Rougier <alexander.rougier@gmail.com>

Date

2021

Version

0.1

Dépôt du code source

https://svn.riouxsvn.com/groom-2021

3. Yuri MOTA

3.1 Présentation personnelle

Application PC
- Informer le visiteur
- Gérer le mode Sonnette / mode présence
- Dialoguer avec le portier connecté
- Importer un calendrier iCalendar et visualiser les événements de la journée
- Afficher les notifications

3.1.1 Rappel du besoin initial

- Gérer un portier connecté via un logiciel programmé sous Qt (avec l'IDE Qt Creator et Qt Designer).

3.1.2 Organisation

Pour l'organisation du projet et la communication entre tous les membres du groupe, les outils suivants seront utilisés :
- **Subversion** qui est un logiciel libre de gestion de versions hébergé sur le site RiouxSVN pour l'ensemble du code source du projet.
- L'espace de stockage commun **Google Drive** pour tous les documents ressources.
- **Beesbusy** qui permet de gérer la planification des tâches dans le projet.

3.1.3 Objectifs

Communiquer avec le portier via le logiciel pc.

3.1.4 Répartition des tâches (globales)

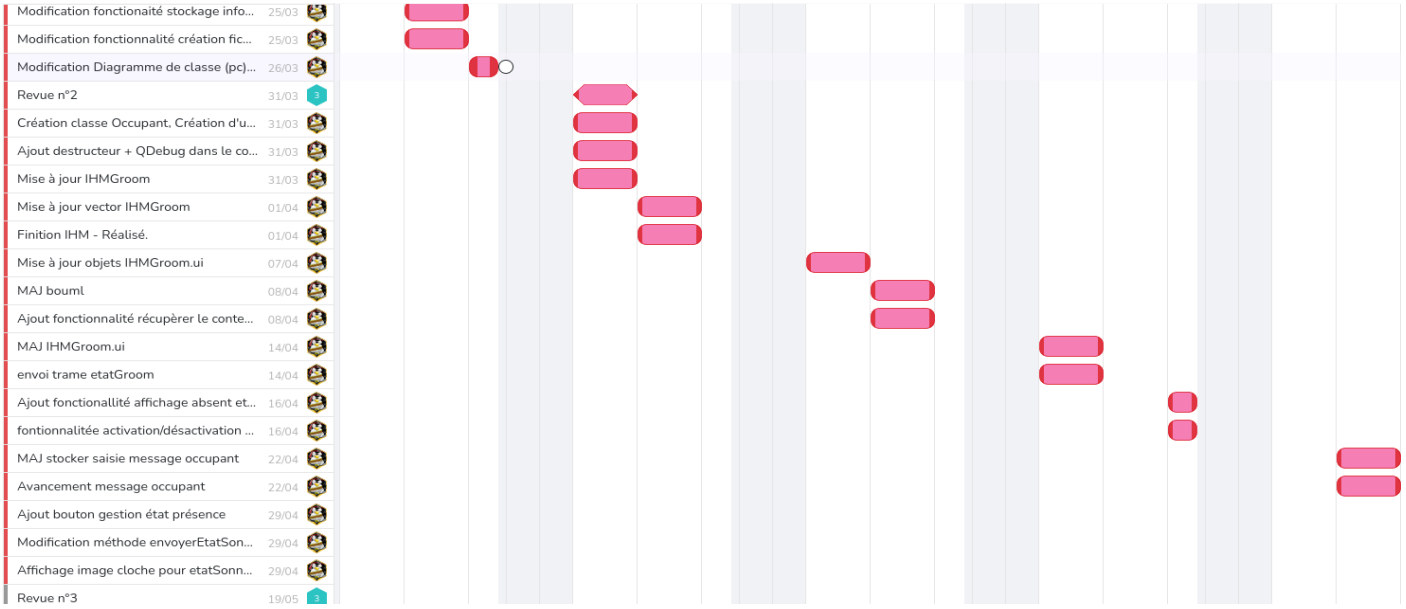
Identification par priorités

Fonctionnalités	Priorité
Définir une IHM (Qt / Android Studio)	Haute
Mettre en place une liaison sans fil (Bluetooth) (Qt / Android Studio)	Haute
Spécifier l'identité de l'occupant du bureau affiché sur un écran tactile	Moyenne
Gérer la présence d'un visiteur	Haute
Gérer un mode sonnette	Haute
Gérer l'affichage des événements	Moyenne
Informer le visiteur de son état : "Libre", "Occupé" ou "Absent"	Haute
Possibilité d'ajouter un message libre qui s'affichera alors sur l'écran du portier	Basse
Informer le visiteur que celui-ci peut "Entrer"	Haute
Chargement et stockage des paramètres en local dans un fichier INI (ou base de données)	Basse

Planification par itérations

Fonctionnalités	Itération
Définir une IHM (Qt / Android Studio)	1
Mettre en place une liaison sans fil (Bluetooth) (Qt / Android Studio)	2
Spécifier l'identité de l'occupant du bureau affiché sur un écran tactile	3
Gérer la présence d'un visiteur	2
Gérer un mode sonnette	2
Gérer l'affichage des événements	3
Informer le visiteur de son état : "Libre", "Occupé" ou "Absent"	2
Possibilité d'ajouter un message libre qui s'affichera alors sur l'écran du portier	3
Informer le visiteur que celui-ci peut "Entrer"	2
Chargement et stockage des paramètres en local dans un fichier INI (ou base de données)	3

Planification d'exemple (revue 2 vers la revue 3)

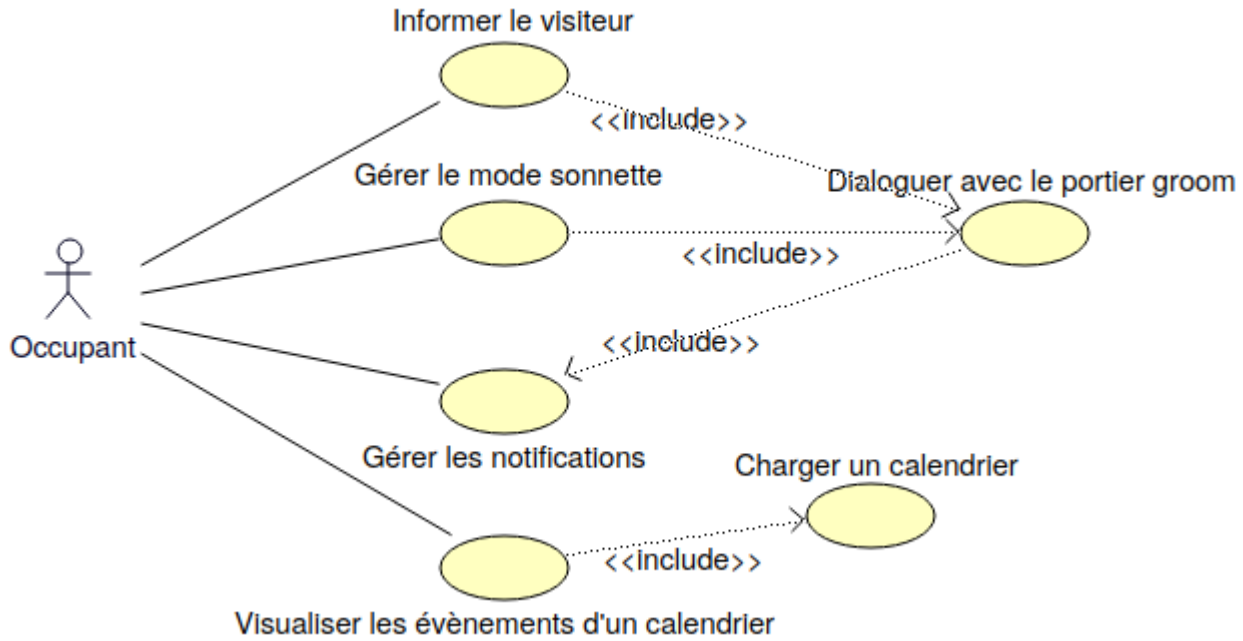


3.1.5 Outils de développement

Désignation	*Caractéristiques*
OS Poste de développement	PC sous GNU/Linux Ubuntu
EDI	Qt Creator et Qt Designer
API GUI	Qt 5.11.2
Atelier de génie logiciel	Bouml v7.11

L'appareil utilisé est un pc utilisant la norme Bluetooth 4.2.

3.1.6 Diagramme de cas d'utilisation



L'acteur occupant est associé avec 4 fonctionnalités :

- Informer le visiteur
- Gérer le mode sonnette
- Gérer les notifications
- Visualiser les événements d'un calendrier

Pour informer le visiteur, il faut dialoguer avec le portier groom,
Pour gérer le mode sonnette, il faut dialoguer avec le portier groom,
Pour dialoguer avec le portier groom, il faut gérer les notifications,
Pour visualiser les évènements, il faut charger un calendrier.

3.1.7 Le protocole de communication

Le protocole de communication groom est basé sur des trames requêtes/réponses via une liaison bluetooth.

Protocole de communication Groom (version 0.9c)

Trame Groom

La trame est composée de caractères ASCII. Le délimiteur de champ est le **';'**. Le délimiteur de début est le **'$'** suivi de **GROOM** qui indique le nom du protocole. Le délimiteur de fin de trame est "**\r\n**".

Sens : Groom → PC/Tablette

Format :

$GROOM;ETAT;SONNETTE;PRESENCE;MODE_SONNETTE;MODE_PRESENCE\r\n

Le champ **ETAT** peut prendre 3 valeurs différentes :

- Libre **0**
- Absent **1**
- Occupé **2**

Le champ **SONNETTE** précise si un appel a été réalisé.

Le champ **PRESENCE** aura la valeur 1 dans le cas où une personne a été détectée devant le groom.

Le champ **MODE_SONNETTE** précise l'état d'activation **1** (activé) ou **0** (désactivé) de la sonnette.

Le champ **MODE_PRESENCE** précise l'état d'activation **1** (activé) ou **0** (désactivé) du détecteur de présence.

Cette trame est envoyée par le Groom lorsque celui-ci a détecté un changement d'état (**ETAT, SONNETTE** et/ou **PRESENCE**) ou de mode (**MODE_SONNETTE/MODE_PRESENCE**).

Exemple :

$GROOM;0;1;1;1;1\r\n

L'accès au bureau pour les visiteurs est possible (**ETAT=0** donc LIBRE). Le visiteur a "sonné" (**SONNETTE=1**) et il a été détecté (**PRESENCE=1**).

Trame Commande

Sens : PC/Tablette → Groom

Format :

$CMD;ORDRE;SONNETTE;PRESENCE\r\n

Le champ **ORDRE** peut prendre 4 valeurs différentes :

- Libre **0**
- Absent **1**
- Occupé **2**
- Entrez **3**

Le champ **SONNETTE** précise l'état d'activation **1** ou **0** (désactivé) de la sonnette.

