Relazione di tirocinio curricolare

"Interazione automatica uomo-macchina via chat"
Tirocinio svolto dal 10 marzo 2020 al 18 maggio 2020
presso l'azienda Lab51

Tirociniante: Asia Lucchi
Tutor didattico: Damiana Lazzaro
Tutur aziendale: Tommaso Dionigi

May 19, 2020

1 Introduzione

L'obiettivo del tirocinio è quello di produrre un chatbot che permetta ai dipendenti di un'azienda di richiedere permessi e ferie, notifichi l'amministratore e in caso di approvazione da parte di quest'ultimo registri eventi in corrispondenza dei permessi in un calendario.
Una delle specifiche fondamentali è la comprensione da parte del chatbot del maggior numero di espressioni comunemente usate per richiedere permessi possibile, ad esempio il riconoscimento di parti della giornata come "mattina" o di quantità di tempo come "una settimana.
Altre funzionalità del chatbot sono quelle di redigere riepiloghi, dare la possibilità di cancellare permessi, comprendere e rispondere con messaggi vocali, avere un'interfaccia responsibe su dispotivi mobili.

A causa dell'emergenza COVID-19 non è stato possibile svolgere il tirocinio nella sede dell'azienda, perciò è stato portato a termine lavorando in modalità smart-working con occasionali confronti con i tutor aziendali via messaggio o meeting in videocall.

Nelle figure sottostanti è riportato il funzionamento in linea di massima del bot, ovvero la richiesta di un permesso al bot e l'inoltro di tale richiesta all'amministratore con i bottoni per scegliere se approvarla o meno.

(a) Chat del dipendente

(b) Chat dell'amminstratore

Figure 1: Funzionamento in linea di massima del bot

2 Tecnologie

Per interpretare le richieste inviate dall'utente il chatbot utilizza Dialogflow, una piattaforma messa a disposizione da Google.
L'interfaccia utente è stata realizzata attraverso un bot Telegram (@ferie_bot).
Come calendario è stato utilizzato Google Calendar attraverso la Google Calendar API.

L'applicativo ha bisogno di registrare una quantità molto scarsa di informazioni perciò non è stato necessario l'ausilio di un database, ma si sono semplicente salvate su un file json (dopo aver verificato che la lettura e scrittura di tale file non comprometessero le permformance dell'applicativo).

Ho scelto di sviluppare con il linguaggio Javascript, usando in particolare le librierie fornite da Node.js in quanto ho iniziato a sviluppare usando l'Inline Editor di Dialogflow (che mette a disposizione unicamente questo linguaggio) e quando sono passata a un Webhook ho deciso di mantenerlo.

Il sistema operativo non è stato in alcun modo rilevante nello sviluppo in quanto l'applicativo è interamente fruibile via web.

Per implementare le tecnologie Text-To-Speech e Speech-To-Text sono state usate le librerie messe a disposizione da Google.

3 Attività

3.1 Architettura generale

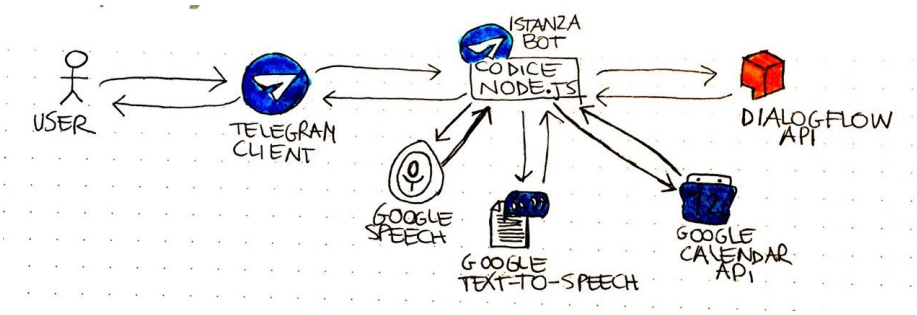

Figure 2: Architettura generale dell'applicativo

Come si può notare dalla figura 2, il funzionamento in linea di massima dell'applicativo prevede che l'user utilizzi un client Telegram per rivolgersi ad un bot e somministrargli le proprie richieste. Il bot è istanziato attraverso la libreria node-telegram-bot-api ed è colui che gestisce tutti i messaggi dell'utente, innanzitutto trasformando i messaggi vocali in testo attraverso la libreria Speech di Google. Il messaggio di testo viene inviato a Dialogflow per l'interpretazione ed in base all'intento individuato si invia il messaggio di risposta al Telegram Client, usando la libreria Text-To-Speech di Google per trasformarlo in vocale nel caso fosse attivita la modalità audio. Se l'intento identifica un'approvazione da parte dell'admin o una cancellazione da parte dell'utente, si utilizza la Google Calendar API per riportare le modifiche necessarie al calendario.

