INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

PROJET D'APPROFONDISSEMENT ET D'OUVERTURE

COMPAGNON VIRTUEL SOUS ANDROID

Institut national des sciences appliquées de Rouen

Rédigé par :
Jitao XU
Chenxin LU

Proposé par :
Alexandre PAUCHET

ASI
2 janvier 2018

Table des matières

Introduction

Dans le cadre de notre scolarité au sein du département Architecture des Systèmes d'Information (ASI) de l'Institut National des Sciences Appliquées (INSA) de Rouen, nous devons réaliser au cours de notre premier semestre de quatrième année un Projet d'Approfondissement et d'Ouverture (PAO), c'est-à-dire un travail sur plusieurs mois, seul ou en équipe, dans lequel nous pouvons mettre en pratique nos connaissances apprises en cours et/ou apprendre de nouvelles connaissances sur des sujets en rapport avec notre formation. Nous avons donc tous les deux décidé de nous intéresser à un sujet proposé par M. Alexandre Pauchet : la création d'un compagnon virtuel sous Android. Ce PAO permet d'une part de mettre en application nos acquis en Java de troisième année, et d'autre part de découvrir la conception d'une application mobile sous Android, chose totalement nouvelle pour nous deux.

Chapitre1

1 Analsye des besoins

1.1 Liste des besoins

Le but principal du projet est de développer des nouvelles fonctionnalités sur l'application Compagnon Virtuel déjà existante, notre verision sera version 4. Avant de commencer le projet, l'application est déjà fonctionnelle, elle propose une interaction avec un personnage animé. Ce dernier est capable d'échanger via reconnaissance vocale et synthèse vocale. L'analyse des requêtes est déportée sur un serveur externe de dialogue intelligent : pandorabots. Le compagnon virtuel s'exprime aussi via des animations. Pendant la préparation du projet, nous avons proposé de réaliser les fonctionnalités suivantes :

- effectuer une recherche dans googlemaps et afficher une carte dans l'application.
- envoyer sms.
- ajouter et supprimer un réveil.
- afficher un calendrier au sein de l'application
- intégrer le processus de gestion de calendrier au chatbot externe.
- afficher le prochain événement dans le calendrier, il est capable de mettre à jour le prochain événement s'il y a une modification dans le calendrier.

Autrement, nous avons proposé de améliorer l'interface d'utilisateurs de l'application car nous n'étions pas satisfaits avec l'interface d'utilisateur actuelle.

1.2 Rendus

En résumé, les rendus demandés sont :

- une application fonctionnelle sous Android,
- un rapport sur le déroulement du projet,
- une documentation du code source,
- un guide utilisateur,
- une vidéo pour faire une démonstration d'application.

Ils doivent être fournis avant la fin du premier semestre de ASI 4.1, afin d'être compabilisés dans la moyenne semestrielle.

Chapitre 2

2 Spécification

2.1 Faisabilité du projet

Dans un premier temps, nous avons suivi un cours sur www.udacity.com pour apprendre la base des connaissances de développer une application sous Android. Ce cours nous permet de développer une simple application, par exemple un compteur de points qui est utilisé dans le match du basket sous Android nous-même. Donc, nous pensons que nous sommes capable de commencer notre PAO.

Ensuite, nous avons effectué une brève étude bibliographique quant à la faisabilité du développent de fonctionnalités que nous avons proposé. Nous avons trouvé des APIs du développement sous Android. Dans ce cas là, nous pensions que tous les fonctionnalités pouvaient être développés sous Android sans trop de difficulté.

Concernant la partie de la réponse intelligente, comme les anciens PAOs sur ce sujet, nous avons décidé d'utiliser le Pandorabots, outils Internet permettant d'héberger un grand nombre de fichiers AIML contenant les question/réponses, pour nous aider de réaliser tous les fonctionnalités. Le choix du IDE a été fait rapidement, nous avons choisi Android Studio qui est un IDE très moderne.

Notre application peut être fonctionnée sur Android 4.4 ou supérieur avec une connexion Internet permanante.

