Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

Progettazione e sviluppo di un'applicazione per la gestione degli appuntamenti tra professionisti e clienti

Relatore:
Chiar.mo Prof.
Luciano Bononi

Presentata da:
Davide Bonaiuti

Sessione II
Anno Accademico 2015-2016

Introduzione

Tutti coloro che lavorano su appuntamento hanno una clientela da gestire. Per farlo, possono avvalersi di due opzioni: o utilizzano una buona parte del loro tempo per rispondere direttamente alle chiamate e ai messaggi dei clienti, oppure assumono una persona che si faccia carico personalmente dell'agenda del negozio, piuttosto che dello studio, a seconda dei casi.

Gli aspetti negativi di queste due scelte non mancano. Nel primo caso, il diretto interessato può trovarsi interrotto da una chiamata nel pieno di un appuntamento. Il cliente lì presente deve attendere pazientemente che la conversazione finisca, ma la pazienza, giustamente, inizia a vacillare nel momento in cui le chiamate diventano numerose. Oltretutto la situazione può creare disagio al professionista in prima persona. In alcuni casi, inoltre, il cliente a casa è costretto ad aspettare tempi troppo lunghi prima di ricevere una risposta. Nel secondo caso, si ha un lavoro più fluido e senza interruzioni. Questo è ovviamente un fattore positivo, tuttavia l'assunzione di un addetto a rispondere al telefono, piuttosto che a messaggi o mail, ha un costo non indifferente.

Negli anni si sono susseguite varie metodologie per la comunicazione tra cliente e professionista. Al giorno d'oggi, possiamo distinguere due principali categorie: chiamate e messaggi. Entrambe hanno i propri vantaggi e svantaggi. Per ovviare ad alcuni degli svantaggi derivanti dall'utilizzo di questi due metodi, con il supporto della tecnologia, si sono aggiunte nuove forme di comunicazione, tra cui spicca l'utilizzo di applicazioni per smartphone. Grazie a queste, ogni attività ha la possibilità di creare la propria app o, come avviene nella maggior parte dei casi, di farsela creare da un'azienda specializzata. Ciò permette al cliente di prendere autonomamente un appuntamento, senza sottrarre tempo a una persona fisica, con l'ulteriore vantaggio di una risposta immediata certa.

Anche tutto questo, però, ha generalmente dei costi elevati da sostenere e, per di più, il metodo di prenotazione viene completamente stravolto rispetto ai metodi più classici, rischiando così di creare complicazioni eccessive al cliente.

Analizzando i vari vantaggi e svantaggi dei metodi sopra citati, è stata progettata una nuova soluzione che, tramite l'utilizzo dei bot e delle app di messaggistica, permette di prendere gli appuntamenti rimanendo il più fedeli possibile ai metodi classici, come i messaggi, ma con tutti i vantaggi dei metodi più tecnologici.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Problema

Il problema classico di chi lavora su appuntamento, che da questo momento in poi viene denominato "professionista", consiste nella gestione della propria agenda, la quale, per essere svolta con cura, richiede molto tempo e dedizione. Gli esempi più immediati di queste professioni sono:

- Parrucchieri
- Barbieri
- Estetisti

a cui possiamo aggiungerne di meno scontati come:

- Dentisti
- Oculisti
- Medici di base

Nella stragrande maggioranza dei casi, tocca al professionista in prima persona prendersi carico di tutte le comunicazioni con il cliente. Questo fatto comporta un notevole dispendio di tempo e di energie, in quanto richiede di dover interrompere il lavoro ad ogni telefonata o messaggio, situazione non piacevole. Ciò si somma al disturbo arrecato al cliente servito in quel momento, il quale, dopo alcune interruzioni, può iniziare a

spazientirsi. Un altro modo di procedere del professionista, consiste nel rispondere alle richieste dei clienti solo quando è libero. In questo caso, anche se migliora l'esperienza del cliente lì presente, quella di comunicazione del cliente "a casa" viene peggiorata e non di poco, in quanto si allungano i tempi di risposta.

Una soluzione può consistere nell'impiego di una persona fisica, addetta solo alla gestione della clientela. In questo caso, il tutto avviene in modo rapido e senza alcuna interruzione del lavoro del professionista. È evidente, però, che non si tratta di una soluzione percorribile da chiunque, in quanto pagare una persona per l'intera giornata lavorativa richiede un notevole dispendio di denaro.

Oggi esiste una varietà molto ampia di metodi di comunicazione, perciò anche il cliente deve imparare a districarsi tra di essi. Si possono differenziare in due principali gruppi, i "classici" e i "tecnologici".

1.1 Metodi Classici

Tra questi metodi rientrano le chiamate e i messaggi. Entrambi sono di semplice utilizzo per il cliente, ma richiedono molto impegno da parte del professionista.