Le champ **PRESENCE** précise l'état d'activation **1** ou **0** (désactivé) du détecteur de présence. Ceci est utile lorsque le Groom est placé dans une zone où de nombreux passages existent (toilette, couloir, …)

Exemple :

$CMD;1;0;0\r\n
L'accès au bureau pour les visiteurs est possible (**ORDRE=1** donc ABSENT). Le visiteur ne sera pas détecté (**PRESENCE=0** donc le détecteur de présence est désactivé) et ne pourra pas "sonner" (**SONNETTE=0** donc désactivée).

Trame Etat

Sens : PC/Tablette → Groom

Format :

$ETAT\r\n

La trame **ETAT** est une trame de requête permettant d'obtenir l'état courant du GROOM. Celui-ci renvoie alors une trame **GROOM.**

Trame de service (application vers système) :

L'application envoie périodiquement (toutes les secondes) une trame ALIVE pour maintenir la connexion ouverte.

Décomposition d'une trame :

$GROOM ; **A** \r\n

$GROOM ; **A**live \r\n

Remarque : la trame ne contient aucun espace

Le système répondra par une trame d'acquittement.

Décomposition d'une trame :

$GROOM ; **A** \r\n

$iotruck ; **A**ck \r\n

Remarque : la trame ne contient aucun espace

Trame Affichage

La trame est composée de caractères ASCII. Le délimiteur de champ est le **';'**. Le délimiteur de fin de trame est "**\r\n**".

Sens : PC/Tablette → Groom

Format :

$AFFICHAGE;NOM;PRENOM;FONCTION\r\n

La trame **AFFICHAGE** transportera le **NOM,** le **PRENOM** et la **FONCTION** de la personne qui occupe le bureau.

Ces différents champs contiennent des chaînes de caractères qui seront utilisées directement pour l'affichage sur le groom.

Exemple :

$AFFICHAGE;COPIN;Olivier;DDFPT\r\n

La personne qui occupe le bureau est **Olivier COPIN** dont la fonction est **DDFPT** (Directeur Délégué aux Formations Professionnelles et Technologiques).

Trame MsgPerso

La trame est composée de caractères ASCII. Le délimiteur de champ est le ';'. Le délimiteur de fin de trame est "**\r\n**".

Sens : PC/Tablette → Groom

Format :

$MSGPERSO;messageperso\r\n

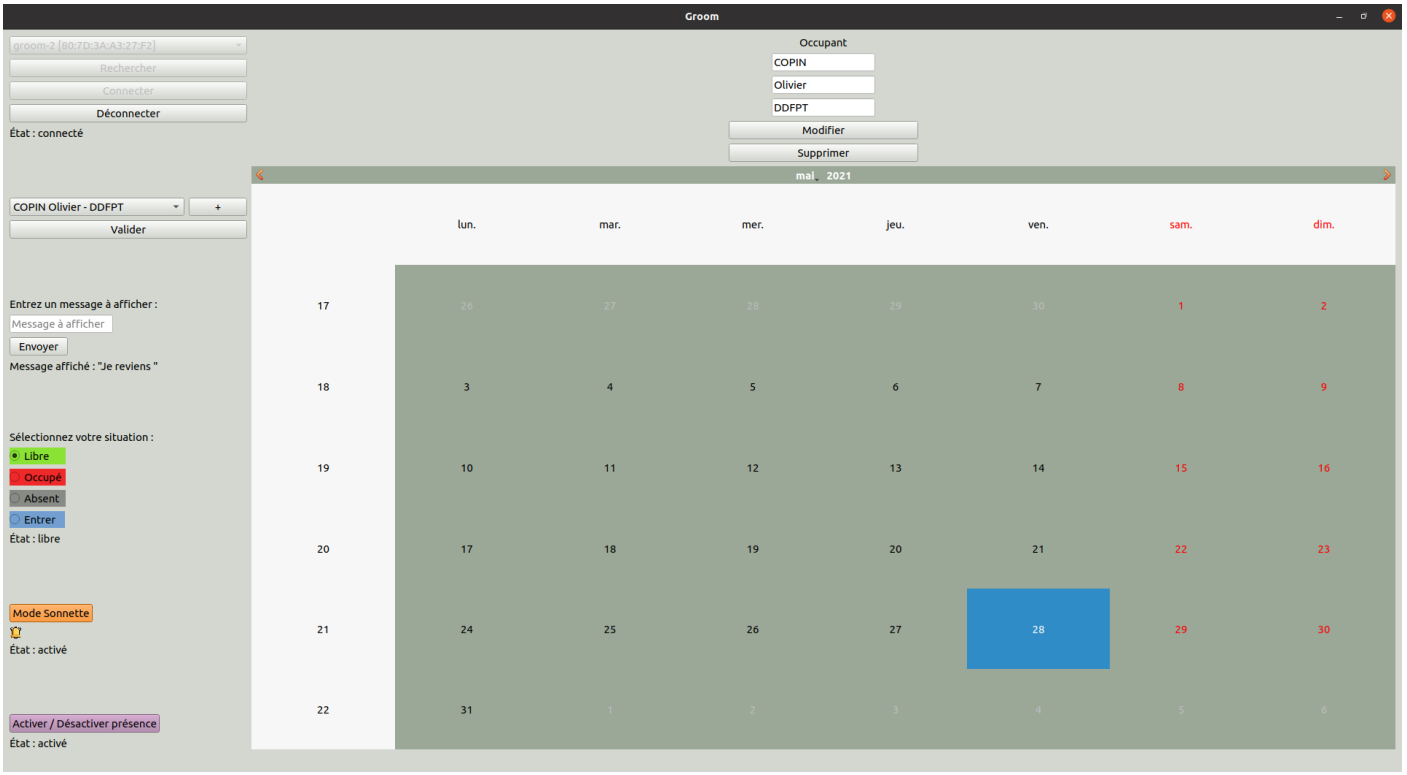
Cette trame sera envoyé que lorsque l'utilisateur du bureau voudra afficher un message personnalisé court.

Exemple :

$MSGPERSO;Je reviens dans 10 minutes\r\n

Le message **Je reviens dans 10 minutes** sera affiché sur l'écran du groom.

3.1.8 IHM



- On peut comme le cahier des charges nous invite à le faire rechercher le module groom, se connecter/se déconnecter. Un acquittement "connecté" ou "déconnecté" est affiché en conséquence.

- On peut choisir un occupant avec la liste déroulante,le modifier avec les zones de saisie et le bouton modifier ou bien le supprimer ainsi qu'en ajouter un nouveau une fois saisie en validant avec le bouton comportant le symbole "+".

- On peut saisir un message libre dans la zone de saisie correspondante et l'envoyer pour affichage sur le portier. Un affichage du message envoyé est affiché ou non en dessous en fonction du message envoyé.

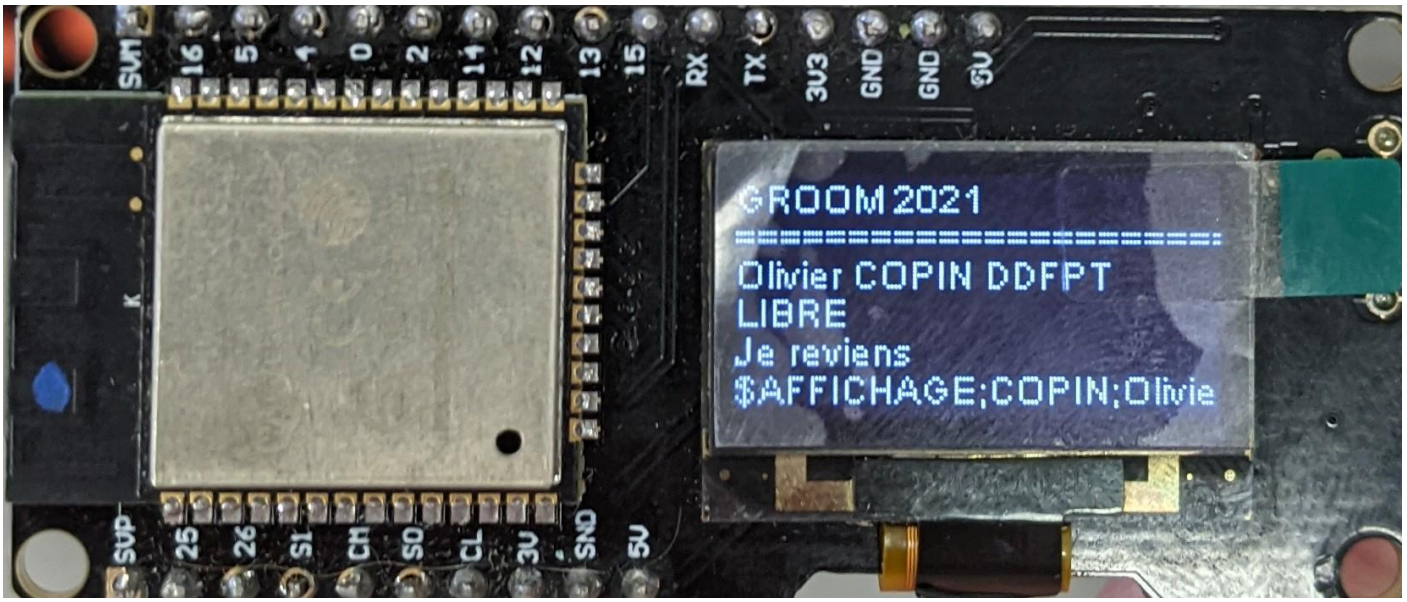
- L'état de l'occupant peut être choisi avec les boutons radio "libre, occupé, absent" (affichage sur le portier) et l'occupant peut choisir le bouton "entrer" permettant d'indiquer à un visiteur d'entrer.

- Un mode sonnette / mode présence est activable/désactivable via les deux boutons en bas à gauche de l'IHM. Un cloche indiquant l'état de la sonnette ainsi qu'un acquittement pour l'un et pour l'autre de la part du portier(sous forme de deux labels).

- Si et seulement si l'état "libre" est choisi, le bouton "entrer" sera cliquable et le mode sonnette et présence activable de même (sinon désactivés si changement d'état).

- Une notification brève peut apparaître si le mode sonnette ou présence est activé. (Un bouton sur le simulateur a permis de tester la fonctionnalité sonnette). Le simulateur n'ayant pas de capteur de présence, la fonctionnalité à été codée tout comme le mode sonnette mais sans vérification réelle n'est pas intégré.