Durante lo sviluppo dell'applicativo la struttura ha subito vari cambiamenti, infatti non l'ho individuata fin da subito trovandomi a lavorare con elementi che non conoscevo, ho iniziato facendo semplici test per sperimentare il funzionamento della console Dialogflow e aggiunto componenti man mano che il progetto prendeva forma. La configurazione finale è stata utilizzata solo negli stadi conclusivi del progetto, inizialmente infatti si usava la funzionalità Integrations di Dialogflow, per cui il client Telegram comunicava in modo diretto con Dialogflow ed era quest'ultimo a comunicare a sua volta con l'istanza del Telegram Bot (che si occupava solo di mandare messaggi). Quando si è deciso di implementare la modalità Speech-To-Text è stato necessario interporre l'istanza del bot fra il Client e Dialogflow in modo che analizzasse i messaggi vocali, li trasformasse in testo usando la libreria Speech di Google e solo successivamente li inoltrasse a Dialogflow per interpretarli.
Inoltre, nella configurazione precedente si creava una fulfillment per Dialogflow facendo un Webhook che comunicava con un applicazione server in esecuzione sul mio pc personale la quale esponeva una porta attraverso il servizio ngrok. In questa configurazione non era possibile settare un Webhook o fare polling dall'istanza del bot Telegram perchè le richieste di nuovi messsaggi entravano in conflitto con quelle fatte da Dialogflow.
Quando si è passati alla struttura finale invece è stato attivato il polling dall'istanza del bot Telegram e non è più stato necessario l'utilizzo del fulfillment con Webhook su Dialogflow in quanto il comportamento del bot in base agli intenti viene gestito direttamente dal processo che invia la query a Dialogflow.

3.2 Interpretazione e Dialogflow

Buona parte del lavoro è stato svolto sull'interpretazione corretta del maggior numero di richieste possibili. Inizialmente sono state implementate quelle più semplici, ad esempio "Vorrei un permesso il 21 maggio dalle 9 alle 10" o "Vorrei delle ferie dal 21 al 24 maggio". Poi sono state aggiunte varie modalità con cui fosse possibile fare una richiesta, alcuni esempi:

- Inserendo solamente una data (es. "Vorrei un permesso il 21 maggio") l'applicativo chiede "Vuoi specificare un orario?"
- Inserendo un giorno e un momento della giornata (es. "Vorrei fare una richiesta di permesso per domani alle 10") l'applicativo chiede "Di quanto tempo?"
- Inserendo una data e un periodo di tempo (es. "Vorrei un permesso di 3 ore il 21 maggio") l'applicativo chiede "Da che ora?"
- Inserendo un periodo di tempo di più giorni (es. "Vorrei 3 giorni di permesso") l'applicativo chiede "Da che giorno?"
- Inserendo un periodo di tempo in ore (es. "Vorrei 2 ore di permesso") l'applicativo chiede prima il giorno e poi l'orario di inizio

In generale, in base alle informazioni fornitegli inizialmente dall'utente il bot pone varie domande per ottenere tutto ciò che gli serve al fine di creare un evento sul calendario, ovvero un momento di inizio ed uno di fine del permesso. Ciò è possibile grazie all'utilizzo dei contesti di Dialogflow, infatti in base a ciò che l'applicativo ritiene che manchi nella richiesta invia una certa domanda all'utente e imposta un contesto su Dialogflow per suggerigli in che intento è probabile che ricada la risposta successiva. I contesti sono fondamentali anche quando la riposta dell'utente è un semplice "Si" o "No", infatti diversi intenti possono avere in ingresso queste risposte (ad esempio "Vuoi inserire l'orario?" o "Confermi di voler richidere un permesso il 21 maggio?") ed è importante capire a che punto della conversazione è inserita tale risposta.

Un'altra parte importante è stata la registrazione in modo sensato di questi periodi di tempo interpretati sul calendario. Si sono attuati vari controlli:

- Che il periodo di tempo richiesto non sia nel passato
- Che il periodo di tempo richiesto contenga giorni festivi, nel caso lo siano solo alcuni il permesso viene comunque registrato nei giorni feriali altrimenti viene impedito di procedere con la richiesta
- Che il periodo di tempo richiesto non contenga dei momenti in cui si era già registrato un permesso. In tal caso se il periodo già registrato è uguale o più ampio di quello richiesto non può procedere con la richiesta, se invece il periodo già registrato è non comprende tutto il periodo richiesto, si espande il permesso già registato.