1.1.1 Chiamate

Si possono classificare come le più utilizzate in assoluto, sebbene adesso inizino a passare in secondo piano.

Professionista

Rispondere a chiamate richiede un notevole sforzo. All'arrivo di ogni chiamata egli è costretto ad interrompere il proprio lavoro, con conseguente perdita di tempo, oppure può non rispondere fino al suo termine, prediligendo la persona lì presente. Questa scelta, però, va discapito di colui che cerca di prendere un appuntamento.

Cliente

Le chiamate risultano un metodo che consente con buona probabilità di ottenere una risposta veloce: nel caso in cui il professionista interrompa il proprio lavoro per rispondere, infatti, l'attesa diventa nulla. Tuttavia possono anche risultare scomode. Questo accade ad esempio quando il professionista non è reperibile, oppure la linea risulta già occupata. Di conseguenza il cliente si trova costretto a ripetere la procedura più volte.

1.1.2 Messaggi

L'altro metodo diventato negli ultimi anni di uso quotidiano, consiste nell'invio di messaggi. Nello specifico, questa diffusione è avvenuta prima con gli SMS e ora con le app di messaggistica quali WhatsApp, Telegram, Facebook Messenger e molte altre.

Professionista

La comunicazione col cliente risulta agevolata, in quanto può permettersi di rispondere ai messaggi nei momenti a lui più comodi. L'altra faccia della medaglia lo vede costretto, tuttavia, a districarsi tra le numerose chat e piattaforme di messaggistica. A questo si aggiunge la necessità di scambiare più messaggi con ogni singola persona, allungando in modo significativo il tempo impiegato rispetto ad una telefonata.

Cliente

È un metodo comodo, in quanto l'invio di messaggi tramite un'app di messaggistica risulta oggi una delle cose più semplici e veloci, ma molto spesso l'attesa per ricevere una risposta è lunga.

1.2 Metodi Tecnologici

Le soluzioni più tecnologiche sono svariate, ma tra queste se ne distinguono due principali: le applicazioni su misura per le singole attività o l'iscrizione di queste ultime all'interno di piattaforme capaci di ospitare tutte le attività di una specifica professione.

1.2.1 Applicazioni su misura

Con "applicazioni su misura" ci si riferisce ad app create appositamente per la gestione dell'agenda del singolo professionista. Il cliente può consultare l'app in modo autonomo, visualizzando gli orari degli appuntamenti disponibili e prenotando così, tra questi, quello a lui più agevole. Tutto questo avviene senza che il professionista debba preoccuparsi di niente, infatti si ritrova sul calendario tutti gli appuntamenti in modo automatico. Ne consegue un notevole risparmio di tempo ed energie.

Bisogna tuttavia prestare attenzione a non complicare troppo il processo di prenotazione, altrimenti rischiano di diminuire i clienti a favore di questi metodi. I paradigmi canonici di comunicazione vengono infatti sconvolti. Il cliente deve scaricare e installare l'applicazione della specifica attività sul proprio smartphone. Attraverso un'interfaccia, deve seguire poi una serie di passaggi necessari a portare a completamento la prenotazione. Per i meno esperti, questi meccanismi possono essere complicati. Inoltre si deve tenere conto del fatto che il cliente potrebbe frequentare più di un'attività che ne fa uso, dovendo installare di conseguenza un'applicazione diversa per ognuna. Questo comporta due rischi: il primo è la ridondanza, perché le applicazioni hanno tutte principalmente il medesimo fine, mentre il secondo consiste nella non omogeneità, in quanto app diverse possono avere interfacce differenti, rischiando così di confondere il cliente tra l'utilizzo di una e l'altra. Per la realizzazione dell'applicazione, è necessario rivolgersi ad aziende specializzate. Economicamente si parla di costi iniziali abbastanza elevati, che vanno ad aggiungersi a quelli del mantenimento periodico, in genere molto contenuti.

1.2.2 Piattaforme

Esistono delle piattaforme capaci di ospitare tutte le attività di una medesima categoria. Il cliente può trovare quelle di interesse tramite l'app della piattaforma. In questo modo, il professionista ha gli stessi vantaggi che aveva nel caso visto sopra, pur diminuendo drasticamente la possibilità di personalizzazioni. L'aspetto nuovo di questa scelta risiede nell'aggiunta di visibilità tra gli utenti della piattaforma. Inoltre il vantaggio economico non è indifferente. Generalmente queste piattaforme, infatti, richiedono una quota periodica molto bassa e il risparmio è notevole rispetto all'applicazione personaliz-

zata vista in precedenza. Per il cliente i passi da seguire sono analoghi alle applicazioni su misura, anche se, in questo, caso non deve più installare l'app dell'attività, bensì quella della piattaforma.

