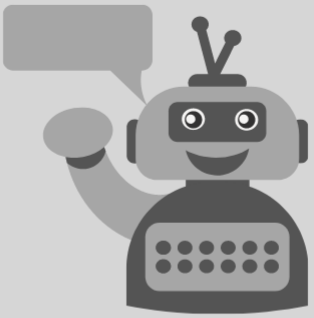
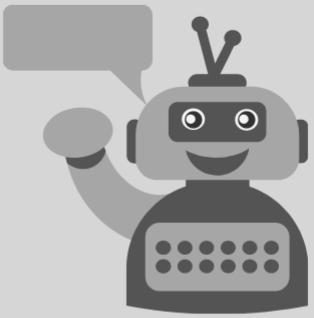
Berton Margot
Debono Juliette
Kacer Inès
Marjollet Iris

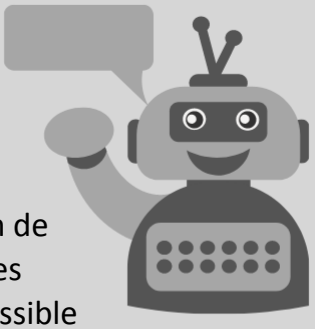


Rapport
[PRO3600]

Table des matières



Introduction



Lors de l'utilisation d'un emploi du temps classique, il est parfois contraignant de devoir remplir une à une les informations telles que : le nom de l'événement, son type (profesionnel ou personnel), ses horaires… De plus, les informations ne sont visibles que dans l'odre chronologique et il n'est pas possible de visualiser une catégorie spécifique d'événements.
Afin que l'utilisation d'un emploi du temps soit plus instinctive, nous proposons un chatbot permettant à l'utilisateur d'aller plus vite dans la gestion de son temps. En effet, il permet de converser avec un assistant virtuel avec de simples phrases.

L'objectif du projet est de **créer une application utilisant un chatbot**, c'est-à-dire un agent conversationnel constituant un **assistant** pour l'utilisateur. Ce chatbot permettra à l'utilisateur de gérer son emploi du temps et ses rendez-vous. Le projet aura donc pour objectif de créer une application contenant ce chatbot, avec une interface graphique permettant d'interagir avec l'assistant.

I. Analyse des besoins

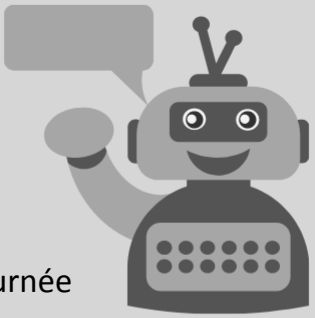
Format :

Afin que notre chatbot soit utilisable et utilisé par un utilisateur quelconque, nous avons fait le choix de le faire sous forme d'une application. Elle contiendra un chatbot relié à un emploi du temps. Pour pouvoir utiliser des bibliothèques plus développées, les interactions avec le chatbot seront en anglais.

Implémentation :

Afin de pouvoir utiliser le langage de programmation **Java** avec lequel nous sommes familières, nous avons décidé de faire notre application sur Android en utilisant **Android Studio**. Nous utilisons également **SQLite** pour pouvoir gérer l'emploi du temps stocké dans une base de données. Le chatbot est implémenté en python, en utilisant le modèle de compréhension **NLU** (Natural Language Understanding) ainsi qu'un **API** (Application Programming Interface) ou des **librairies logicielles** qui nous permettent d'interagir avec le NLU. Enfin, pour l'implémentation de l'interface graphique ainsi que pour la connexion avec le serveur python, nous utilisons **XML** et **Java**.

Cahier des charges :



JIMI le chatbot devra être capable :

- d'afficher l'emploi du temps de l'utilisateur pour un jour donné
- d'ajouter/supprimer/modifier un rendez-vous à l'emploi du temps de l'utilisateur pour une date précise
- de choisir un créneau libre afin de placer un rendez-vous dans une journée donnée