Résultat sur simulateur :

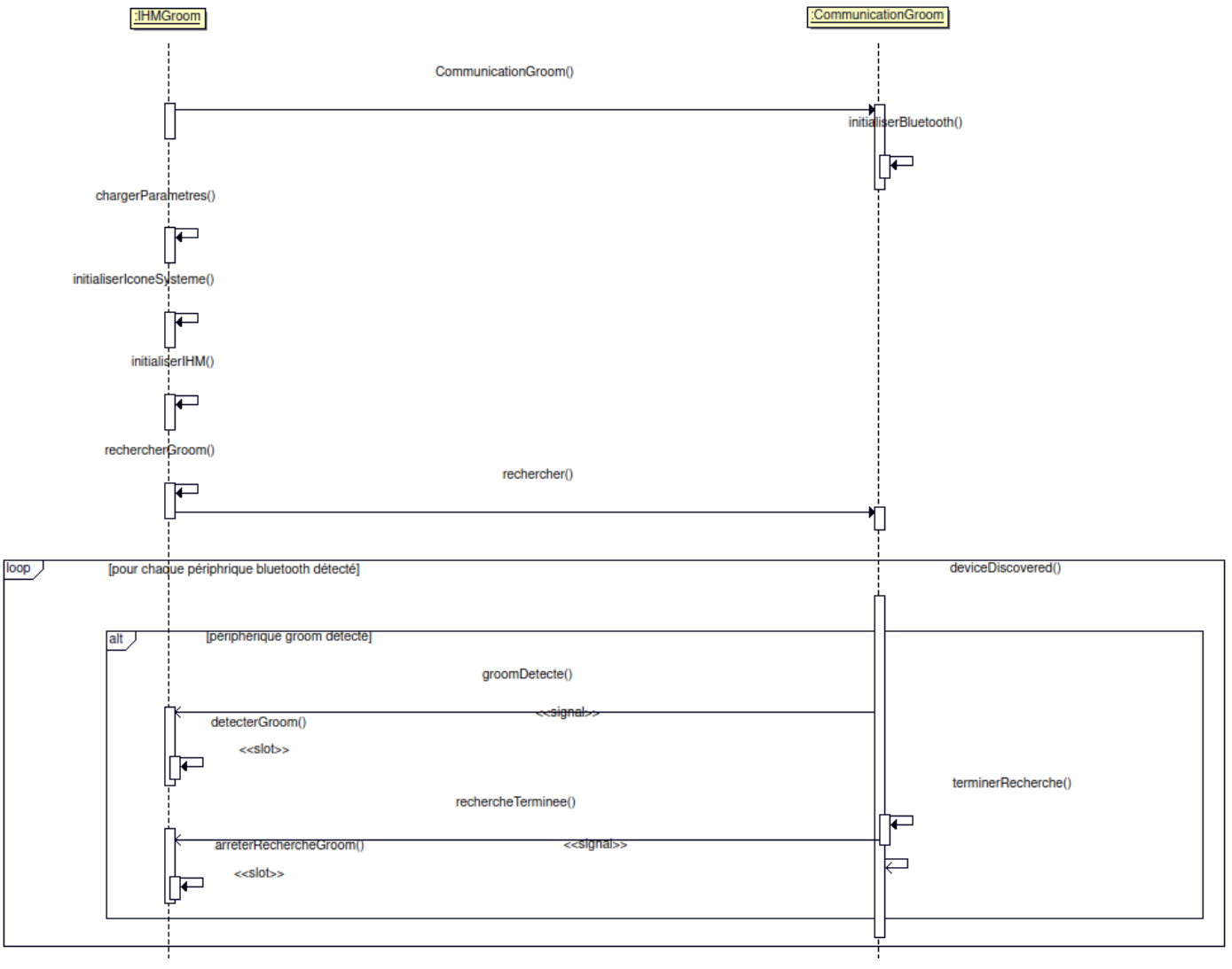


3.1.9 Diagramme de classes

3.1.10 Diagramme de séquence du démarrage du logiciel

Le diagramme de séquence ci-contre montre les interactions entre les classes IHMGroom et CommunicationGroom lors du démarrage du logiciel :

- La méthode chargerParametres() de IHMGroom permet le stockage des données des occupants (nom, prénom, fonction) dans un fichier de stockage en .INI.
- La méthode initialiserIconeSysteme() de IHMGroom permet l'initialisation de l'icône système groom.png.
- La méthode initialiserIHM() de IHMGroom permet toutes les interactions lors du démarrage de l'application et de l'affichage de l'IHM (soit les connexions signaux/slots, les affectations d'états des objets graphiques de l'IHM.
- La méthode rechercheGroom() IHMGroom permet la recherche de périphériques bluetooth.
- Pour chaque périphérique bluetooth détecté la condition suivante est réalisée : Soit que si le périphérique groom est détecté le slot detecterGroom() de IHMGroom est appelé et permet de confirmer avoir trouvé le bon périphérique et un signal est envoyé par la méthode groomDetecte().
- Quand la recherche est terminée par le signal rechercheTerminee() le slot arreterRechercheGroom() de IHMGroom est appelé pour arrêter la recherche et mettre fin à la communication avec la méthode terminerRecherche() de CommunicationGroom.



3.1.11 Tests de validation

Tests	Application	Validation
Voir si la connexion Bluetooth est fonctionnelle et automatique et contrôler si on peut changer la connexion.	Test graphique	
Entrer nom et travail / contrôler si à chaque démarrage de l'application le nom et le statut du particulier s'affichent sur l'application.	Test graphique	
Pour envoyer des messages, contrôler si on peut les écrire et afficher sur l'écran LCD du GROOM.	Test graphique	
Contrôler si on a des messages pré-enregistrés pour faciliter l'envoi des messages.	Débogage	

3.1.12 Recette

Fonctionnalités réalisées :

- Connexion bluetooth
- activation/désactivation capteur de présence
- entrer le nom et la fonction de l'occupant
- envoyer le nom et la fonction de l'occupant via le bluetooth
- entrer un message personnelle de l'occupant
- envoyer le message personnelle de l'occupant
- envoyer les états de l'occupant
- envoyer l'ordre d'entrer
- envoyer l'autorisation d'utiliser la sonnette

3.1.13 Documentation (diagramme de séquence)

Liaison au diagramme de séquence de démarrage.
(Issu de la documentation Doxygen du fichier Doxyfile dans le dépôt svn)

3.1.13.1 Classe IHMGroom

a. Documentation de la classe IHMGroom

Déclaration de la classe IHMGroom.

Cette classe s'occupe de l'affichage de l'IHM du Groom

b. Documentation des fonctions membres

◆ arreterRechercheGroom
void IHMGroom::arreterRechercheGroom ( )
slot
Définition à la ligne 560 du fichier IHMGroom.cpp.

```
561 {
562   uiIHMGroom->listeBluetooth->setEnabled(true);
563   uiIHMGroom->boutonRechercherBluetooth->setEnabled(true);
564   if(uiIHMGroom->listeBluetooth->count() > 0 )
565   uiIHMGroom->boutonConnecterBluetooth->setEnabled(true);
566   uiIHMGroom->boutonDeconnecterBluetooth->setEnabled(false);
567   uiIHMGroom->labelEtatConnexion->setText("État : déconnecté");
568 }
```

Références uiIHMGroom.

Référencé par initialiserIHM().

◆ chargerParametres()
IHMGroom::chargerParametres ( )
private
Charge les paramètres dans le fichier .ini de l'application Groom.

Charge les paramètres de l'application.

Définition à la ligne 493 du fichier IHMGroom.cpp.

```
494 {
495   // Fichier de stockage des noms, prénoms et fonctions des occupants de la pièce
496   QString nomFichier = QApplication::applicationDirPath() + "/parametres.ini";
497   qDebug() << Q_FUNC_INFO << nomFichier;
498   QSettings settings(nomFichier, QSettings::IniFormat);
499
```

```
500   int nbOccupants = settings.value("NbOccupants", 0).toInt();
501   qDebug() << Q_FUNC_INFO << "NbOccupants" << nbOccupants;
502   indexOccupant = settings.value("Occupant", 0).toInt();
503   qDebug() << Q_FUNC_INFO << "Occupant courant" << indexOccupant;
504
505   uiIHMGroom->listeOccupant->clear();
506   for(int i = 0; i < nbOccupants; i++)
507   {
508         QString nomOccupant = "Occupant" + QString::number(i+1);
509         settings.beginGroup(nomOccupant);
510         qDebug() << Q_FUNC_INFO << nomOccupant <<
settings.value("Nom").toString() << settings.value("Prenom").toString() <<
settings.value("Fonction").toString();
511         Occupant occupant;
512         occupant.nom = settings.value("Nom").toString();
513         occupant.prenom = settings.value("Prenom").toString();
514         occupant.fonction = settings.value("Fonction").toString();
515         occupants.push_back(occupant);
516         settings.endGroup();
517         uiIHMGroom->listeOccupant->addItem(occupant.nom + " " + occupant.prenom +
" - " + occupant.fonction);
518   }
519   uiIHMGroom->listeOccupant->setCurrentIndex(indexOccupant-1);
520 }
```

Références Occupant::fonction, indexOccupant, Occupant::nom, occupants, Occupant::prenom, et uiIHMGroom.

Référencé par IHMGroom().

◆ detecterGroom

void IHMGroom::detecterGroom (QString nomPeripherique, QString adressePeripherique)
slot
Méthode déclenchée lorsque le module Bluetooth GROOM est détectée.

Paramètres
nomPeripherique Le nom du module Bluetooth GROOM
adressePeripherique L'adresse MAC du module Bluetooth GROOM
Définition à la ligne 292 du fichier IHMGroom.cpp.

```
293 {
294   QString module = nomPeripherique + " [" + adressePeripherique + "]";
295   qDebug() << Q_FUNC_INFO << module;
296
297   // déjà présent dans la liste ?
298   if(uiIHMGroom->listeBluetooth->findText(module) == -1)
299   {
300         uiIHMGroom->listeBluetooth->addItem(module);
```

```
301         uiIHMGroom->boutonConnecterBluetooth->setEnabled(true);
302     }
303 }
```

Références uiIHMGroom.

Référencé par initialiserIHM().

◆ initialiserIconeSysteme()

IHMGroom::initialiserIconeSysteme ( )
private
Méthode qui permet à l'application de s'installer dans la barre système.

Affichage dans barre des taches.