Capitolo 2

Soluzione

Il metodo di comunicazione attualmente più immediato, che risulta molte volte il preferito dei clienti, è l'invio di messaggi tramite una delle tante app di messaggistica. Nell'elaborazione di una soluzione agli svantaggi precedentemente esposti, è sensato quindi fare in modo che i clienti continuino ad utilizzare i messaggi, evitando di cambiare loro un processo di prenotazione molto intuitivo. La sfida è riuscire, però, ad evitare loro l'inutile tempo d'attesa di una risposta.

Si deve inoltre fare in modo d'eliminare gli svantaggi che con questo metodo si riversano sul professionista. Il tempo che perde nel rispondere ai messaggi, infatti, non è indifferente. Al fattore tempo si aggiungono le complicazioni derivanti dalla comunicazione contemporanea con clienti diversi, talvolta anche su più applicazioni di messaggistica.

2.1 Cos'è un bot?

> Il bot (abbreviazione di robot) in terminologia informatica in generale è un programma che accede alla rete attraverso lo stesso tipo di canali utilizzati dagli utenti umani (per esempio che accede alle pagine Web, invia messaggi in una chat, si muove nei videogiochi, e così via). - *Wikipedia*[1]

Nella soluzione esposta in seguito, il bot non è nient'altro che un programma capace di accedere alle chat delle app di messaggistica. Esso capisce quello che vuole comunicargli

[1] https://it.wikipedia.org/wiki/Bot

l'utente e gli fornisce una risposta. Più nello specifico, nel caso qui analizzato, è capace di "leggere" il messaggio del cliente e di individuare le date di un possibile appuntamento, rispondendogli di conseguenza.

2.2 Chat bot

Questo metodo consiste nella creazione di un bot per le applicazioni di messaggistica, capace di rispondere autonomamente nelle chat dei clienti. I paradigmi canonici di comunicazione per il cliente vengono così mantenuti e, al contempo, diminuisce la mole di lavoro del professionista.

Professionista

Inizia con l'iscrizione online, durante la quale deve fornire i vari dati dell'attività e collegare il proprio calendario lavorativo. Da questo momento in poi, affinché si abbia un corretto funzionamento, tutti gli appuntamenti vanno gestiti su quel calendario. Alla fine dell'iscrizione, vengono creati in automatico i bot per le varie app di messaggistica, che devono rappresentare l'attività in questione all'interno della app. Con questa soluzione, tutto viene demandato al bot e il professionista non deve fare altro che controllare i propri appuntamenti di volta in volta segnati automaticamente sul calendario. Naturalmente, nel momento in cui il professionista decida d'intraprendere questa strada, è necessario che sponsorizzi ai propri clienti il servizio loro offerto, in modo che possano venirne a conoscenza. Economicamente risulta vantaggioso, in quanto richiede un costo molto ridotto.

Cliente

Anziché dover salvare un numero di telefono in rubrica, al cliente basta semplicemente aprire un'app di messaggistica e cercare l'attività inserendone il nome nel campo di ricerca, oppure l'username del bot fornitogli dal professionista. In questo modo, inizia a chattare come faceva in precedenza, quando comunicava in modo diretto con il professionista. Non cambia assolutamente niente, anzi si riscontra un notevole miglioramento sul

tempo di risposta. Le risposte arrivano in pochi secondi, senza più inutili attese. Inoltre i clienti non devono installare nessuna nuova applicazione, ma semplicemente usare un'app di messaggistica che nella stragrande maggioranza dei casi è già presente sullo smartphone.

2.3 Confronto

	Velocità ricezione risposta	**Facilità di interazione**
Chiamate	Veloce	Molto buona
Messaggi	Lenta	Ottima
Operatore dedicato	Molto veloce	-
App su misura	Istantanea	Sufficiente
Piattaforme	Istantanea	Sufficiente
Chat bot	Istantanea	Ottima

Tabella 2.1: Confronto tra i metodi di comunicazione dal punto di vista del cliente

	Impiego di tempo per rispondere	**Costo**
Chiamate	Molto elevato	Nullo
Messaggi	Molto elevato	Nullo
Operatore dedicato	Nullo	Molto elevato
App su misura	Nullo	Elevato
Piattaforme	Nullo	Molto contenuto
Chat bot	Nullo	Molto contenuto

Tabella 2.2: Confronto tra i metodi di comunicazione dal punto di vista del professionista

Capitolo 3

Progettazione

Durante la progettazione, si sono dovute fare alcune scelte, tra le quali l'utilizzo di Telegram come app di messaggistica, di Google Calendar come agenda del professionista e di Google Sign-In per l'iscrizione dell'attività. Viene utilizzato inoltre Google App Engine come infrastruttura per la realizzazione del back end.