2.2 Analyse Descendante

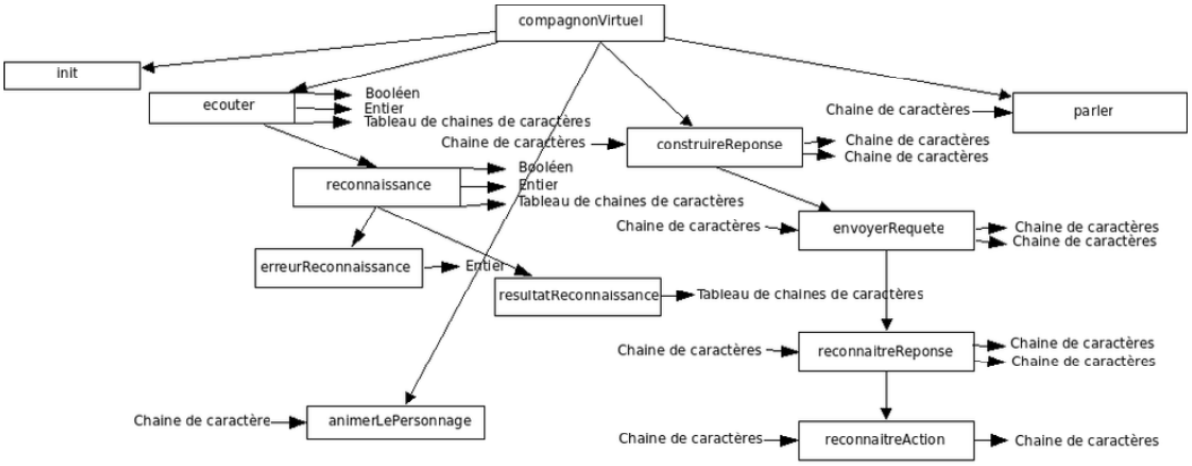

FIGURE 1 – Analyse descendante de la version ancienne.

L'analyse descendante a été décrit dans le rapport de 1ère version.

Chapitre 3

3 Conception Préliminaire

Cette étape nous permet, à partir des différents éléments de l'analyse, de mettre en forme les fonctions et procédures afin d'en expliciter les nouveaux fonctionnements.

3.1 Diagramme de cas d'utilisation

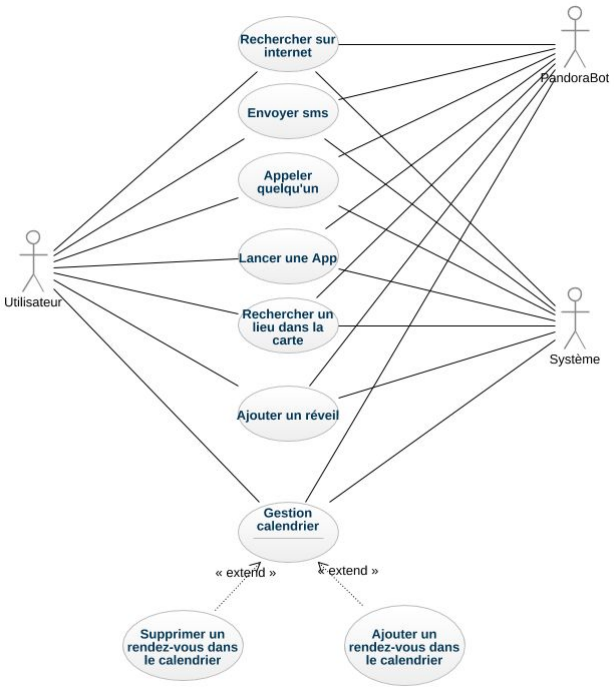

FIGURE 2 – Diagrammes de cas d'utilisation.

3.2 Diagrammes de Sequence

C'est un diagramme de séquence pour les fonctionnalités que nous avons proposé à réaliser. Les parties à la boite noire est la partie du code qui a été complété dans les dernières versions. Ce que nous avons construit est la partie au-desous de ***process_oobContent(String oobContent, String textToSpeak)***.