Définition à la ligne 79 du fichier IHMGroom.cpp.

```
80 {
81     // Crée les actions
82     actionMinimiser = new QAction(QString::fromUtf8("Minimiser"), this);
83     actionMaximiser = new QAction(QString::fromUtf8("Maximiser"), this);
84     actionRestaurer = new QAction(QString::fromUtf8("Restaurer"), this);
85     actionQuitter = new QAction(QString::fromUtf8("&Quitter"), this);
86 
87     // Connecte les actions
88     connect(actionMinimiser, SIGNAL(triggered(bool)), this, SLOT(hide()));
89     connect(actionMaximiser, SIGNAL(triggered(bool)), this, SLOT(showMaximized()));
90     connect(actionRestaurer, SIGNAL(triggered(bool)), this, SLOT(showNormal()));
91     connect(actionQuitter, SIGNAL(triggered(bool)), qApp, SLOT(quit()));
92 
93     // Crée le menu
94     menuIconeSysteme = new QMenu(this);
95     menuIconeSysteme->addAction(actionMinimiser);
96     menuIconeSysteme->addAction(actionMaximiser);
97     menuIconeSysteme->addAction(actionRestaurer);
98     menuIconeSysteme->addSeparator();
99     menuIconeSysteme->addAction(actionQuitter);
100 
101     // Crée l'icône pour la barre de tâche
102     iconeSysteme = new QSystemTrayIcon(this);
103     iconeSysteme->setContextMenu(menuIconeSysteme);
104     iconeSysteme->setToolTip("Groom");
105     QIcon icone(":/groom.png");
106     iconeSysteme->setIcon(icone);
107     setWindowIcon(icone);
108 
109     connect(iconeSysteme, SIGNAL(messageClicked()), this,
SLOT(acquitterNotification()));
110     //connect(iconeSysteme, SIGNAL(activated(QSystemTrayIcon::ActivationReason)),
```

```
this, SLOT(aActiveIconeSysteme(QSystemTrayIcon::ActivationReason)));
111
112   iconeSysteme->show();
113   etatInitialIconeSysteme = true;
114 }
```

Références acquitterNotification(), actionMaximiser, actionMinimiser, actionQuitter, actionRestaurer, etatInitialIconeSysteme, iconeSysteme, et menuIconeSysteme.

Référencé par IHMGroom().

◆ initialiserIHM()

IHMGroom::initialiserIHM ( )
private
Initialise l'IHM GROOM.

A faire:
Intégrer le mode Détection de présence dans l'IHM
Définition à la ligne 165 du fichier IHMGroom.cpp.

```
166 {
167   uiIHMGroom->listeBluetooth->clear();
168   uiIHMGroom->boutonRechercherBluetooth->setEnabled(true);
169   uiIHMGroom->labelEtatConnexion->setText("État : déconnecté");
170   uiIHMGroom->boutonConnecterBluetooth->setEnabled(false);
171   uiIHMGroom->boutonDeconnecterBluetooth->setEnabled(false);
172   connect(uiIHMGroom->boutonRechercherBluetooth, SIGNAL(clicked(bool)), this,
SLOT(rechercherGroom()));
173   connect(uiIHMGroom->boutonConnecterBluetooth, SIGNAL(clicked(bool)), this,
SLOT(connecterGroom()));
174   connect(uiIHMGroom->boutonDeconnecterBluetooth, SIGNAL(clicked(bool)), this,
SLOT(deconnecterGroom()));
175
176   uiIHMGroom->boutonEnvoyerOccupant->setEnabled(false);
177   connect(uiIHMGroom->boutonEnvoyerOccupant, SIGNAL(clicked(bool)), this,
SLOT(envoyerAffichageOccupant()));
178
179   uiIHMGroom->lineEditNom->setPlaceholderText("Nom");
180   uiIHMGroom->lineEditPrenom->setPlaceholderText("Prénom");
181   uiIHMGroom->lineEditFonction->setPlaceholderText("Fonction");
182   uiIHMGroom->zoneTexteMessage->setPlaceholderText("Message à afficher");
183
184   uiIHMGroom->boutonAjouterOccupant->setEnabled(false);
185   uiIHMGroom->boutonEditerOccupant->setEnabled(false);
186   uiIHMGroom->boutonSupprimerOccupant->setEnabled(false);
187   connect(uiIHMGroom->listeOccupant, SIGNAL(currentIndexChanged(int)), this,
SLOT(selectionnerOccupant(int)));
188   connect(uiIHMGroom->boutonAjouterOccupant, SIGNAL(clicked(bool)), this,
SLOT(ajouterOccupant()));
189   connect(uiIHMGroom->boutonEditerOccupant, SIGNAL(clicked(bool)), this,
```

```
SLOT(modifierOccupant()));
190   connect(uiIHMGroom->boutonSupprimerOccupant, SIGNAL(clicked(bool)), this,
SLOT(supprimerOccupant()));
191   connect(uiIHMGroom->boutonRadioLibre, SIGNAL(clicked(bool)), this,
SLOT(envoyerEtatLibre()));
192   connect(uiIHMGroom->boutonRadioAbsent, SIGNAL(clicked(bool)), this,
SLOT(envoyerEtatAbsent()));
193   connect(uiIHMGroom->boutonRadioOccupe, SIGNAL(clicked(bool)), this,
SLOT(envoyerEtatOccupe()));
194   connect(uiIHMGroom->boutonRadioEntrer, SIGNAL(clicked(bool)), this,
SLOT(envoyerEtatEntrer()));
195   connect(uiIHMGroom->boutonPoussoirSonnette, SIGNAL(clicked(bool)), this,
SLOT(envoyerEtatSonnette()));
196   connect(uiIHMGroom->boutonActiverDesactiverPresence, SIGNAL(clicked(bool)),
this, SLOT(envoyerEtatPresence()));
197   connect(uiIHMGroom->boutonEnvoyerMessage, SIGNAL(clicked(bool)), this,
SLOT(envoyerMessageOccupant()));
198
203   connect(uiIHMGroom->widgetCalendrier, SIGNAL(clicked(QDate)), this,
SLOT(selectionnerDate(QDate)));
204
205   connect(communicationGroom, SIGNAL(groomDetecte(QString,QString)), this,
SLOT(detecterGroom(QString,QString)));
206   connect(communicationGroom, SIGNAL(connexionGroom(QString,QString)), this,
SLOT(afficherConnexionGroom(QString,QString)));
207   connect(communicationGroom, SIGNAL(deconnexionGroom(QString,QString)), this,
SLOT(afficherDeconnexionGroom(QString,QString)));
208   connect(communicationGroom, SIGNAL(rechercheTerminee()), this,
SLOT(arreterRechercheGroom()));
209 }
```

Références afficherConnexionGroom(), afficherDeconnexionGroom(), ajouterOccupant(), arreterRechercheGroom(), communicationGroom, connecterGroom(), deconnecterGroom(), detecterGroom(), envoyerAffichageOccupant(), envoyerEtatAbsent(), envoyerEtatEntrer(), envoyerEtatLibre(), envoyerEtatOccupe(), envoyerEtatPresence(), envoyerEtatSonnette(), envoyerMessageOccupant(), modifierOccupant(), rechercherGroom(), selectionnerDate(), selectionnerOccupant(), supprimerOccupant(), et uiIHMGroom.

Référencé par IHMGroom().

◆ rechercherGroom

void IHMGroom::rechercherGroom ( )
slot
Définition à la ligne 548 du fichier IHMGroom.cpp.

```
549 {
550   uiIHMGroom->listeBluetooth->clear();
551   uiIHMGroom->listeBluetooth->setEnabled(false);
552   uiIHMGroom->boutonRechercherBluetooth->setEnabled(false);
```

```
553    uiIHMGroom->boutonConnecterBluetooth->setEnabled(false);
554    uiIHMGroom->boutonDeconnecterBluetooth->setEnabled(false);
555    uiIHMGroom->labelEtatConnexion->setText("État : déconnecté");
556
557    communicationGroom->rechercher();
558 }
```

Références communicationGroom, CommunicationGroom::rechercher(), et uiIHMGroom.

Référencé par IHMGroom(), et initialiserIHM().

La documentation de cette classe a été générée à partir des fichiers suivants :
IHMGroom.h
IHMGroom.cpp

3.1.13.2 Classe CommunicationGroom

a. Documentation de la classe CommunicationGroom.

Déclaration de la classe CommunicationGroom.

Cette classe s'occupe de la communication des trames d'envoi et de reception.

b. Documentation des fonctions membres

◆ **initialiserBluetooth()**

```
void CommunicationGroom::initialiserBluetooth    (    )
private
```

Définition à la ligne 31 du fichier CommunicationGroom.cpp.

```
32 {
33    // vérifier la présence du Bluetooth
34    if(peripheriqueBluetooth.isValid())
35    {
36          // activer le bluetooth
37          peripheriqueBluetooth.powerOn();
38
39          // récupérer le nom du périphérique local
40          nomPeripheriqueBluetooth = peripheriqueBluetooth.name();
41          qDebug() << Q_FUNC_INFO << nomPeripheriqueBluetooth;
42
43          discoveryAgent = new QBluetoothDeviceDiscoveryAgent(this);
44          connect(discoveryAgent, SIGNAL(finished()), this,
SLOT(terminerRecherche()));
```

```
45          connect(discoveryAgent, SIGNAL(deviceDiscovered(QBluetoothDeviceInfo)),
this, SLOT(deviceDiscovered(QBluetoothDeviceInfo)));
46      }
47      else qDebug() << Q_FUNC_INFO << "Pas de bluetooh !";
48  }
```

Références deviceDiscovered(), discoveryAgent, nomPeripheriqueBluetooth, peripheriqueBluetooth, et terminerRecherche().
Référencé par CommunicationGroom().

◆ **rechercheTerminee**

void CommunicationGroom::rechercheTerminee ( )
signal
Référencé par terminerRecherche().

◆ **terminerRecherche**

void CommunicationGroom::terminerRecherche ( )
slot
Définition à la ligne 134 du fichier CommunicationGroom.cpp.

```
135 {
136   qDebug() << Q_FUNC_INFO;
137   emit rechercheTerminee();
138 }
```

Références rechercheTerminee().
Référencé par deviceDiscovered(), et initialiserBluetooth().

La documentation de cette classe a été générée à partir des fichiers suivants :
CommunicationGroom.h
CommunicationGroom.cpp

3.1.14 Informations

Auteur

Yuri Mota <motayuri13@gmail.com>

Date

2021

Version

1.0

Dépôt du code source

https://svn.riouxsvn.com/groom-2021

4. Alexander ROUGIER

4.1 Présentation personnelle

4.1.1 Rappel du besoin initial

- Gérer un portier connecté via une application programmée sous Android Studio (avec l’IDE Android Studio).

4.1.2 Organisation

Pour l'organisation du projet et la communication entre tous les membres du groupe, les outils suivants seront utilisés :

- **Subversion** qui est un logiciel libre de gestion de versions hébergé sur le site RiouxSVN pour l'ensemble du code source du projet.
- L'espace de stockage commun **Google Drive** pour tous les documents ressources.
  Beesbusy qui permet de gérer la planification des tâches dans le projet.