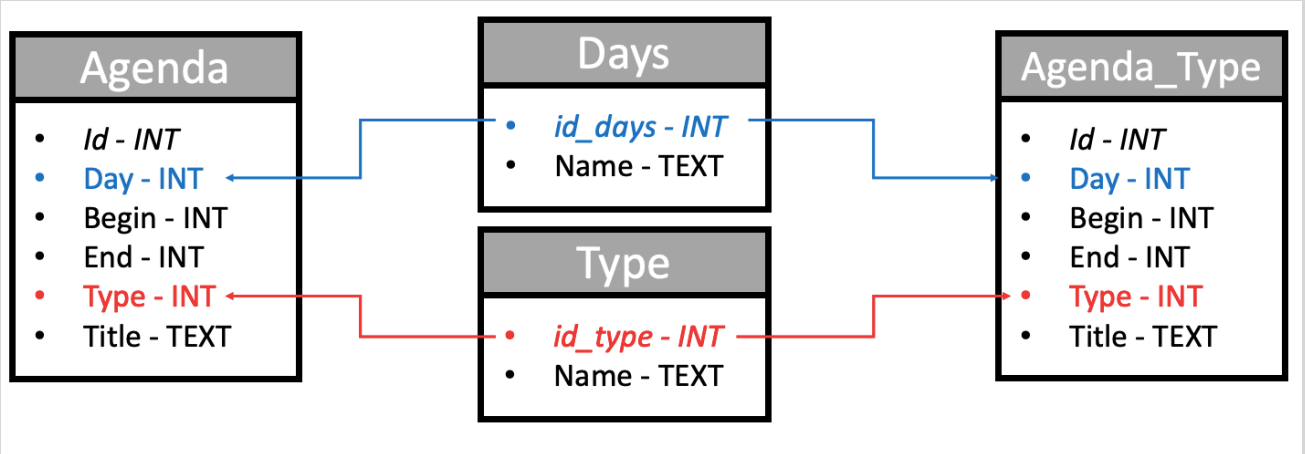
Fonctionnalités :

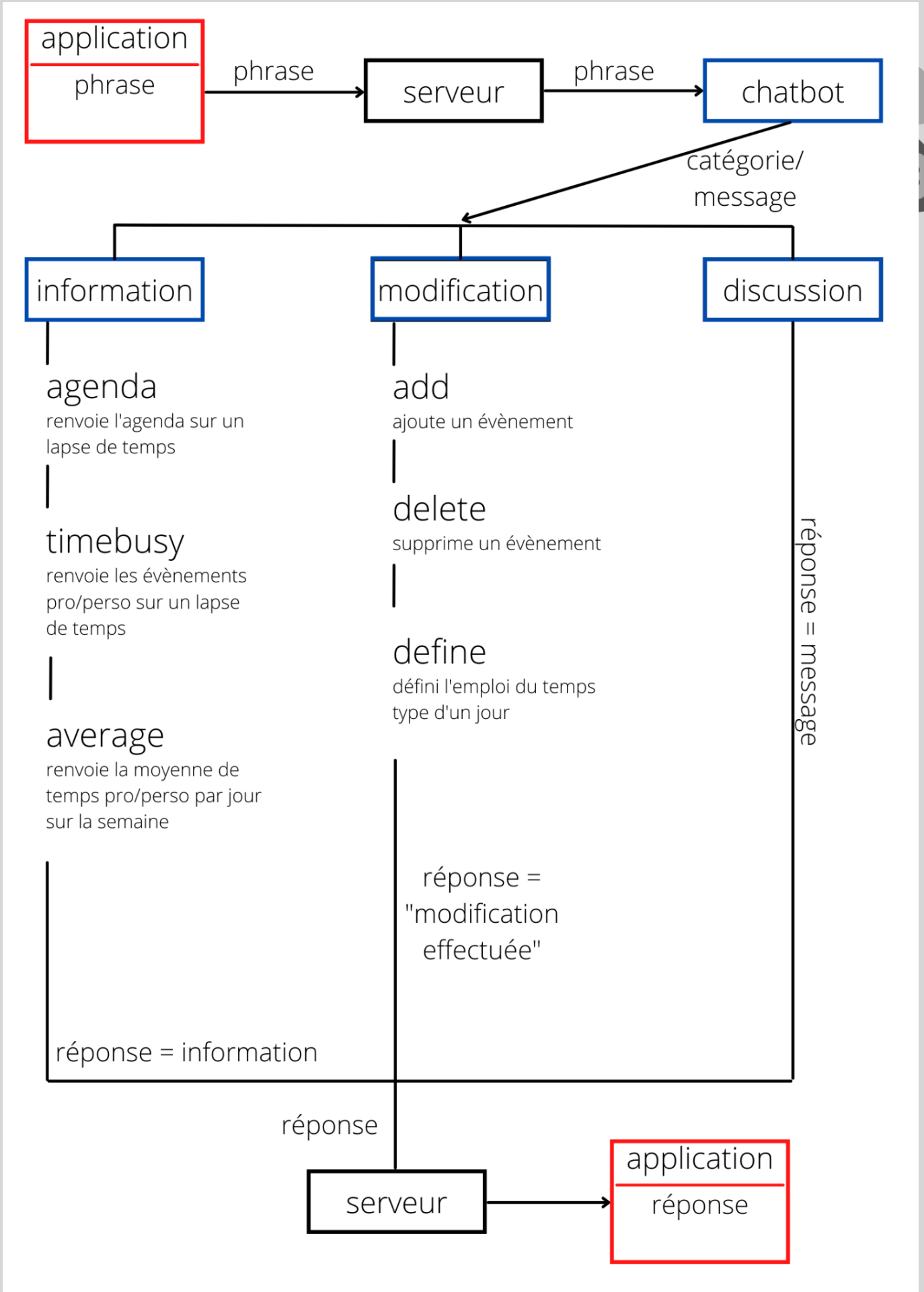
Les fonctionnalités du chatbot sont :