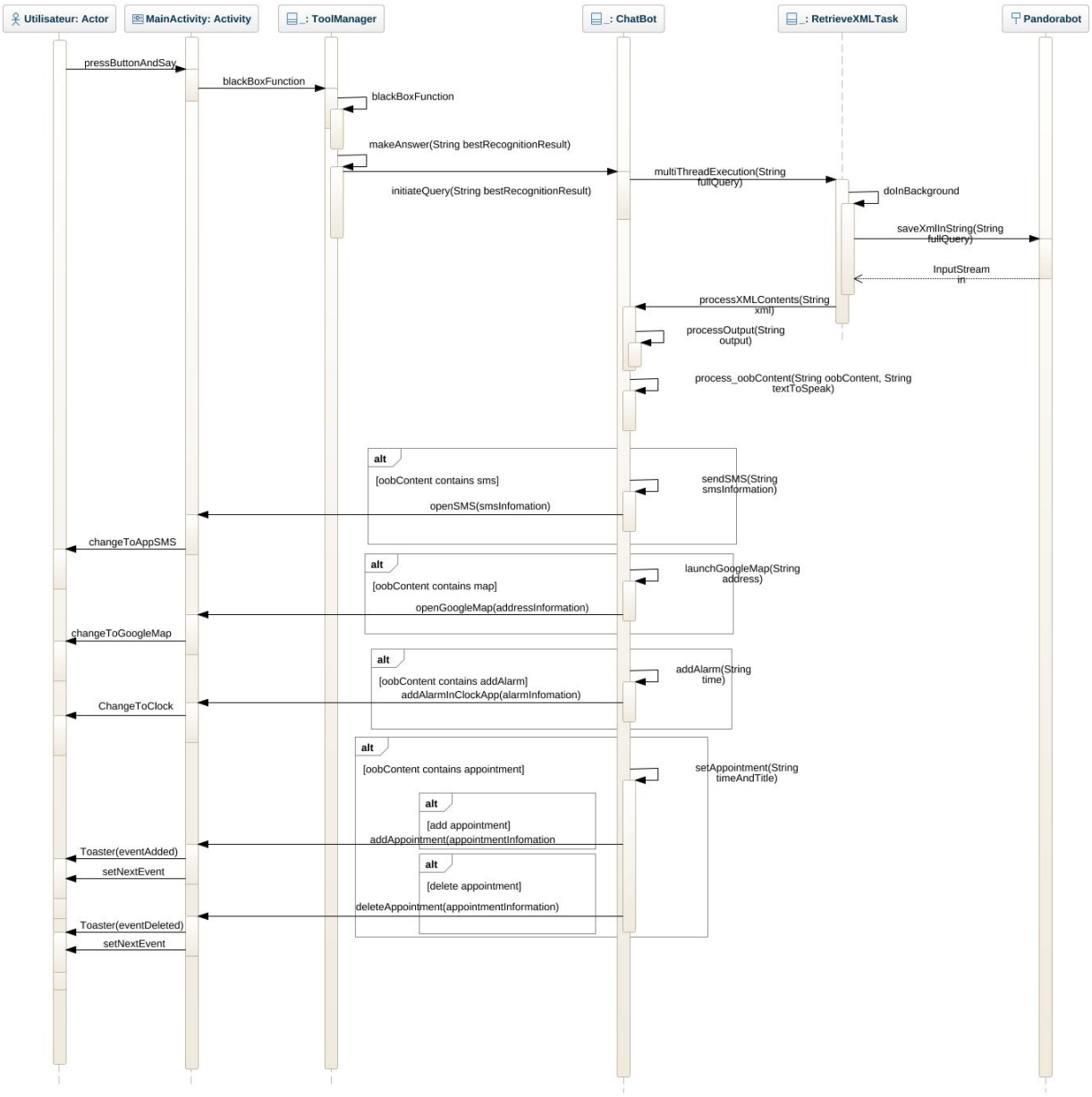

FIGURE 3 – Diagramme de Sequence.

3.3 Signatures Partie Chatbot

procédure setAppointement (E/S Gcal : GestionCalendar, E oobContent : String, operationType : String)

fonction setBeginTimeAndGetTitle (E oobContent : String, beginTime : Calendar, operationType : String) : String

procédure googleQuery (E/S googleSearchText : String)

procédure launchApp (E app : String)

procédure launchUrl (E/S url : String)

procédure launchGoogleMap (E/S address : String)

procédure sendSMS (E oobContent : String)

procédure makePhoneCall (E oobContent : String)

procédure addAnAlarm(E oobContent : String)

procédure deleteAnAlarm (E oobContent : String)

3.4 Signatures Partie ToolManager

procédure setNextEvent()
fonction getNextEvent() : String

Chapitre 4

4 Conception Détaillée

Cette étape nous permet, à partir de conception préliminaire, de préciser l'algorithme des fonctions ou procédures principales.

En effet, nous avons construit notre conception détaillée sur les codes déjà existants. Dans ce cas-là, les algorithmes contenaient les expressions spécifiques de dévoloppement Android.