3.1 Requisiti e limitazioni

3.1.1 App di messaggistica

Al momento, non tutte le applicazioni di messaggistica supportano i bot. Supportare i bot, significa mettere a disposizione delle API che permettano agli sviluppatori di creare applicazioni capaci di interagire all'interno delle chat. Tra le principali app di messaggistica che supportano i bot, rientrano Telegram e Facebook Messenger, mentre altre importanti come WhatsApp no.

In questo lavoro, viene utilizzato solo Telegram, anche se in futuro è possibile che vengano supportate anche altre applicazioni con poco lavoro aggiuntivo.

3.1.2 Intelligenza del bot

Se il bot fosse capace di rispondere in tutti casi come farebbe l'operatore umano, si tratterebbe della soluzione perfetta. Ovviamente questo non è possibile, infatti il pro-

gramma addetto ad estrapolare le informazioni dal testo dell'utente, che nel nostro caso sono le date, pone alcuni limiti. È probabile che messaggi troppo contorti e complicati non vengano capiti. Questo può succedere anche a seguito di eventuali errori di battitura.

3.1.3 Calendario

È necessario che il professionista utilizzi Google Calendar, in quanto, avendo le API, permette al bot di accedere al calendario per conoscere gli altri appuntamenti. Questa soluzione è preferibile rispetto alla creazione di un calendario proprietario, in quanto il professionista non necessita di installare nessuna nuova applicazione, bensì gli basta utilizzare l'app di Google, che in molti casi è già installata su smartphone e tablet.

3.2 Tecnologie utilizzate

3.2.1 Telegram bot API

Queste API permettono di connettere il bot a Telegram. Un bot Telegram non è altro che un account speciale, al quale non è collegato nessun numero di telefono. Più nello specifico, si tratta di un'interfaccia per il codice che gira su un server. In pratica è un'applicazione che gira all'interno di Telegram, con la quale l'utente può interagire mandando dei messaggi. Per controllare il bot, si usano delle richieste HTTPS verso le bot API. Tutti i messaggi vengono passati al programma sul server, tramite un server intermediario che gestisce tutta la comunicazione e il criptaggio. La soluzione descritta fa uso di queste API per ricevere i messaggi dagli utenti e fornire loro una risposta.

3.2.2 Google App Engine

Si tratta di una piattaforma utilizzata per realizzare web app e back end scalabili. Inoltre offre dei servizi già integrati e l'accesso rapido ad alcune API di altri servizi Google, tra cui NoSQL Datastore. Quest'ultimo viene utilizzato per salvare i dati degli account e delle attività inseriti dal professionista. App Engine scala l'applicazione automaticamente a seconda del traffico. Questo permette di pagare solo per le risorse usate e

garantisce così dei costi contenuti. Inoltre, non è necessario né mantenere né aggiornare alcun server: basta caricare il codice e Google si prende cura del resto.

3.2.3 Google Sign-In e Calendar API

È un sistema di autenticazione sicuro e veloce, che riduce il numero di passaggi necessari ad effettuare il sign in, permettendo all'utente di registrarsi tramite il proprio account Google. Consente, inoltre, un accesso sicuro e rapido ai servizi Google, come Calendar. Per velocizzare il processo di autenticazione del professionista, viene quindi utilizzato Google Sign-In. Questo consente un rapido accesso sia alle informazioni basilari, sia, previo consenso da parte del professionista, al calendario Google. Il bot può così accedere, tramite le API, all'agenda del professionista.

3.3 Struttura

3.3.1 Iscrizione

Registrazione dell'attività

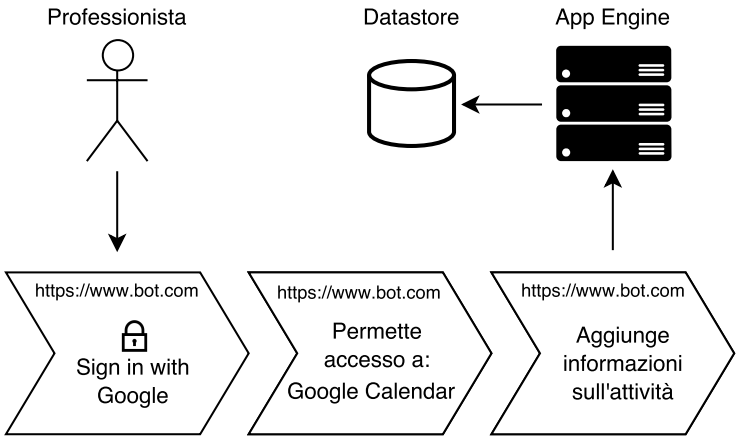

Figura 3.1: Registrazione dell'attività

Il professionista registra la propria attività tramite un front end web. Per fare ciò, deve iscriversi effettuando il sign in tramite il proprio account Google di lavoro (per

intendersi, quello in cui vuole che vengano segnati gli appuntamenti). Questo è possibile grazie alla piattaforma Google Sign-In, che permette inoltre di richiedere l'utilizzo di Google Calendar direttamente durante la fase di registrazione. Dopo avere dato il consenso, il professionista compila un form con le altre informazioni sull'attività. Tra queste rientrano il nome, gli orari di apertura e le preferenze sull'agenda, come gli orari delle pause, la durata degli appuntamenti e, infine, il numero massimo di appuntamenti sincroni. Tutte queste informazioni permettono al bot di adattarsi al meglio ad ogni singola attività.