- utiliser une base de données correspondant à un emploi du temps
- modifier la base de donnée en fonction des requêtes de l'utilisateur
- permettre à l'utilisateur d'écrire des requêtes en langage naturel grâce à une interface graphique
- transformer les requêtes en langage naturel en requêtes permettant d'accéder à une base de données
- répondre de manière pertinente aux requêtes de l'utilisateur

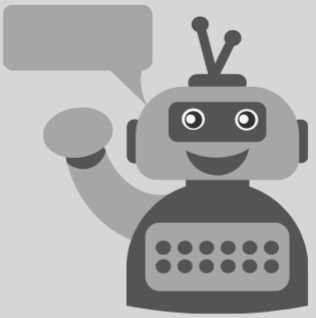
II. Conception

Ci-dessous, se trouve le squelette du projet. L'application Java sert d'interface, et les phrases inscrites par l'utilisateur sont envoyées à un serveur, puis traitées par le chatbot. Il traite le message pour identifier la catégorie à laquelle correspond la phrase (information sur l'agenda, modification de l'agenda ou discussion) ainsi que le message. À l'aide d'un fichier python et d'une base de données (dont le schéma se trouve ci-dessous), on réalise la requête demandée et on retourne une réponse, qui est à nouveau envoyée au serveur, puis à l'application. L'utilisateur obtient donc l'information qu'il souhaite.




application
phrase
phrase
serveur
phrase
chatbot
catégorie/
message
information
modification
discussion
agenda
renvoie l'agenda sur un lapse de temps
timebusy
renvoie les évènements pro/perso sur un lapse de temps
average
renvoie la moyenne de temps pro/perso par jour sur la semaine
add
ajoute un évènement
delete
supprime un évènement
define
défini l'emploi du temps type d'un jour
réponse = message
réponse = "modification effectuée"
réponse = information
réponse
serveur
application
réponse

III. Gestion de projet



Planning prévisionnel :

Février

- Lancement du projet avec notre tuteur Walid Gaaloul
- Réalisation du livrable 1 pour le 18/02
- Recherches pour comprendre comment créer un chatbot

Mars

- Réalisation du livrable 2 pour le 25/03
- Convertir les phrases en langage naturel en requêtes
- Création de la structure de base du chatbot

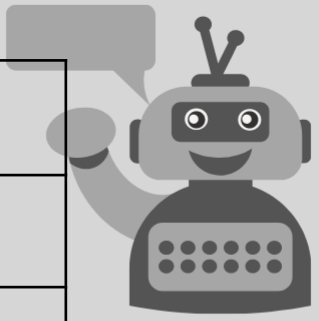
Avril

- Création de l'emploi du temps
- Relier le chatbot à l'emploi du temps
- Création de l'interface graphique
- Enrichissement de la base de données du chatbot
- Mise en place des fonctionnalités spécifiques à notre chatbot