4.1 addAnAlarme

```
procédure addAnAlarm(E oobContent:String)
début
    Entier hours <- première partie d'information d'heures
    si deuxième partie d'information d'heures est vide
        Entier minutes <- 0
    sinon
        Entier minutes <- deuxième partie d'information d'heures
    finsi
    Intent intent <- new Intent(AlarmClock.ACTION_SET_ALARM)
    intent.putExtra(AlarmClock.EXTRA_HOUR, hours)
    intent.putExtra(AlarmClock.EXTRA_MINUTES, minutes)
    si oobContent contient balise "<repetition>"
        ChaineDeCaractère days <- chaine de caractère entre la balise "<repetition>"
        daysNumber <- nouvel tableau d'entier
        si days contient "jours"
            pour i de 1 à 7
                daysNumber.ajouter(i)
            finPour
        sinon
            si days contient "dimanche"
                daysNumber.ajouter(1)
            finsi
            si days contient "lundi"
                daysNumber.ajouter(2)
            finsi
            si days contient "mardi"
                daysNumber.ajouter(3)
            finsi
            si days contient "mecredi"
                daysNumber.ajouter(4)
            finsi
            si days contient "jeudi"
                daysNumber.ajouter(5)
            finsi
            si days contient "vendredi"
                daysNumber.ajouter(6)
            finsi
            si days contient "samedi"
                daysNumber.ajouter(7)
            finsi
        finsi
        intent.putExtra(AlarmClock.EXTRA_DAYS, daysNumber)
    finsi
    callingActivity.startActivity(intent)
fin
```

Listing 1 – addAnAlarm

4.2 sendSMS

```
procédure sendSMS(E oobContent : String)
début
    String sendSmsTo <- Chaine de caractère entre la balise "<people>"
    si la première caractère de sendSmsTo n'est pas une lettre
        si oobContent contient la balise "<message>"
            String smsContent <- Chaine de caractère entre la balise "<message>"
            Intent intent <- new Intent(Intent.ACTION_SENDTO, Uri.parse("smsto:" +
                sendSmsTo))
            intent.setData(Uri.parse("sms_body" + smsContent))
            commencer l'activité intent par callingActivity
        finsi
    sinon
        Cursor c <- callingActivity.getContentResolver().query(ContactsContract.Contacts.
            CONTENT_URI, null, null, null, null)
        String name <- ""
        String number <- ""
        String numberToSendSms <- ""
        String id
        bouge c au premier
        booléen trouve <- faux
        tant que non trouve
            si c.getString(0) est non null
                name <- CDC de la colonne de ContactsContract.Contacts.
                    DISPLAY_NAME
                id <- CDC de la colonne de ContactsContract.Contacts._ID
                si ContactsContract.Contacts.HAS_PHONE_NUMBER est vrai
                    Cursor pCur <- callingActivity.getContentResolver().query(
                        ContactsContract. CommonDataKinds.Phone.CONTENT_URI, null,
                        ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " =?",
                         new String[]{id}, null);
                    tant que pCur peut bouger à la suivante
                        number <- CDC de la colonne de ContactsContract.
                            CommonDateKinds.Phone.NUMBER
                    fin tant que
                    ferme pCur
                finsi
                String[] words <- CDC de sendSmsTo séparée par " "
                StringBuilder sb <- new StringBuilder
                si longueur de words[0] est supérieur à 0
                    sb.append(words[0])
                    pour j de 1 à longueur de words
                        sb.append(" ")
                        sb.append(words[j])
                    finPour
                finsi
                String nom <- sb.toString()
                si name = nom
                    numberToSendSms <- number
                    trouve <- vrai
                finsi
            finsi
            bouge c à la suivante
        fin tant que
        c.close()
        si oonContent contient la balise "<message>"
            String smsContent <- CDC entre la balise "<message>"
            Intent intent <- new Intent(Intent.ACTION_SENDTO, Uri.parse("smsto:" +
                numberToSendSms))
```

```
            intent.putExtra("sms_body", smsContent)
            commencer l'activité intent par callingActivity
        finsi
    finsi
fin
```

Listing 2 – sendSMS

Cette fonction est modifiée selon la fonction ***makePhoneCall(String oobContent)***. Le principe d'algorithme reste la même. Nous avons changé ce qui concernent à appeler à ce qui concernent à envoyer SMS.