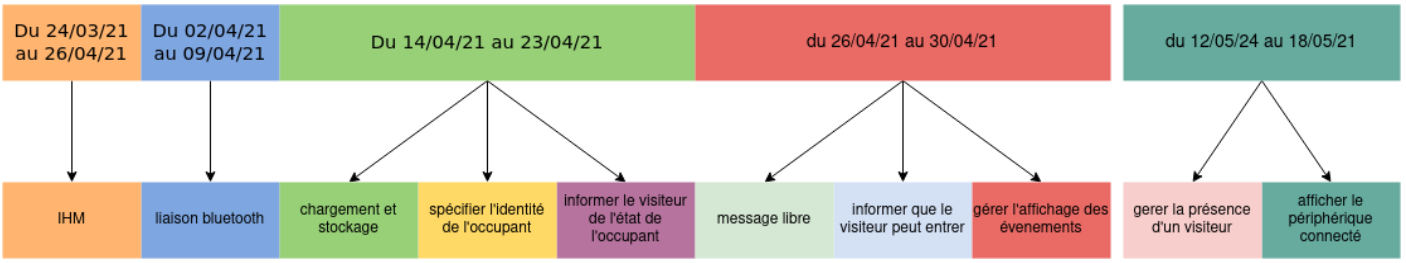
4.1.3 Objectifs

Communiquer avec le portier via l'application mobile et une liaison bluetooth

4.1.4 Répartition des tâches (globales)

Fonctionnalités	itération	priorité
Définir une IHM (Android studio	1	Haute
Mettre en place une liaison sans fil (bluetooth)(Android studio)	2	Moyenne
Ajouter ou modifier le nom prénom et fonction de l'occupant	2	Moyenne
Envoyer le nom prénom et fonction au module ESP pour l'affichage	2	Moyenne
Gérer l'activation ou désactivation du détecteur de présence d'un visiteur	2	Moyenne
Gérer un mode sonnette	2	Moyenne
Gérer l'affichage de l'appuie de la sonnette par le visiteur	3	Basse
Informer le visiteur de son état "Libre, Occupé, Absent"	2	Moyenne
Possibilité d'ajouter un message libre qui s'affichera alors sur l'écran du portier	2	Moyenne
Informer le visiteur que celui-ci peut "Entrer"	2	Moyenne
Chargement et stockage des paramètres en local dans un fichier INI	3	Basse
Afficher la présence d'un visiteur	3	Basse
Afficher l'état indiquée au visiteur sur l'application mobile	3	Basse

4.1.5 Planification



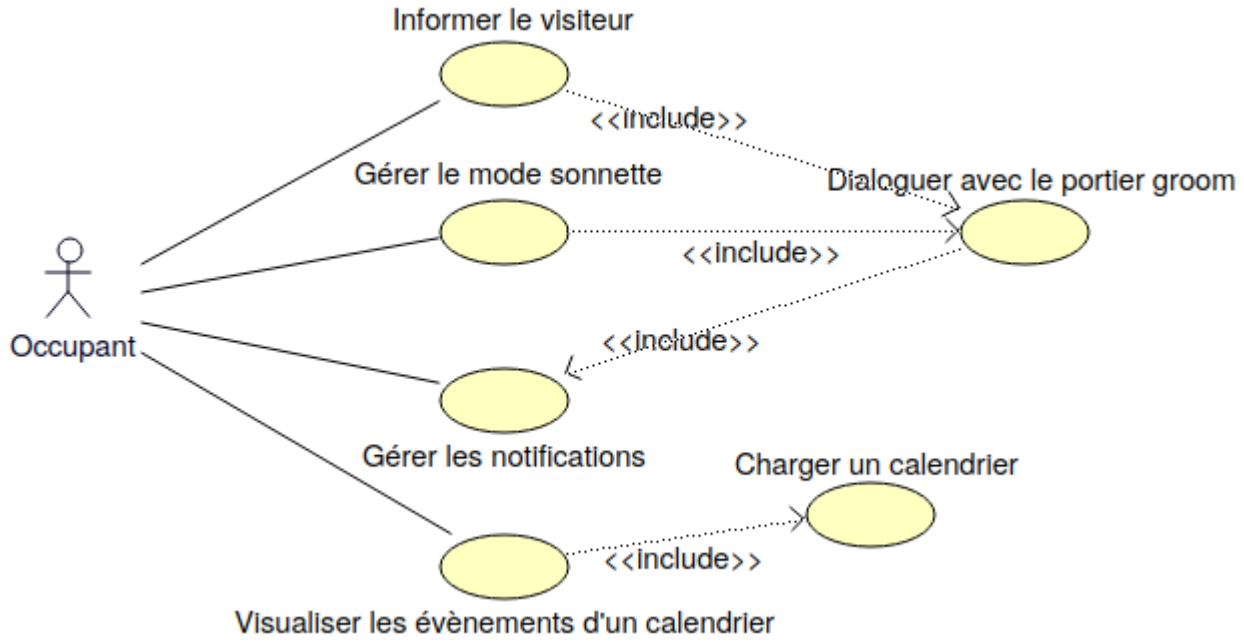
4.1.6 Outils de développement

Désignation	*Caractéristiques*
OS Poste de développement	PC sous GNU/Linux Ubuntu
EDI	Android Studio
API GUI	Android Studio 4.1.2
Atelier de génie logiciel	Bouml v7.11
OS Appareil mobile	Android 8.1.0 API niveau 26

L'appareil mobile est une tablette Samsung A6 de 2016, avec :

- Batterie
  7300 mAh
- Connectique
  Micro-USB-B 2.0, 3.5 mm Audio in/out
- Connectivité
  Bluetooth 4.2
- Mémoire
  16 GB
- Taille de la RAM
  2.0 GB

4.1.7 Diagramme de cas d'utilisation



L'acteur occupant est associé avec 4 fonctionnalités :

- Informer le visiteur
- Gérer le mode sonnette
- Gérer les notifications
- Visualiser les événements d'un calendrier

Pour informer le visiteur, il faut dialoguer avec le portier groom,

Pour gérer le mode sonnette, il faut dialoguer avec le portier groom,
Pour dialoguer avec le portier groom, il faut gérer les notifications,
Pour visualiser les évènements, il faut charger un calendrier.

4.1.8 Le protocole de communication

Le protocole de communication groom est basé sur des trames requêtes/réponses via une liaison bluetooth.

Protocole de communication Groom (version 0.9c)

Trame Groom

La trame est composée de caractères ASCII. Le délimiteur de champ est le **';'**. Le délimiteur de début est le **'$'** suivi de **GROOM** qui indique le nom du protocole. Le délimiteur de fin de trame est "**\r\n**".

Sens : Groom → PC/Tablette

Format :

$GROOM;ETAT;SONNETTE;PRESENCE;MODE_SONNETTE;MODE_PRESENCE\r\n

Le champ **ETAT** peut prendre 3 valeurs différentes :

- Libre **0**
- Absent **1**
- Occupé **2**

Le champ **SONNETTE** précise si un appel a été réalisé.

Le champ **PRESENCE** aura la valeur 1 dans le cas où une personne a été détectée devant le groom.

Le champ **MODE_SONNETTE** précise l'état d'activation **1** (activé) ou **0** (désactivé) de la sonnette.

Le champ **MODE_PRESENCE** précise l'état d'activation **1** (activé) ou **0** (désactivé) du détecteur de présence.

Cette trame est envoyée par le Groom lorsque celui-ci a détecté un changement d'état (**ETAT, SONNETTE** et/ou **PRESENCE**) ou de mode (**MODE_SONNETTE/MODE_PRESENCE**).

Exemple :

$GROOM;0;1;1;1;1\r\n

L'accès au bureau pour les visiteurs est possible (**ETAT=0** donc LIBRE). Le visiteur a "sonné" (**SONNETTE=1**) et il a été détecté (**PRESENCE=1**).

Trame Commande

Sens : PC/Tablette → Groom

Format :

$CMD;ORDRE;SONNETTE;PRESENCE\r\n

Le champ **ORDRE** peut prendre 4 valeurs différentes :

- Libre **0**
- Absent **1**
- Occupé **2**
- Entrez **3**

Le champ **SONNETTE** précise l'état d'activation **1** ou **0** (désactivé) de la sonnette.

Le champ **PRESENCE** précise l'état d'activation **1** ou **0** (désactivé) du détecteur de présence. Ceci est utile lorsque le Groom est placé dans une zone où de nombreux passages existent (toilette, couloir, …)

Exemple :

$CMD;1;0;0\r\n

L'accès au bureau pour les visiteurs est possible (**ORDRE=1** donc ABSENT). Le visiteur ne sera pas détecté (**PRESENCE=0** donc le détecteur de présence est désactivé) et ne pourra pas "sonner" (**SONNETTE=0** donc désactivée).

Trame Etat

Sens : PC/Tablette → Groom

Format :

$ETAT\r\n

La trame **ETAT** est une trame de requête permettant d'obtenir l'état courant du GROOM. Celui-ci renvoie alors une trame **GROOM**.

Trame de service (application vers système) :

L'application envoie périodiquement (toutes les secondes) une trame ALIVE pour maintenir la connexion ouverte.

Décomposition d'une trame :
$GROOM ; **A** \r\n
$GROOM ; **A**live \r\n

Remarque : la trame ne contient aucun espace

Le système répondra par une trame d'acquittement.

Décomposition d'une trame :
$GROOM ; **A** \r\n
$iotruck ; **A**ck \r\n

Remarque : la trame ne contient aucun espace

Trame Affichage

La trame est composée de caractères ASCII. Le délimiteur de champ est le **';'**. Le délimiteur de fin de trame est **"\r\n"**.

Sens : PC/Tablette → Groom

Format :

$AFFICHAGE;NOM;PRENOM;FONCTION\r\n

La trame **AFFICHAGE** transportera le **NOM,** le **PRENOM** et la **FONCTION** de la personne qui occupe le bureau.

Ces différents champs contiennent des chaînes de caractères qui seront utilisées directement pour l'affichage sur le groom.

Exemple :

$AFFICHAGE;COPIN;Olivier;DDFPT\r\n

La personne qui occupe le bureau est **Olivier COPIN** dont la fonction est **DDFPT** (Directeur Délégué aux Formations Professionnelles et Technologiques).

Trame MsgPerso

La trame est composée de caractères ASCII. Le délimiteur de champ est le ‘;’. Le délimiteur de fin de trame est “**\r\n**”.