Mai

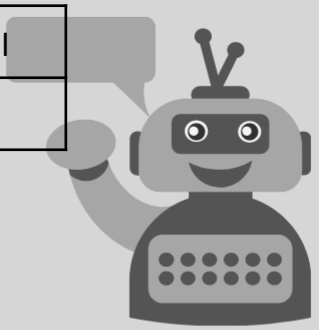
- Finalisation de l'application
- Ajouts de modules supplémentaires
- Ajout de modules (suggestions de requêtes, voix au chatbot, design de
- personnage, questions sur l'humeur au cours de la journée)
- Tests
- Réglage des problèmes éventuelles
- Réalisation du livrable 3 pour le 20 mai
- Soutenance

Plan de charge prévisionnel :



	Berton Margot	**Debono Juliette**	**Kacer Inès**	**Marjollet Iris**
Gestion de projet				
Réunions hebdomadaires avec le tuteur	30 min/semaine	30 min/semaine	30 min/semaine	30 min/semaine
Réunions des membres du groupe	1 h/semaine	1 h/semaine	1 h/semaine	1 h/semaine
Rédaction	30 min/semaine	30 min/semaine	30 min/semaine	30 min/semaine
Conception				
Définition des classes, méthodes	/	1h	/	/
Auto-formation	3h	3h	3h	3h
Implémentation				
Implémentation du chatbot	2h	2h	2h	2h
Implémentation des fonctions python	/	2h/semaine	/	2h/semaine
Communication serveur / application	1h/semaine	/	1h/semaine	/
Implémentation de l'interface graphique	1h/semaine	1h/semaine	1h/semaine	1h/semaine
Implémentation des tests	1h au total	3h au total	1h au total	1h au total
Soutenance				

Préparation	3h au total	3h au total	3h au total	3h au total
Soutenance	1h	1h	1h	1h



Suivi d'activité :

Mars :

- **Toutes :** création de l'identité du chatbot JIMI (petite anecdote : le nom du chatbot vient de nos initiales : **J**uliette, **I**ris, **M**argot et **I**nès) : définition de ses fonctionnalités et cahier des charges ;

- **Juliette et Iris :** création de la base de données, implémentation d'une fonction permettant de récupérer les informations utiles de la requête en langage naturel (fonction info), gestion des heures (traitement des heures anglophones), implémentation des fonctions average (moyenne de temps pro ou perso libre dans la semaine), agenda et timebusy (renvoie le temps pro ou perso ou total dédié sur un laps de temps) et add (ajout un événement pro ou perso) ;

- **Inès et Margot :** appropriation de la bibliothèque OkHttpClient afin de créer une communication serveur web/application, pour finalement abandonner l'idée d'une bibliothèque et utiliser nos propres fonctions pour établir la connexion, test d'affichage d'un message envoyé par le serveur (local) sur notre émulateur de téléphone.

Avril :

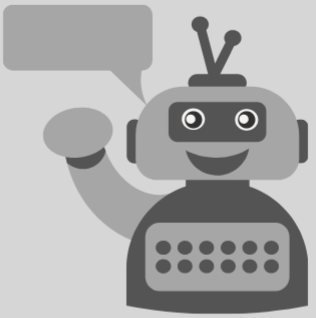
- **Juliette :** Mise en place du serveur local python avec Flask, essaie infructueux de le déployer (problème de bibliothèque python avec les serveurs de MiNET, rendant impossible un déploiement via l'association) ;

- **Iris, Inès et Margot :** travail sur l'interface de l'application (image de JIMI en haut à gauche, spécifiquement pour Inès et Iris l'affichage de bulles alternées, couleur du fond et des différents éléments affichés).

Mai :

- **Juliette :** Remarque que l'agenda était général et non propre à un utilisateur, ajout dans la base de données pour chaque évènement l'adresse MAC du téléphone pour identifier l'utilisateur de manière unique et lui permettre de retrouver ses événements et récupération de l'adresse MAc dans Android Studio ;

- **Toutes :** poursuite du travail sur l'interface (bulles alternées, musique en fond avec un bouton pause/démarrer en haut à droite, bouche de JIMI qui se modifie pour parler), tests, finir les livrables.



IV. Interprétation des résultats

Atteinte des objectifs :

Nous avons abandonné l'idée de faire un agenda sur un an et nous nous sommes limitées à une timeline d'une semaine.

Nous n'avons pas de fonctionnalité modification d'un événement : si l'utilisateur souhaite modifier un événement, il doit supprimer celui existant et ajouter un nouvel événement.