4.3 setAppointement

```
procédure setAppointement(E/S Gcal : GestionCalendar, E oobContent, operationType :
    String)
début
    Calendar beginTime <- temps réel
    String title <- setBeginTimeAndGetTitle(oobContent, beginTime, operationType)
    si operationType contient la balise "</add>"
        Calendar endTime <- beginTime
        Gcal.ajouterRDV(title, beginTime, endTime)
    sinon
        Calendar endTime <- le dernier minute du jour de beginTime
        Gcal.supprimerRDV(title, beginTime, endTime)
    finsi
fin
```

Listing 3 – setAppointement

4.4 setBeginTimeAndGetTitle

```
fonction SetBeginTimeAndGetTitle(oobContent : String, beginTime : Calendar,
    operationType : String) : String
début
    String date <- Chaine de caractère entre la balise "<eventdate>"
    String title <- chaine vide
    Entier day <- 0
    Entier month <- 0
    Entier year <- 0
    Entier hour <- 0
    Entier minute <- 0
    Entier i <- 0
    String[] words <- séparer date par ' '
    si longueur de words = 0
        String sDay <- date
    sinon
        String sDay <- words[i]
    finsi
    si sDay est un Entier
        day <- parseInt(sDay)
        i <- i + 1
    sinon
        day = beginTime.DAY
    finSi
    Dictionnaire<String, Integer> months <- nouvel Hashmap
    months.put("janvier", 0)
    months.put("février", 1)
```

```
    months.put("mars", 2)
    months.put("avril", 3)
    months.put("mai", 4)
    months.put("juin", 5)
    months.put("juillet", 6)
    months.put("août", 7)
    months.put("septembre", 8)
    months.put("octobre", 9)
    months.put("novembre", 10)
    months.put("décembre", 11)
    si words[i] est dans months
        month <- months.get(words[i])
        i <- i + 1
    sion
        month <- beginTime.MONTH
    finSi
    si words[i] est un entier
        year <- parseInt(words[i])
        i <- i + 1
    sinon
        year = beginTime.YEAR
    finSi
    si oobContent contient la balise "<eventtime>"
        String time <- Chaine de caractère entre la balise "<eventtime>"
        String[] hr <- Chaine de caractère séparée par "h"
        si hr[0] est un entier
            hour <- parseInt(hr[0])
        sinon
            hour <- 0
        finSi
        si longueur de hr supérieure à 1
            minute <- parseInt(hr[1])
        sinon
            minute <- 0
        finsi
        si oobContent contient la balise "<event>"
            title <- Chaine de caractère entre la balise "<event>"
        sinon
            title <- Chaine de caractère entre la balise "</eventtime>" et la balise
                operationType
        finsi
    sinon
        hour <- beginTime.HOUR
        minute <- beginTime.MINUTE
        si i inférieur à la longueur de words
            title <- sous-chaine de date commence à indice de words[i]
        finsi
    finsi
    si operationType = "</delete>"
        beginTime.set(year, month, day, 0, 0)
    sinon
        beginTime.set(year, month, day, hour, minute)
    finsi
    retourner title
fin
```

Listing 4 – setBeginTimeAndGetTitle

4.5 getNextEvent

```
fonction getNextEvent() : String;
début
    Context context <- callingActivity.getApplicationContext()
    GestionCalendar gc <- new GestionCalendar(context)
    Cursor cursor <- gc.getEvents()
    mettre le curseur au premier
    Entier eventNumber <- nombre d'événement
    si eventNumber > 0
        StringBuilder sb = new StringBuilder
        Entier Long[] eventMillis <- tableau d'entité long à la taille eventNumber + 10
        Entier j <- 0
        répéter
            eventMillis[j] = le temps de début d'événement j
            j <- j + 1
        jusqu'à cursor arrive au dernier
        tirer tableau eventMillis
        mettre le curseur au premier
        Entier i <- 0
        tant que l'heure actuelle >= eventNumber[i]
            i <- i + 1
        fin tant que
        tant que le temps de début d'événement i <> eventMillis[i]
            cursor.moveToNext()
        fin tant que
        Entier Long endMillis <- le temps de fin d'événement trouvé
        String eventTitle <- titre d'événement trouvé
        Date startDate <- le temps de début d'événement trouvé
        Date endDate <- le temps de fin d'événement trouvé
        mettre les dates au format "d MM yyyy, H:m"
        String sb = "L'événement plus proche:\n" + eventTitle + startDate + endDate
        retourner sb
    sinon
        retourner vide
    finsi
fin
```