Sens : PC/Tablette → Groom

Format :

$MSGPERSO;messageperso\r\n

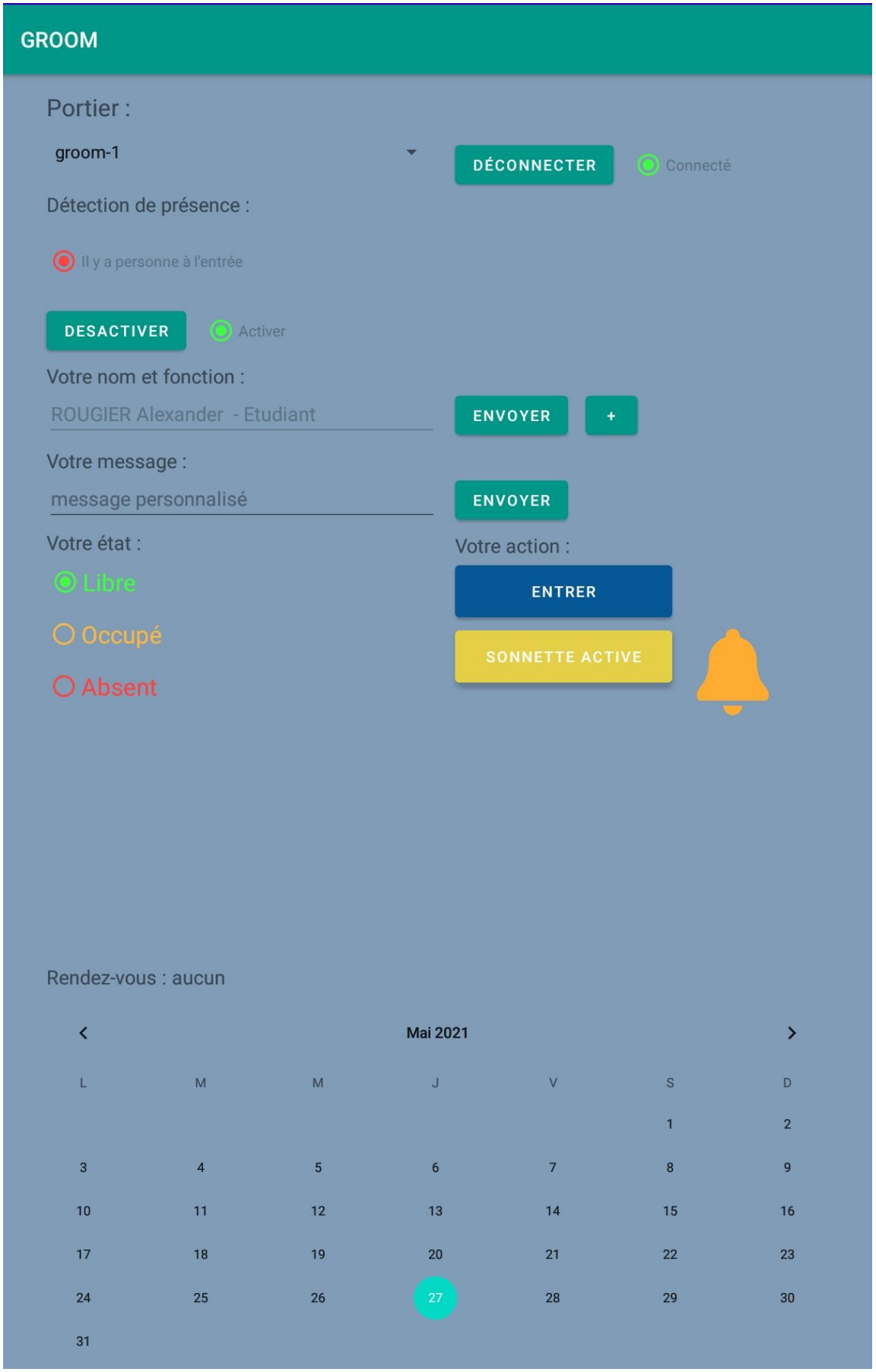
Cette trame sera envoyé que lorsque l’utilisateur du bureau voudra afficher un message personnalisé court.

Exemple :

$MSGPERSO;Je reviens dans 10 minutes\r\n

Le message **Je reviens dans 10 minutes** sera affiché sur l’écran du groom.

4.1.9 Maquette IHM



Cette IHM a plusieurs parties, la première partie est la connexion au module GROOM. Annoncée par un titre "Portier", elle indique le composant auquel elle est connectée, et permet la connexion et la confirmation de connexion.

Ensuite, la partie présence, annoncée par un titre "Détection de présence". Cette partie permet d'indiquer la présence ou non d'un individu se situant à l'entrée, et l'activation ou non de ce capteur de présence.

Troisièmement, la partie d'ajout de l'occupant, définie par le titre "Votre nom et fonction :", cette partie permet de changer le nom prénom et fonction via le bouton + se situant à droite. Cette modification est affichée sur le texte à gauche et le bouton envoyer permet l'envoie de cette modification.

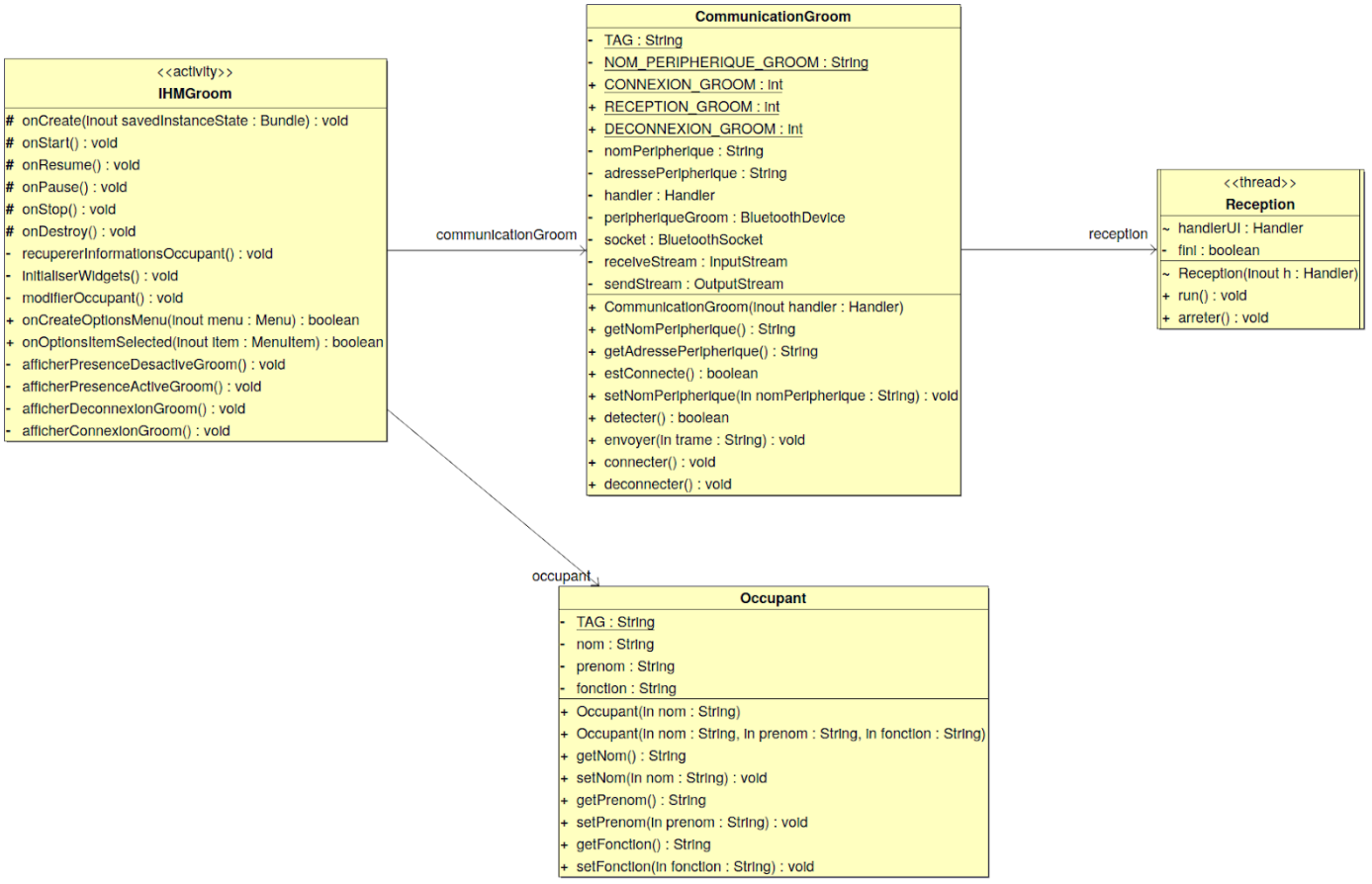
Ensuite, la partie message personnalisé, définie par le titre "Votre message :", qui permet d'envoyer un message personnel à l'individu visiteur en modifiant le texte et en l'envoyant avec le bouton envoyer.

Cinquièmement, la partie état de l'occupant définie par le titre " Votre état :", ce qui permet d'indiquer l'état dans lequel se trouve l'occupant au visiteur. Avec le bouton Libre, Occupé, Absent.

Ensuite, la partie pour indiquer l'action que l'occupant veut donner au visiteur, définie par le titre "Votre action :". Elle possède les boutons Entrer, qui permet de dire au visiteur d'entrer et le bouton sonnette, qui laisse au visiteur la possibilité de sonner si l'occupant est en état libre.

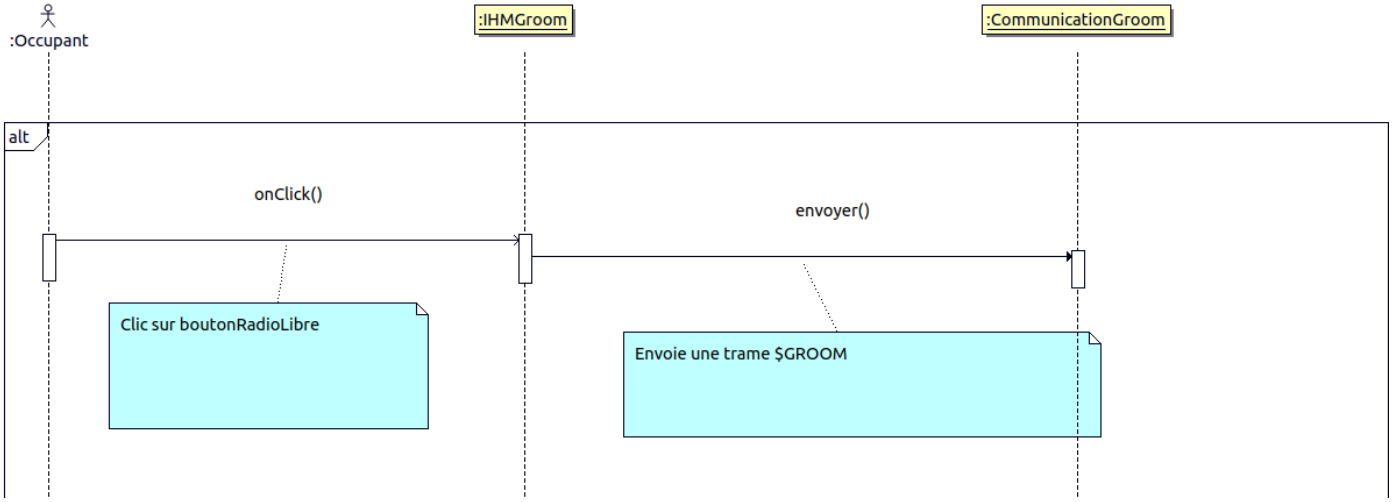
Enfin, la partie calendrier, se trouve en bas de l'IHM, nous montre un calendrier.