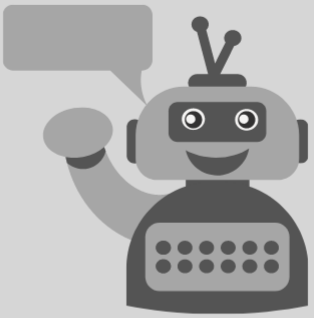
Nous souhaitions initialement créer un système de rappels journaliers afin que l'utilisateur reçoive des notifications. Par manque de temps et par volonté d'avoir une application facile à utiliser avec des fonctionnalités simples, nous avons abandonné cette idée.

En cours de projet, nous nous sommes amusées à compléter l'interface avec une musique de fond libre de droit et un design original du bot réalisé par Iris.

Problèmes rencontrés et leur solution :

Problème rencontré	Solution
Gestion de GIT et du travail sur plusieurs ordinateur, difficulté de s'habituer (bien que une fois maîtrisé c'est très utile pour avancer en groupe sur un projet)	Nous avons travaillé exclusivement à l'aide GIT pour améliorer notre prise en main, le tuteur nous a aidé et nous a expliqué le fonctionnement et nous avons relié IntelliJ à GIT afin d'utiliser les raccouris d'IntelliJ
Connection au serveur pour relier le script python et l'application	Nécessité de faire tourner en local le serveur pour pouvoir utiliser le chatbot sur l'interface graphique
Création de bulles de message pour ressembler à une messagerie	Nous avons fait appel à des étudiants de MiNET afin qu'ils nous aident. Nous avons ainsi réussi à modifier dynamiquement le layout de l'application à chaque fois qu'une requête est envoyée
Gestion de plusieurs utilisateurs sur une même base de données	Utilisation de l'adresse MAC pour identifier l'utilisateur

V. Manuel utilisateur



Fonctionnalités de l'application :

- Possibilité de créer un agenda sur une semaine

  Conseil : créer l'agenda de la semaine le lundi matin

- Faire des requêtes en anglais

- Formulation des requêtes : libre

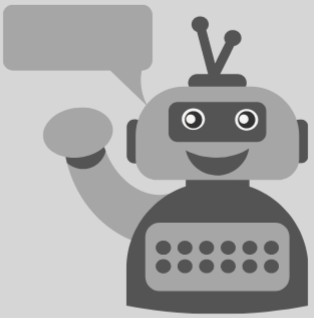
- Ajouter un événement
  - **Requête type :** add jour, heure début, heure fin, "titre", personnel/professionnel
  - **Par défaut :**
    - Si on ne précise pas de jour, le chatbot considère que l'événement a lieu aujourd'hui.
    - Si on précise uniquement la date de début, le chatbot considère que la durée de l'événement est d'une heure.
    - Si on ne précise pas de titre, l'événement s'appellera "Untitled event"
    - Si on ne précise pas le type professionnel ou personnel, le chatbot le considère comme "Undefined"

- Supprimer un événement
  - **Requête type :**
    - Delete jour, heure de début.
    - Delete titre : dans ce cas, le chatbot supprimera tous les événements qui ont ce titre.

- Demander notre emploi du temps sur une plage horaire
  - **Requête type :**
    - show me my day
    - show me my agenda of Tuesday
    - show me my day from 3pm to 7pm