Listing 5 – getNextEvent

Chapitre 5

5 Développememnt

La partie développement s'est déroulée en quatre sous parties. Tout d'abord, nous nous sommes informés des méthodes proposées sur Android pour mettre en place les fonctionnalités précisés dans le Chapitre 4. Ensuite nous avons ajouté plusieurs catégories de questions/réponses sur le site Pandorabots pour faire fonctionner nos fonctionnalités. Enfin, nous avons modifié l'interface d'utilisateurs de l'application pour améliorer l'expérience d'utilisation. Dans ce cas là, Nous avons changé le personnage d'animation pour afficher plus d'informations sur l'activité principale. Ces quatre sous parties ont été implémenté itérativement tout au long du développement.

5.1 Mis en place des différentes foncitonnalités

Comme nous l'avons dit précédemment, nous avons suivi un cours sur *www.udacity.com* pour apprendre la base de développer une application sous Android et nous avons effectué une recherche bibliographique pour mettre en place les foncition suivantes : launchGoogleMap, sendSMS, addAnAlarm, deleteAnAlarm, setAppointement, getNextEvent, setNextEvent. Les cinq premières fonctions ont été implémenté dans la classe Chatbot, les deux suivantes dans la classe ToolManager.

Toutes les foncitions sont appelées dans la méthode ***process_oobContent(String oobContent, String textToSpeak)*** sauf getNextEvent et setNextEvent. Le paramètre oobContent contient l'information retourner par le serveur Pandorabots entre "<oob>" et "</oob>". Le deuxième "textToSpeak" ne concerne pas nos fonctions.

5.1.1 Lancer Google Map

Premièrement, selon notre recherche, toutes les chaînes de caractère transmises aux intentions Google Maps doivent être encodées sous forme d'URI. Dans notre fonction, il faut remplacer tous les espaces ' ' par "%20" ou '+'. Nous avons choisi de les remplacer par '+'. Ensuite, nous avons créé un instance de ***Intent***, qui permet de lancer une application depuis l'activité actuelle.

En effet, nous avions essayé d'afficher la carte de Google Map dans notre application, néanmoins elle ne garde pas tous les fonctionnalité de l'application Google Map. Nous avons donc décidé de l'abandonner. Nous allons le préciser dans la chapitre 6.

5.1.2 Envoyer SMS

Nous proposons deux types de destination du sms : un numéro de téléphone mobile ou un correspondant dans le carnet d'adresses. Pour les réaliser, il faut toujours déterminer le type de première caractère de destination. Si c'est une lettre, donc nous prenons le mode 'correspondant', sinon, nous prenons le mode 'numéro de téléphone mobile'. Grâce au instance ***Intent***, notre fonction pouvons lire le String qui décrit la destination du sms.

Si nous sommes tombé dans le deuxième mode, nous allons chercher le numéro de téléphone mobile dans le carnet d'adresses, et puis envoyer le sms par le numéro.

Pareil, nous écrivons le contenu du sms par l'instance ***Intent***.

5.1.3 Ajouter et Supprimer un réveil

Idée principale :

Pour ajouter un réveil, simplement, nous déterminons la date et l'heure du réveil avec la reconnaissance vocale. Ensuite, nous utilisons toujours un instance de ***Intent*** qui nous permet de réaliser l'ajout des informations pour un réveil. Après, c'est important de déterminer la répétition du réveil. Par différents chaines de caractères, nous trouvrons la demande de répétition d'utilisateur. Finalement, nous finisons l'ajout du réveil.

Pour supprimer un réveil, c'est plus simple. Nous cherchons la date et l'heure du réveil et le supprimer directement.

Pour les détails du code :

Nous avons trouvé que la chaîne de reconnaissance vocale pour le temps est sous format "*h*", par exemple "14h35". Nous avons donc séparé la chaîne par 'h' et pris la première partie comme l'heure du réveil. Nous avons mis la minute du réveil à 0 par défaut s'il manque la deuxième partie dans la chaîne. Il faut ajouter une permission de régler l'alarme dans le fichier ***AndroidManifest.xml*** :

<uses−permission android:name="com.android.alarm.permission.SET_ALARM"/>

Nous avons implémenté la fonction ***deleteAnAlarm(String oobContent)***. Pour réaliser la fonctionnalité, nous devions utiliser une constante ***AlarmClock.ACTION_DISMISS_ALARM*** qui a besoin d'une API au niveau supérieur ou égal à 23. Cependant, les appareils que nous avions ne possédait qu'au niveau 19. En conséquence, nous ne pourrions pas réaliser cette fonctionnalité.Nous allons le préciser dans la chapitre 6.