4.1.10 Diagramme de classes



4.1.11 Diagramme de Séquence

Diagramme de séquence d'envoie de trame de l'état Libre.



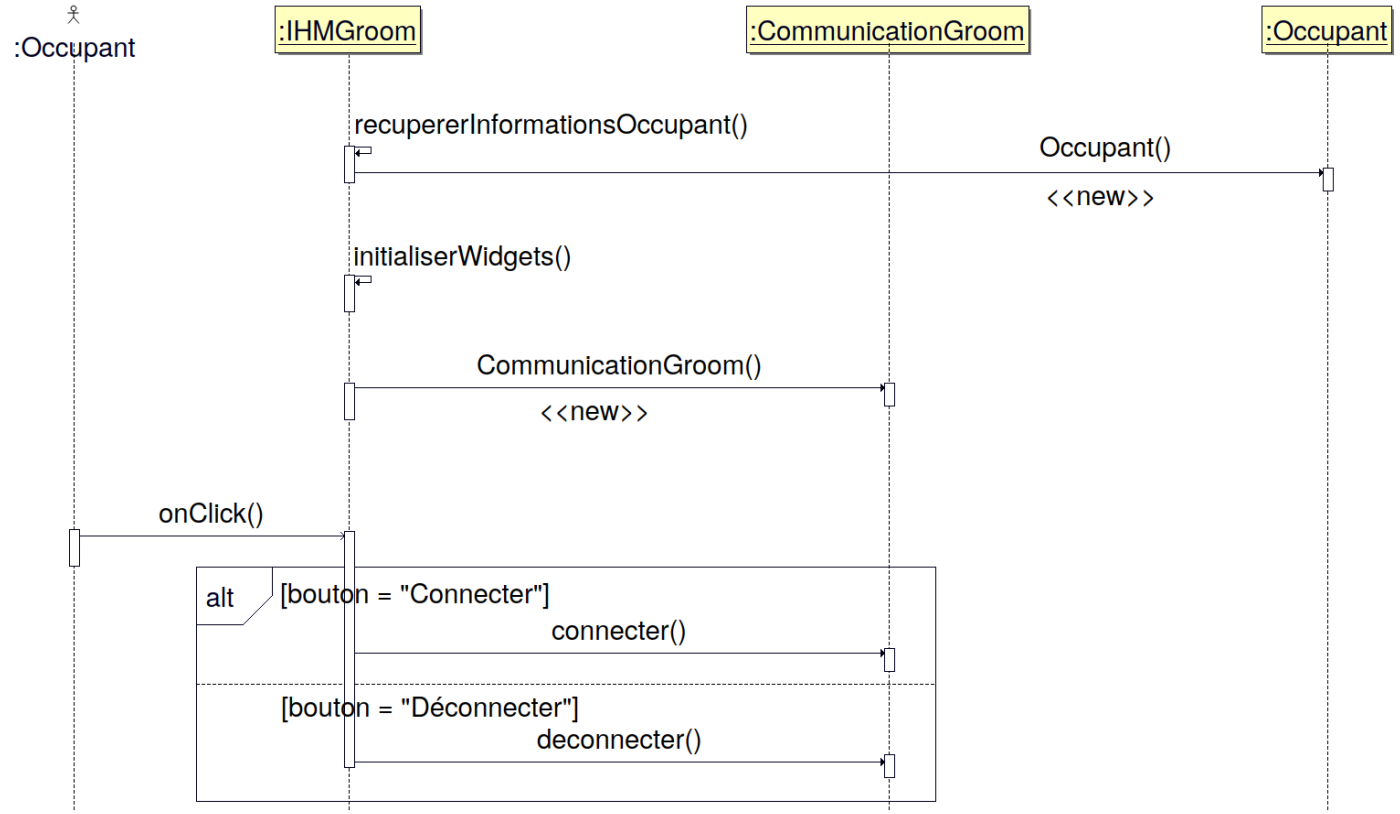
La fonction envoyer() :

```
◆ envoyer()
void com.example.groom.CommunicationGroom.envoyer ( String trame )

Définition à la ligne 128 du fichier CommunicationGroom.java.
129  {
130      // Vérifications
131      if(peripheriqueGroom == null || socket == null)
132     return;
133
134     new Thread()
135     {
136           @Override public void run()
137           {
138                try
139                {
140                    if(socket.isConnected())
141                    {
142                        sendStream.write(trame.getBytes());
143                        sendStream.flush();
144                    }
145
146                    Log.d(TAG, "Envoyer trame : " + trame);
147                }
148                catch (IOException e)
149                {
150                    Log.d(TAG, "Erreur écriture socket");
151                    e.printStackTrace();
152                }
153           }
154     }.start();
155  }
```

Le diagramme explique comment marche le changement d'état. L'occupant clique sur le bouton "Libre", la fonction onClick() s'exécute, cette méthode est une fonction appelée en publique. Lorsque l'utilisateur appuie sur un bouton, la fonction onClick() va exécuter une liste d'actions. L'action exécutée dans notre cas est la fonction envoyer() (la fonction se trouve au-dessus). Cette fonction, après avoir vérifié que la socket et le périphérique n'est pas null, elle convertit la trame en une séquence d'octets, et va ensuite envoyer la trame.