Al termine del riempimento del form, tutti i dati inseriti e lo `one_time_code` generato da Google Sign-In vengono passati all'applicazione su App Engine. Arrivato al server, lo `one_time_code` viene convalidato tramite Google Sign-In e, nel caso sia andato tutto a buon fine, vengono ritornate le credenziali, da utilizzare per accedere alle API di Google, che in seguito vengono salvate nel Datastore insieme alle altre informazioni dell'attività.

Creazione del bot

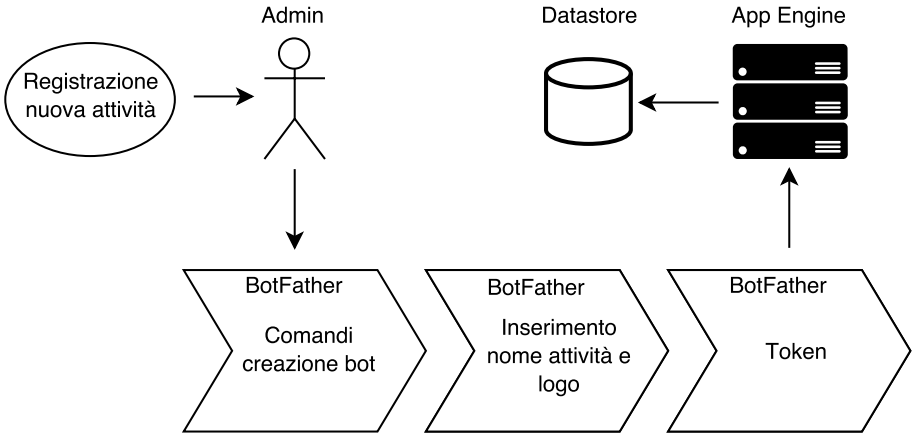

Figura 3.2: Creazione del bot

Una volta terminato il processo di registrazione, il professionista deve attendere un po' di tempo prima di vedere il proprio bot operativo su Telegram. Infatti la creazione di un bot Telegram non può essere fatta in modo automatico, bensì da una persona fisica. In questo caso, un addetto, tramite Telegram, deve inserire dei comandi in una chat con

un altro bot chiamato BotFather. Questo bot è predisposto per il solo compito di crearne altri. Durante la creazione, vengono richieste alcune informazioni, tra cui il nome (che, in questo caso, è quello dell'attività), l'username e un'eventuale immagine profilo, a cui si può fare corrispondere il logo dell'attività. Inoltre, viene impostato un URL al quale inoltrare i messaggi. In questo modo, ogni volta che ci sono dei nuovi messaggi per il bot, Telegram si preoccupa di inviare una HTTPS POST request all'URL specificato, che contenga il messaggio. Ad ogni attività corrisponde un bot diverso, perciò ad ognuno viene associato un URL univoco, così da differenziare le attività a cui appartengono le varie richieste al momento della ricezione sul server.

Al termine del processo di creazione, il bot è operativo, quindi basta l'invio di un'e-mail per comunicarlo al professionista. Viene inoltre rilasciato un token che serve per l'accesso alle Telegram bot API. È necessario che venga salvato sul Datastore insieme alle informazioni sull'attività caricate in precedenza.

3.3.2 Comunicazione

Dal cliente al bot

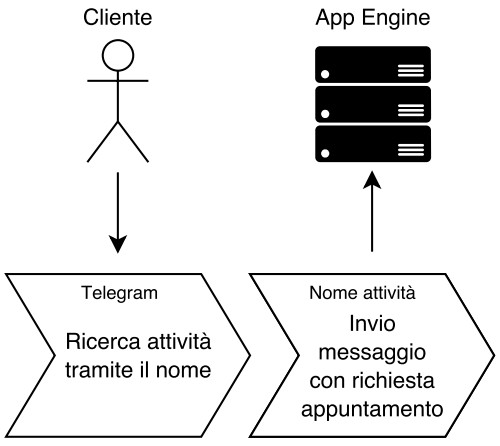

Figura 3.3: Processo di comunicazione dal cliente al bot

Trovata l'attività cercata, il cliente può aprire la chat e iniziare a scrivere. Va fatta una piccola precisazione: egli può scrivere qualsiasi cosa, ma gli unici messaggi presi in

considerazione sono quelli che contengono delle date, presumendo quindi che si tratti di una richiesta di appuntamento.