5.1.4 Gestion du rendez-vous

Ajouter et Supprimer un rendez-vous

Nous avons modifié la version précédente pour intégrer le traitement de gestion de calendrier au serveur externe, Pandorabots. Nous avons gardé les méthodes pour initialiser l'instance de ***GestionCalendar*** et les méthodes ***ajouterRDV(String titre, Calendar beginTime, Calendar endTime)*** et ***supprimerRDV(String titre, Calendar beginDate,Calendar endDate)***. Ces deux dernières nous permet d'ajouter ou de supprimer un rendez-vous dans l'agenda de l'appareil.

Nous avons créer une méthode ***setAppointement(GestionCalendar Gcal, String oobContent, String operationType)*** qui nous permet de distinguer l'ajout et la suppression du rendez-vous. Pour pouvoir utiliser les méthodes déjà existantes, il faut avoir le titre, la date de début et la date de fin du rendez-vous. Pour simplifier le traitement, nous avons créer une méthode ***setBeginTimeAndGetTitle(String oobContent, Calendar beginTime, String operationType)*** qui régler la date de début et retourner le titre de rendez-vous en même temps. Dans cette dernière méthode, le paramètre ***beginTime*** en entrée est la date actuel du système. Nous avons met à défaut toutes les parties de la date commes la date actuel. Nous avons vérifié la chaîne ***oobContent*** dans l'ordre suivantes : jour, mois, an, heure, minute, titre. Pour chaque partie précisée, nous avons met à jour l'information. Si l'utilisateur n'a pas précisé le titre du rendez-vous, le titre par défaut sera une chaîne vide.

La méthode ***setBeginTimeAndGetTitle*** permet l'utilisateur de ne pas préciser n'importe quelle partie d'une date tant que les informations précisées sont dans l'ordre.

Mettre à jour le prochain rendez-vous

Nous avons aussi supprimé la méthode ***prochainRDV*** pour créer une nouvelle dans la classe ***ToolManager***. La méthode ***getNextEvent()*** permet de retourner une chaîne de caractère

qui contient les informations du prochain rendez-vous. Dans la méthode, nous avons créer un instance de ***Cursor*** qui peut se déplacer comme un pointeur. Nous cherchions dans la liste d'événement stocké dans l'agenda de l'appareil pour trouver l'événement le plus proche de la date actuel.

5.2 Ajout plusieurs catégories de questions/réponses sur le site Pandorabots

Nous continuions à utiliser le compte créé par les étudiants précédents pour ajouter des catégories de questions/réponses sur ***https ://www.pandorabots.com/botmaster/fr/home***, avec les identifiants suivants :

Email : ***alexandre.levacher@insa-rouen.fr***
Mot de passe : ***PAOCompagnonVirtuel3***

Tout les informations d'une quesion/réponse sont comprises entre *<**category**>* et *</**category**>*. La balise *<**pattern**>* contient la question et la balise *<**template**>* est la partie de réponse intelligente. La balise *<**oob**>* se situe dans la balise *<**template**>*.

5.2.1 Ajouter un réveil sans répétition

```
<category><pattern>SONNER POUR *</pattern><template><oob><alarmclock>
<add><star/></add></alarmclock></oob></template></category>

<category><pattern>AJOUTER UN RÉVEIL À* </pattern><template><srai>SONNER
    POUR <star/></srai></template></category>

<category><pattern>AJOUTER UN RÉVEIL POUR * </pattern><template><srai>
    SONNER POUR <star/></srai></template></category>
```

5.2.2 Ajouter un réveil avec répétition

```
<category><pattern>AJOUTER UN RÉVEIL À* POUR TOUS LES * </pattern><
    template><oob><alarmclock><add><star index="1"/></add><repetition><star
    index="2"/></repetition></alarmclock></oob></template></category>

<category><pattern>AJOUTER UN RÉVEIL POUR * POUR TOUS LES * </pattern><
    template><srai>AJOUTER UN RÉVEIL À<star index="1"/>POUR TOUS LES<star
    index="2"/></srai></template></category>

<category><pattern>AJOUTER UN RÉVEIL POUR TOUS LES * POUR * </pattern><
    template><srai>AJOUTER UN RÉVEIL À<star index="2"/>POUR TOUS LES<star
    index="1"/></srai></template></category>

<category><pattern>AJOUTER UN RÉVEIL POUR TOUS LES * À* </pattern><
    template><srai>AJOUTER UN RÉVEIL À<star index="2"/>POUR TOUS LES<star
    index="1"/></srai></template></category>
```