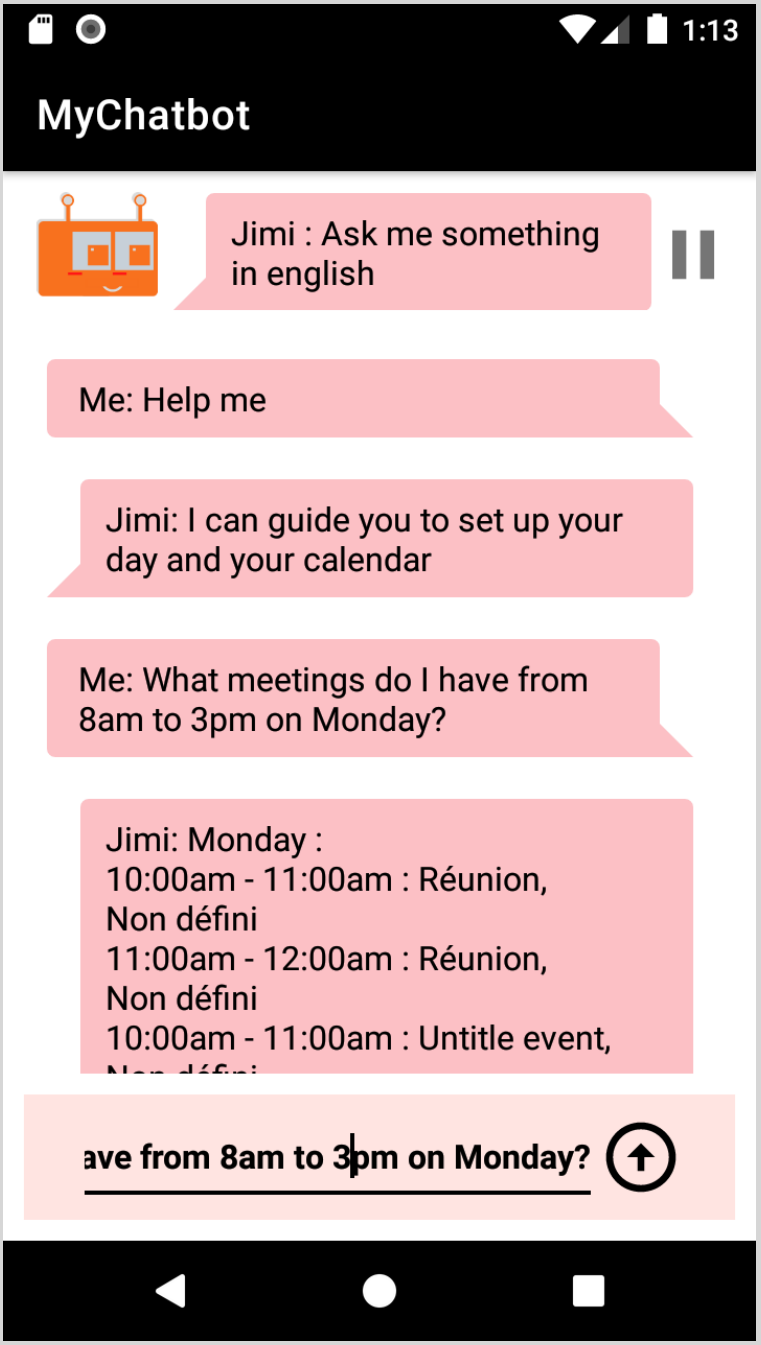
- Demander combien de temps on est occupé dans la journée :

- **Requête type :** how long am I busy today professionally / personally

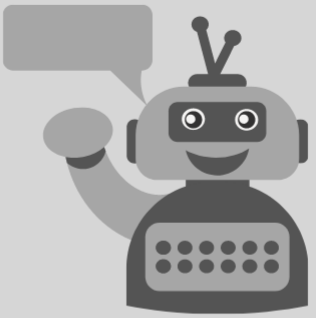


❖ Fonctionnalités supplémentaires :
- **Requêtes :** I love you / Hello et bien d'autres …
- Activer / Désactiver la musique

VI. Capture d'écran de l'application



VII. Tests



Interface Graphique :

Durant notre projet, il a été nécessaire de tester toutes les fonctionnalités relatives à l'application. L'interface graphique sur Android Studio n'a pas nécessité de réels tests, si ce n'est l'utilisation d'un émulateur de téléphone android pour regarder le fonctionnement de tous les paramètres : mise en forme, utilisation des boutons, connexion au serveur. L'émulateur utilisé est un Galaxy Nexus avec un API 26, est utilisant Android 8.0.

Chatbot détection de la catégorie :

Nous avons aussi testé le programme qui identifie la catégorie de la question, en posant différentes questions et en vérifiant que la catégorie est bien celle supposée.
Les tests du fichier python answer_chatbot.py est le fichier tests_chatbot.py

Détection des informations dans la phrase et application de la requête :

Ce qui a surtout nécessité des tests sont les fonctions python qui extraient les informations dans la phrase de l'utilisateur et accèdent à la base de données.
Pour cela, nous avons testé au fur et à mesure chaque fonction qui ajoute/accède/supprime à un événement dans la base de données. À chaque test nous vérifions que toutes les informations sont bien entrées dans la base de données et sont à jour.
Les tests du fichier python traitement.py est le fichier tests_traitement.py