Dopo che il cliente invia un messaggio, le bot API procedono con l'inoltro al server del testo del messaggio, assieme ad altre informazioni, come i dati del cliente. Ciò avviene grazie al webhook impostato dopo la creazione del bot, in quanto, da quel momento in poi, tutti i nuovi messaggi vengono inviati all'URL specificato sotto forma di JSON.

Controllo del calendario

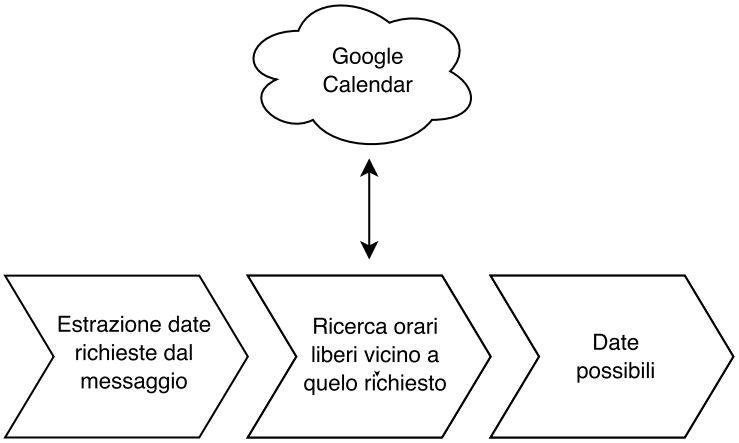

Figura 3.4: Ricerca orari liberi in base a quelli richiesti

Alla ricezione del JSON, il messaggio di testo viene estrapolato e analizzato per ricavare le date presenti nel testo. Si presume che queste siano le date nelle quali il cliente chiede un appuntamento, qui identificate come "date richieste", al plurale, in quanto possono essere anche più di una. Infatti un possibile messaggio può essere: "Ciao, hai posto lunedì o martedì?". A questo punto, il programma controlla nel calendario dell'attività gli appuntamenti disponibili vicino alle date richieste e ritorna un elenco di possibili date, qui identificate come "date possibili".

Dal bot al cliente

A questo punto, il bot deve presentare le date trovate al cliente. Per fare ciò, invia nella chat quelle prima definite "date possibili". Piuttosto che un semplice messaggio di

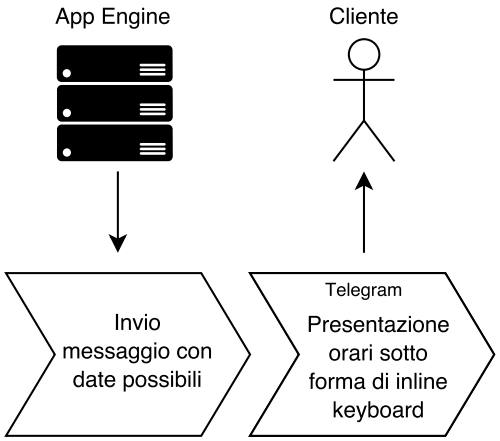

Figura 3.5: Invio messaggio con possibili orari per appuntamento

testo, si predilige in questo contesto l'invio degli appuntamenti disponibili sotto forma di inline keyboard (3.6). Si tratta di una speciale tastiera che compare sotto il messaggio, fornita di opzioni predefinite tra cui scegliere. Nel caso nessuna sia di gradimento del cliente, egli può semplicemente chiederne altre con l'invio di un altro messaggio.

Figura 3.6: Telegram inline keyboard

Conferma

A questo punto, al cliente, per prendere l'appuntamento, basta selezionare uno degli orari proposti. Dopo aver cliccato uno dei tasti dalla inline keyboard, la scelta viene subito inviata al server, come avviene per i normali messaggi. Il programma si occupa di verificare sul calendario del professionista se l'appuntamento è ancora realmente di-

sponibile e, in caso affermativo, di segnarlo, inviando poi un messaggio di conferma al cliente. A questo punto, la procedura di prenotazione è terminata e sul calendario del professionista compare l'appuntamento.

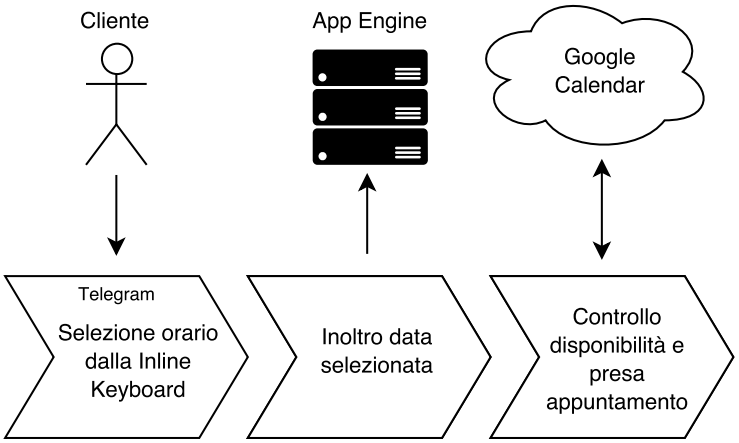

Figura 3.7: Processo di conferma dell'appuntamento

Capitolo 4

Implementazione

4.1 Google Sign-In for server-side apps

Per poter fare uso dei servizi Google per conto di un utente, anche quando egli risulta offline, è necessario utilizzare un flusso di autorizzazione, chiamato server-side:

1. L'utente effettua il sign in con Google sul client. Nello specifico, egli invia una richiesta di autorizzazione ai server di Google tramite le Javascript API.

2. Nel caso in cui l'utente non abbia ancora autorizzato l'applicazione, compare la finestra di OAuth 2.0 (4.2), che chiede di autorizzare l'app all'utilizzo dei servizi descritti. L'applicazione prevede la richiesta di autorizzazione a Google Calendar. Lo scope utilizzato, che permette l'accesso sia in lettura che in scrittura al calendario, è il seguente: `https://www.googleapis.com/auth/calendar`. Lo scope serve per indicare a Google quali servizi vengono utilizzati.

3. Se l'utente consente le autorizzazioni, vengono restituiti tre codici: `access_token`, `id_token` e il `one_time_code`.

4. Il `one_time_code` viene inviato al server.

5. Il server scambia, tramite le Google API, il `one_time_code` con l'`access_token`, `id_token` e `refresh_token` (`Credentials`), che vengono salvati nel Datastore.

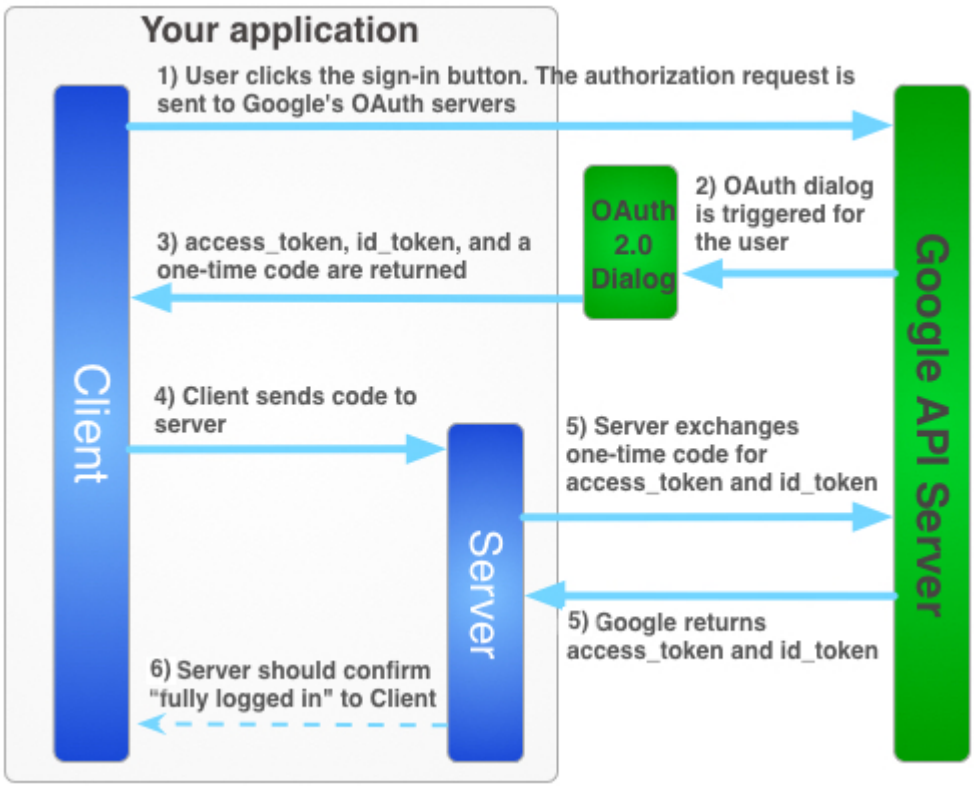

Figura 4.1: Google Sign-In for server-side apps flow[1]

Utilizzando App Engine, il processo di salvataggio viene semplificato con la creazione di un Model che ha come property la seguente classe: `oauth2client.contrib.appengine.CredentialsProperty`.

```
from google.appengine.ext import db
from oauth2client.contrib.appengine import CredentialsProperty
class CredentialsModel(db.Model):
credentials = CredentialsProperty()
```

A questo punto, il server può fare richieste ai servizi Google tramite le apposite API, indipendentemente dal client, sia che l'utente sia online o meno. Per utilizzare