5.2.3 Supprimer un réveil

```
<category><pattern>SUPPRIMER UN RÉVEIL DE *</pattern><template><oob>
<alarmclock><delete><star/></delete></alarmclock></oob></template></category>

<category><pattern>SUPPRIMER UN RÉVEIL À*</pattern><template><srai>
    SUPPRIMER UN RÉVEIL DE <star/></srai></template></category>
```

5.2.4 Ajouter un rendez-vous

```
<category><pattern>AJOUTER UN RENDEZ-VOUS LE *</pattern><template><oob><
    appointment><add><eventdate><star/></eventdate></add></appointment></oob
    ></template></category>

<category><pattern>AJOUTER UN RENDEZ-VOUS LE * À* , *</pattern><template><
    oob><appointment><add><eventdate><star index="1"/></eventdate><eventtime><
    star index="2"/></eventtime><event><star index="3"/></event></add></
    appointment></oob></template></category>

<category><pattern>AJOUTER UN RENDEZ-VOUS LE * À*</pattern><template><oob
    ><appointment><add><eventdate><star index="1"/></eventdate><eventtime><star
     index="2"/></eventtime></add></appointment></oob></template></category>
```

5.2.5 Supprimer un rendez-vous

```
<category><pattern>SUPPRIMER UN RENDEZ-VOUS LE *</pattern><template><oob
    ><appointment><delete><eventdate><star/></eventdate></delete></appointment
    ></oob></template></category>

<category><pattern>SUPPRIMER UN RENDEZ-VOUS LE * À* INTITULÉ *</pattern
    ><template><oob><appointment><delete><eventdate><star index="1"/></
    eventdate><eventtime><star index="2"/></eventtime><event><star index="3"/></
    event></delete></appointment></oob></template></category>

<category><pattern>SUPPRIMER UN RENDEZ-VOUS LE * INTITULÉ *</pattern><
    template><oob><appointment><delete><eventdate><star index="1"/></eventdate
    ><eventtime>0</eventtime><event><star index="2"/></event></delete></
    appointment></oob></template></category>
```

5.2.6 Réaliser une recherche en Google Map

```
<category><pattern>_MAPS *</pattern><template><oob><maps><star/></maps></
    oob></template></category>

<category><pattern>_ MAPS *</pattern><template><oob><maps><star/></maps></
    oob></template></category>
```

5.2.7 Envoyer un sms

```
<category><pattern>_ENVOYER UN SMS POUR * POUR DIRE *</pattern><template
    ><oob><send><people><star index="1"/></people><message><star index
    ="2"/></message></send></oob></template></category>

<category><pattern>_ENVOIE UN SMS POUR * POUR DIRE *</pattern><template><
    oob><send><people><star index="1"/></people><message><star index="2"/></
    message></send></oob></template></category>
```

5.3 Modification de l'interface d'utilisateur

Premièrement, nous avons séparé le centre de l'interface d'utilisateur à deux partie. La partie gauche affiche les cinq reconnaissaces vocales du voix de l'utilisateur plus possibles. La partie droite affiche la réponse de notre application s'il existe. En bas de l'écran, nous avons ajouté un ***textField*** pour afficher le prochain événement dans le calendrier. Ce ***textField*** va se mettre à jour s'il existe un changement d'événement plus proche dans le calendrier grâce à la fonction ***getNextEvent***. Autrement, nous avons changé le personnage d'animation. Nous allons parler dans la paragraphe suivante.

5.4 Changement du personnage

Nous avons choisi un personnage d'animation de Microsoft qui est open source. Nous avons téléchargé l'ensemble d'image du personnage tout d'abord. Puis, nous avons utilisé l'outil pour le couper par chaque action. Pour différentes actions, nous avons proposé un temp de continu. Avec ces images, nous avons réalisé les animations suivantes : saluer, écouter, trouver une erreur, parler, rire, répondre, etc. Ce changement du personnage améliore notre l'interface d'utilisateur, de plus, il nous permet d'avoir plus d'espace sur l'écran pour afficher d'autres informations. Nous pensons que c'est un changement indispensable.