3.3 File data.json

Ho ristrutturato varie volte il file JSON per mantenere in memoria solo le informazioni strettamente necessarie in ogni momento, la struttura finale è esemplificata nella Figura 3 e contiene:

- actualRequests: mantiene le richieste dei vari utenti quando non sono ancora confermate e in certi casi incomplete (ad es. manca l'orario)
- requests: mantiene le richieste quando sono state inviate agli amministratori e sono in attesa di approvazione, ogni richiesta contiene informazioni sull'utente, il periodo e i messaggi di riferimento mandati agli admin
- admins: un array per registrare i chat_id degli amministratori
- colors: ad ogni chat_id è assegnato un numero che indica di che colore sono i suoi permessi sul calendario Google
- TTS: ad ogni chat_id è assegnato un booleano che indica se è attiva la modalità di risposta tramite messaggio vocale
- sessionIds: ad ogni chat_id è assegnato un codice che indica la sessione di DialogFlow che corrisponde a quella conversazione

```
{
  "actualRequests": {
    "user642249337": {
      "date": {
        "startDateTime": "2020-05-20T12:00:00.000Z",
        "endDateTime": "2020-05-20T12:00:00.000Z"
      },
      "quanto": 10800000,
      "mode": "create"
    }
  },
  "requests": [
    {
      "mode": "create",
      "date": {
        "endDateTime": "2020-02-03T12:00:00+02:00",
        "startDateTime": "2020-02-03T09:00:00+02:00"
      },
      "user": {
        "id": 642249337,
        "first_name": "Asia",
        "last_name": "Lucchi"
      },
      "id": 3,
      "adminId": 1,
      "sentMessages": [
        {
          "chat_id": 642249337,
          "message_id": 12815
        }
      ]
    }
  ],
```


(a) ActualRequests, Requests

```
  "admins": [
    642249337
  ],
  "colors": {
    "642249337": 1,
    "1016040343": 11
  },
  "tts": {
    "642249337": false,
    "1016040343": false
  },
  "sessionIds": {
    "642249337": "20a08569-c422-4b6c-a836-545692d844c8",
    "1016040343": "904f0523-a2a5-441d-9287-a77352601142"
  },
  "fallbacks": {
    "20a08569-c422-4b6c-a836-545692d844c8": 0,
    "904f0523-a2a5-441d-9287-a77352601142": 0
  }
}
```


(b) admins, colors, TTS, sessionIds, fallbacks

Figure 3: Contenuto file data.json

– fallbacks: per ogni sessionIds si indica quanti intenti di fallback (in cui Dialogflow non ha compreso cosa è stato detto) consecutivi ci sono stati in una certa conversazione, se sono più di 3 il messaggio inviato all'utente cambia indicando cosa può scrivere per farsi compredere meglio dal bot.

3.4 Altro

Ho fatto uso di npm che non avevo mai sfruttato prima per integrare varie librerie. Ho studiato vari construtti di javascript, in particolare ho usato molto le funzioni di callback in quando erano quelle presenti inizialmente nell'Editor Inline di Dialogflow. Ho anche appreso il funzionamento di Promise, sebbene non ci sia stato tempo di reificare l'intero codice per utilizzarlo al posto delle funzioni di callback. Ho sporadicamente usato async e await, in particolare per mandare richieste a Dialogflow ed a i client TTS e STT ed essere certa di ricevere la risposta prima di procedere.

Per evitare errori di polling nel caso vengano lanciate più istanze del server contemporaneamente che quindi richiedono entrambe i messaggi inviati al bot, ho predisposto un sistema per chiudere immediatamente il server messo in esecuzione più recentemente.

Per poter inviare query ai client Dialogflow, STT e TTS ho dovuto autenti-

carmi come owner del progetto Google sui cui sono registrati anche gli intenti Dialogflow e includere un file con le credenziali nel codice per permettergli di utilizzare i servizi Google.
Per quanto riguarda l'accesso al calendario è stato necessario scaricare e includere sia un file con le credenziali per accedere con un mio profilo che un token che autorizzi l'applicativo a leggere e scrivere su uno specifico calendario del profilo. Ogni volta che si accede al calendario per fare un listing degli eventi, crearne o eliminarne alcuni è necessario utilizzare sia il file con le credenziali sia il token. Di seguito alcuni esempi di permessi registrati sul calendario, quelli di un utente tutti dello stesso colore:

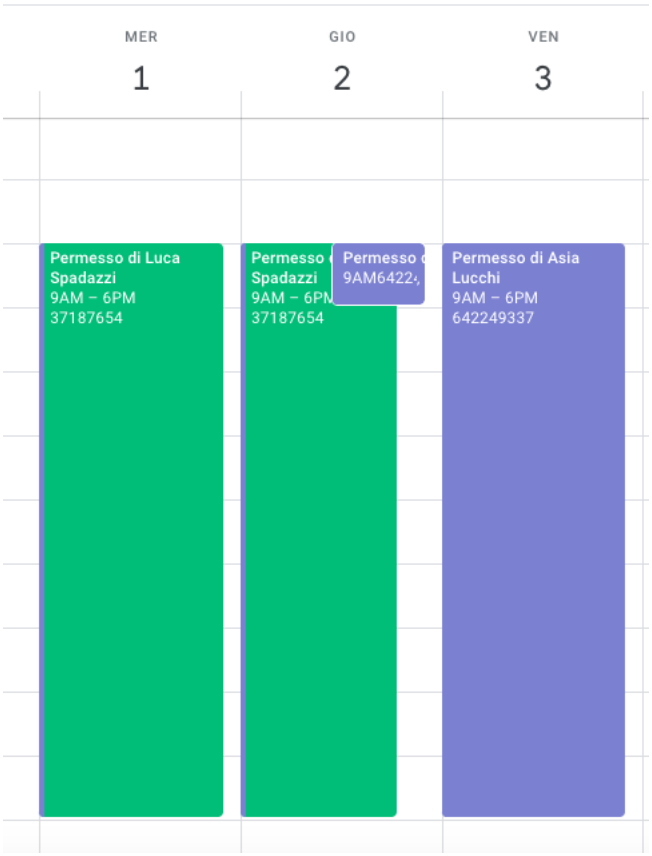

Figure 4: Esempi di eventi registrati sul calendario nei primi giorni di Luglio

Accedendo al calendario è anche possibile produrre un riepilogo dei permessi per un qualcunque mese, che appare nella seguente forma ed è richiedibile solo dall'admin.
Infine ho usato la libreria pkg che permette di produrre eseguibili per vari sis-

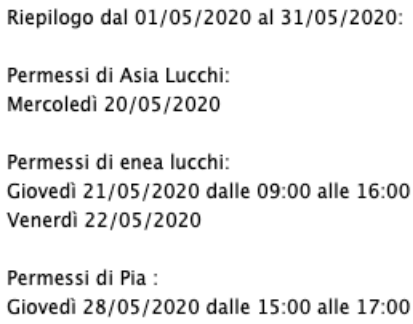

Figure 5: Esempi di riepilogo dei permessi di Maggio

temi operativi (Windows, Macos, Linux). Per includere i vari file usati durante lo sviluppo ho dovuto distinguere quelli a cui accedevo solamente in lettura (principalmente token o file con le credenziali) da quelli su cui scrivevo (il file di data, gli audio di input e output). Ho predisposto la creazione di una cartella contenente i file in scrittura al momento del lancio dell'eseguibile.

4 Conclusioni

L'esperienza è stata molto stimolante e ho imparato tante nuove nozioni, soprattutto per quanto riguarda Node.js che non avevo mai utilizzato prima ed è sicuramente un tecnologia ampiamente sfruttata al momento.
In generale sono riuscita a usare diverse API (principalmente Dialogflow, STT, TTS, Telegram Bot, Google Calendar) che in precedenza non conoscevo assolutamente basandomi sulla documentazione fornita, gli esempi reperibili sul web e un minimo supporto da parte dell'azienda.
Questo è stato un grande cambiamento per me che non avevo mai programmato usando strumenti il cui funzionamento non mi era stato mostrato in precedenza dai professori universitari.
Ho acquisito una certa autonomia nell'apprendimento e l'utilizzo di tecnologie a me sconosciute anche se, ripeto, i dipendente dell'azienda erano sempre pronti a venirmi in aiuto nel caso fossi in difficoltà.

L'ambito del progetto sviluppato era stato discusso in precedenza con l'azienda dato il mio interesse nella Data Science e sopratutto l'interpetazini dei dati.
Sono stata soddisfatta del progetto assegnatomi in quanto riguardante il riconoscimento del linguaggio naturale ma anche abbastanza "pratico" da permettermi di vedere i risulati del mio lavoro usando la chatbot su Telegram, di conoscere un linguaggio nuovo e delle API sfruttabili in tanti ambiti.
Ritengo che la mole di lavoro che ho svolto sia adeguata alle finalità di un tirocinio curricolare di una Laurea triennale in Ingegneria e Scienze Informatiche

e che gli strumenti fornitomi dall'azienda siano stati soddisfacenti.

I tutor aziendali che mi sono stati assegnati sono sempre stati disponibili e anche se la modalità a distanza non ci ha sempre permesso di dialogare in tempo reale hanno sempre risposto ai miei quesiti in maniera esaustiva e in tempi brevi.

Il fatto di dover svolgere il tirocinio da remoto è stato sicuramente un disagio sopratuttuo nella fase iniziale in cui ero un po' spaesata e mi avrebbe giovato la presenza di un team che mi potesse affiancare anche per i dubbi più banali. Proseguendo nel lavoro sono entrata sempre più in confidenza con le tecnologie che mi sono ritrovata ad utilizzare e ho apprezzato il livello di indipendenza che questa modalità di lavoro mi ha fornito, soprattutto nella flessibilità degli orari che mi ha permesso di seguire le lezioni in contemporanea al tirocinio.