Diagramme séquence démarrage application




```
◆ recupererInformationsOccupant()
void com.example.groom.IHMGroom.recupererInformationsOccupant     (      )
    private

Récupère et crée l'objet occupant avec les informations stockées localement.

Définition à la ligne 202 du fichier IHMGroom.java.
203  {
204  // On récupère l'élément si il existe
205  if (sharedPreferences.contains(PREFERENCES_NOM_OCCUPANT))
206  {
207    nomOccupant = sharedPreferences.getString(PREFERENCES_NOM_OCCUPANT, "COPIN");
// null ou une valeur par défaut
208  }
209
210  if (sharedPreferences.contains(PREFERENCES_PRENOM_OCCUPANT))
211  {
212    prenomOccupant = sharedPreferences.getString(PREFERENCES_PRENOM_OCCUPANT,
```

```
"Olivier"); // null ou une valeur par défaut
213   }
214
215   if (sharedPreferences.contains(PREFERENCES_FONCTION))
216   {
217     fonctionOccupant = sharedPreferences.getString(PREFERENCES_FONCTION, "DDFPT");
// null ou une valeur par défaut
218   }
219
220     // Crée l'objet occupant avec les informations récupérées localement
221     occupant = new Occupant(nomOccupant, prenomOccupant, fonctionOccupant);
222   }
```

```
◆ Occupant() [1/2]
com.example.groom.Occupant.Occupant     (     String   nom    )

//Constructeur.

//Paramètres
      //nom    Le nom de l'occupant

//Définition à la ligne 29 du fichier Occupant.java.
30   {
31     this.nom = nom;
32     this.prenom = "";
33     this.fonction = "";
34   }

//Références com.example.groom.Occupant.nom

◆ Occupant() [2/2]
com.example.groom.Occupant.Occupant     (     String   nom,
       String      prenom,
       String      fonction
    )

/*Constructeur.

Paramètres
      nom    Le nom de l'occupant
      prenom    Le prénom de l'occupant
      fonction    La fonction de l'occupant
*/

//Définition à la ligne 42 du fichier Occupant.java.
43   {
44     this.nom = nom;
45     this.prenom = prenom;
46     this.fonction = fonction;
```

```
47  }
```

```
◆ initialiserWidgets()
void com.example.groom.IHMGroom.initialiserWidgets     (           )
    private

//Initialise les widgets de l'IHM.

Définition à la ligne 227 du fichier IHMGroom.java.
228  {
229  titreConnexion = (TextView) findViewById(R.id.titreConnexion);
230  texteNomOccupant = (TextView) findViewById(R.id.texteNomOccupant);
231  listeAppareilsBluetooth = (Spinner) findViewById(R.id.listeAppareilsBluetooth);
232
233  ArrayAdapter<String> adapter = new ArrayAdapter<String>(getApplicationContext(),
android.R.layout.simple_spinner_item, communicationGroom.getListePeripheriques());
234  listeAppareilsBluetooth.setAdapter(adapter);
235  listeAppareilsBluetooth.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener()
236  {
237  @Override
238  public void onItemSelected(AdapterView<?> parent, View view, int position, long
id)
239  {
240  Log.d(TAG, "onClick() listeAppareilsBluetooth : " +
communicationGroom.getListePeripheriques());
241  }
242
243  @Override
244  public void onNothingSelected(AdapterView<?> parent)
245  {
246
247  }
248  });
249
250  boutonConnexion = (Button) findViewById(R.id.boutonConnexion);
251  boutonConnexion.setOnClickListener(new View.OnClickListener()
252  {
253  @Override
254  public void onClick(View v)
255  {
256  Log.d(TAG, "onClick() boutonConnexion : " +
boutonConnexion.getText().toString());
257  if(boutonConnexion.getText().toString().equals("Connecter"))
258  {
259  communicationGroom.connecter();
260  }
261  else
```

```
262  {
263  communicationGroom.deconnecter();
264  }
265  }
266  });
267  etatConnexion = (RadioButton) findViewById(R.id.etatConnexion);
268
269  boutonAjoutOccupant = (Button) findViewById(R.id.boutonAjoutOccupant);
270  boutonAjoutOccupant.setOnClickListener(new View.OnClickListener()
271  {
272  @Override
273  public void onClick(View v)
274  {
275  Log.d(TAG, "onClick() boutonAjoutOccupant");
276  modifierOccupant();
277  }
278  });
279  boutonEnvoieOccupant = (Button) findViewById(R.id.boutonEnvoieOccupant);
280  boutonEnvoieOccupant.setOnClickListener(new View.OnClickListener()
281  {
282  @Override
283  public void onClick(View v)
284  {
285  communicationGroom.envoyer("$AFFICHAGE;" + occupant.getNom() + ";" +
occupant.getPrenom() + ";" + occupant.getFonction() + "\r\n");
286  Log.d(TAG, "onClick() boutonEnvoieOccupant");
287  }
288  });
289  texteNomFonction = (EditText) findViewById(R.id.texteNomFonction);
290  texteNomFonction.setText(occupant.getNom() + " " + occupant.getPrenom() + " - "
+ occupant.getFonction());
291  texteNomFonction.setEnabled(false);
292
293  // zoneMessageOccupant
294  titreMessageOccupant = (TextView) findViewById(R.id.titreMessageOccupant);
295  texteMessageOccupant = (EditText) findViewById(R.id.texteMessageOccupant);
296  envoyerMessageOccupant = (Button) findViewById(R.id.envoyerMessageOccupant);
297  envoyerMessageOccupant.setOnClickListener(new View.OnClickListener()
298  {
299  @Override
300  public void onClick(View v)
301  {
302  communicationGroom.envoyer("$MSGPERSO;" + texteMessageOccupant.getText() +
"\r\n");
303  Log.d(TAG, "onClick() envoyerMessageOccupant");
304  }
305  });
306
307  // zoneEtat
```

```
308  titreEtat = (TextView) findViewById(R.id.titreEtat);
309  boutonRadioLibre = (RadioButton) findViewById(R.id.boutonRadioLibre);
310  boutonRadioLibre.setOnClickListener(new View.OnClickListener()
311  {
312  @Override
313  public void onClick(View v)
314  {
315  Log.d(TAG, "onClick() boutonRadioLibre");
316  etatOccupant = ETAT_LIBRE;//temporaire
317  communicationGroom.envoyer("$GROOM;" + etatOccupant + ";" + etatModeSonnette +
";" + etatModeDetecteur + "\r\n");
318  }
319  });
320  boutonRadioOccupe = (RadioButton) findViewById(R.id.boutonRadioOccupe);
321  boutonRadioOccupe.setOnClickListener(new View.OnClickListener()
322  {
323  @Override
324  public void onClick(View v)
325  {
326  Log.d(TAG, "onClick() boutonRadioOccupe");
327  etatOccupant = ETAT_OCCUPE;//temporaire
328  communicationGroom.envoyer("$GROOM;" + etatOccupant + ";" + etatModeSonnette +
";" + etatModeDetecteur + "\r\n");
329  }
330  });
331  boutonRadioAbsent = (RadioButton) findViewById(R.id.boutonRadioAbsent);
332  boutonRadioAbsent.setChecked(false);
333  boutonRadioAbsent.setOnClickListener(new View.OnClickListener()
334  {
335  @Override
336  public void onClick(View v)
337  {
338  Log.d(TAG, "onClick() boutonRadioAbsent");
339  etatOccupant = ETAT_ABSENT; //temporaire
340  communicationGroom.envoyer("$GROOM;" + etatOccupant + ";" + etatModeSonnette +
";" + etatModeDetecteur + "\r\n");
341  }
342  });
343
344  // zoneEntrerSonnette
345  boutonEntrer = (Button) findViewById(R.id.boutonEntrer);
346  boutonEntrer.setOnClickListener(new View.OnClickListener()
347  {
348  @Override
349  public void onClick(View v)
350  {
351  Log.d(TAG, "onClick() boutonEntrer");
352  etatOccupant = ETAT_ENTRER;//temporaire
353  communicationGroom.envoyer("$GROOM;" + etatOccupant + ";" + etatModeSonnette +
```

```
";" + etatModeDetecteur + "\r\n");
354  }
355  });
356  boutonSonnette = (Button) findViewById(R.id.boutonSonnette);
357  boutonSonnette.setOnClickListener(new View.OnClickListener()
358  {
359  @Override
360  public void onClick(View v)
361  {
362  Log.d(TAG, "onClick() boutonSonnette");
363  etatOccupant = ETAT_OCCUPE; //temporaire, il faudra faire en fonction de la
trame reçu, pas le faire nous
364  etatModeSonnette = SONNETTE_ON; //temporaire
365  communicationGroom.envoyer("$GROOM;" + etatOccupant + ";" + etatModeSonnette +
";" + etatModeDetecteur + "\r\n");
366  }
367  });
368  //textView = (TextView)findViewById(R.id.textView);
369
370  // Donne le focus
371  boutonSonnette.setFocusableInTouchMode(true);
372  boutonSonnette.requestFocus();
373
374  boutonPresence= (Button) findViewById(R.id.boutonPresence);
375  boutonPresence.setOnClickListener(new View.OnClickListener(){
376  public void onClick(View v)
377  {
378  Log.d(TAG, "onClick() boutonPresence :" + boutonPresence.getText().toString());
379  if(boutonPresence.getText().toString().equals("Desactiver"))
380  {
381  afficherPresenceDesactiveGroom();
382  etatModeDetecteur = DETECTEUR_OFF;
383  communicationGroom.envoyer("$GROOM;" + etatOccupant + ";" + etatModeSonnette +
";" + etatModeDetecteur + "\r\n");
384
385  }else
386  {
387  afficherPresenceActiveGroom();
388  etatModeDetecteur = DETECTEUR_ON;
389  communicationGroom.envoyer("$GROOM;" + etatOccupant + ";" + etatModeSonnette +
";" + etatModeDetecteur + "\r\n");
390  }
391  }
392  });boutonRadioActivationPresence = (RadioButton)
findViewById(R.id.boutonRadioActivationPresence);
393
394  afficherDeconnexionGroom();
395  }
```

```
Références com.example.groom.IHMGroom.communicationGroom,
com.example.groom.CommunicationGroom.getListePeripheriques(),
com.example.groom.IHMGroom.listeAppareilsBluetooth,
com.example.groom.IHMGroom.texteNomOccupant, et
com.example.groom.IHMGroom.titreConnexion.

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

◆ CommunicationGroom()

```
com.example.groom.CommunicationGroom.CommunicationGroom     (     Handler
handler    )

Définition à la ligne 47 du fichier CommunicationGroom.java.
48  {
49  this.handler = handler; // permet de communiquer par message avec l'activité
50
51  // Détection du périphérique groom
52  detecter();
53
54  // Création de la socket pour communiquer
55  try
56  {
57  if(peripheriqueGroom != null)
58  {
59  Log.d(TAG, "Création socket");
60  socket =
peripheriqueGroom.createRfcommSocketToServiceRecord(UUID.fromString("00001101-0000-10
00-8000-00805F9B34FB"));
61  receiveStream = socket.getInputStream();
62  sendStream = socket.getOutputStream();
63  }
64  }
65  catch (IOException e)
66  {
67  Log.d(TAG, "Erreur création socket");
68  socket = null;
69  }
70
71  // Création de du thread Reception
72  if(socket != null)
73  {
74  reception = new Reception(handler);
75  }
76  }
```

◆ connecter()

```
void com.example.groom.CommunicationGroom.connecter     (          )

Définition à la ligne 157 du fichier CommunicationGroom.java.
158  {
159  // Vérifications
160  if(peripheriqueGroom == null || socket == null)
161  return;
162
163  new Thread()
164  {
165  @Override public void run()
166  {
167  // Connexion de la socket
168  try
169  {
170  socket.connect();
171
172  // signale à l'activité la connexion
173  Message msg = Message.obtain();
174  msg.what = CONNEXION_GROOM;
175  handler.sendMessage(msg);
176
177  Log.d(TAG, "Connexion socket");
178  }
179  catch (IOException e)
180  {
181  Log.d(TAG, "Erreur connexion socket");
182  e.printStackTrace();
183  }
184
185  // Démarrage de la reception
186  if(socket.isConnected())
187  {
188  reception.start();
189  }
190  }
191  }.start();
192  }
```

◆ deconnecter()

```
void com.example.groom.CommunicationGroom.deconnecter     (        )

Définition à la ligne 194 du fichier CommunicationGroom.java.
195  {
196  // Vérifications
197  if(peripheriqueGroom == null || socket == null)
198  return;
199
200  // Exécute la déconnexion dans un Thread pour éviter un blocage
201  new Thread()
202  {
203  @Override public void run()
204  {
205  // Déconnexion de la socket
206  try
207  {
208  socket.close();
209
210  // signale à l'activité la déconnexion
211  Message msg = Message.obtain();
212  msg.what = DECONNEXION_GROOM;
213  handler.sendMessage(msg);
214
215  Log.d(TAG, "Déconnexion socket");
216  }
217  catch (IOException e)
218  {
219  Log.d(TAG, "Erreur déconnexion socket");
220  e.printStackTrace();
221  }
222
223  // Arrêt de la reception
224  if(!socket.isConnected())
225  {
226  reception.arreter();
227  }
228  }
229  }.start();
230  }
```

Le diagramme de séquence au dessus explique les actions qui se déroulent lors du démarrage de l'application.
En premier, la fonction recupererInformationsOccupant() va récupérer le nom, prénom et fonction de l'occupant pour l'afficher sur l'IHM.
Ces nouvelles informations sont récupérées dans la classe Occupant.
L'IHMGroom va initialiser tous les différents widgets avec la fonction initialiserWidgets().
Une nouvelle connexion va être faite avec la fonction CommunicationGroom().
Les différentes fonctions exécutées au démarrage étant faites, l'application est prête à démarrer.

L’acteur occupant va cliquer sur le bouton “Connecter” et entrer dans un test, ce qui va exécuter la fonction connecter(). Ce qui va lancer le démarrage via la classe CommunicationGroom de la connexion entre l’appareil mobile et le module ESP32.
Sinon, ce test va exécuter la fonction deconnecter() si l’acteur appui sur le bouton “Déconnecter”

4.1.12 Tests de validation

Tests	Comment faire les tests	Validation
Vérification connexion au module Bluetooth	Visuel + débogage log	✔
Vérification de l'ajout du nom+prénom+fonction et envoie	Visuel + débogage log	✔
Vérification de l'envoie d'un message personnel	Visuel + débogage log	✔
Activation/désactivation du fonctionnement du capteur de présence	Visuel + débogage log	✔
Sélection et envoie des états de l'occupant	Visuel + débogage log	* ✔
Sélection et envoie des impératif et de la sonnette	débogage log	* ✔
Prendre connaissance d'une présence à l'entrée	Visuel	* ✘

4.1.13 Recette

Fonctionnalités réalisées :

- Connexion bluetooth
- activation/désactivation capteur de présence
- entrer le nom et la fonction de l'occupant
- envoyer le nom et la fonction de l'occupant via le bluetooth
- entrer un message personnelle de l'occupant
- envoyer le message personnelle de l'occupant
- envoyer les états de l'occupant
- envoyer l'ordre d'entrer
- envoyer l'autorisation d'utiliser la sonnette

4.1.13 Informations

Auteur

Alexander Rougier alexander.rougier@gmail.com

Date

2021

Version

1.0

Dépôt du code source

https://svn.riouxsvn.com/groom-2021