[1]https://developers.google.com/identity/sign-in/web/server-side-flow

i servizi, devono essere aggiunte le credenziali a tutti gli headers delle richieste. Per fare ciò, viene usata la funzione `authorize()` della classe `Credentials`:

```
import httplib2
http_auth = credentials.authorize(httplib2.Http())
```

6. Viene inviato un messaggio al client con la conferma del login.

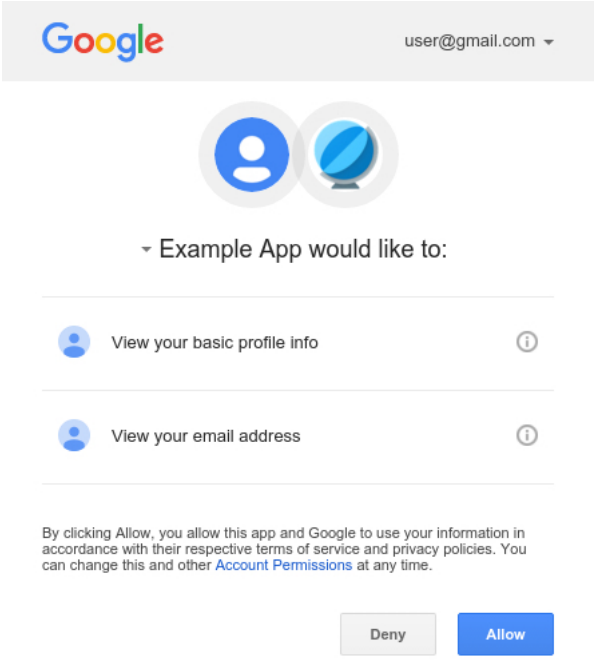

Figura 4.2: Finestra di OAuth 2.0 [2]

4.2 Telegram

4.2.1 Creazione bot

BothFather è un bot Telegram che serve per creare gli altri bot. Dopo aver aperto una chat, con esso è solo necessario digitare il comando `/newbot` per creare un nuovo bot.

[2]https://developers.google.com/accounts/images/approvaldevice.png

A questo punto, si inserisce il nome del bot (nome dell'attività) e l'username. Dopodiché viene generato un authentication token per il nuovo bot. Il token è una stringa come `"123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11"`, che serve per effettuare le richieste alle bot API.

Esitono anche altri comandi per personalizzare meglio il bot, tra cui:

- `/setdescription`: cambia la descrizione del bot (descrizione attività)
- `/setuserpic`: cambia l'immagine profilo del bot (logo attività)
- `/setjoingroups`: determina se un bot può essere aggiunto o meno ad un gruppo

4.2.2 Ricezione messaggi

Ogni richiesta alle bot API avviene via HTTPS e si presenta nella seguente forma: `https://api.telegram.org/bot<token>/METHOD_NAME`. Sono supportate le richieste sia GET che POST, oltre a quattro metodi per il passaggio dei parametri:

- `URL query string`
- `application/x-www-form-urlencoded`
- `application/json` (tranne per caricare i files)
- `multipart/form-data` (per caricare i files)

La risposta viene presentata sotto forma di JSON.

Si usa il metodo `setWebhook` per impostare un URL al quale ricevere gli aggiornamenti dal bot. Ad ogni messaggio ricevuto, viene inviata una HTTPS POST request all'URL specificato, contenente un JSON-Serialized `Update`.

L'applicazione qui discussa deve gestire molti bot, non solo uno singolo. Quindi il server deve essere in grado di capire da quale bot arriva ogni update. Per fare ciò, nell'URL che viene impostato per ricevere gli aggiornamenti, viene aggiunta una stringa diversa per ogni bot. In questo modo, il server è in grado di differenziarli.

4.2.3 Type

Update

This object represents an incoming update.

Field	Type	Description
`update_id`	`Integer`	The update‘s unique identifier.
`message`	`Message`	Optional. New incoming message of any kind
`callback_query`	`CallbackQuery`	Optional. New incoming callback query

Tabella 4.1: Update object [1]

Message

This object represents a message.

Field	Type	Description
`message_id`	`Integer`	Unique message identifier
`from`	`User`	Optional. Sender, can be empty for messages sent to channels
`date`	`Integer`	Date the message was sent in Unix time
`chat`	`Chat`	Conversation the message belongs to
`text`	`String`	Optional. For text messages, the actual UTF-8 text of the message
`voice`	`Voice`	Optional. Message is a voice message, information about the file

Tabella 4.2: Message object [1]

[1] https://core.telegram.org/bots/api

User

This object represents a Telegram user or bot.

Field	Type	Description
`id`	`Integer`	Unique identifier for this user or bot
`first_name`	`String`	User‘s or bot’s first name
`last_name`	`String`	Optional. User‘s or bot’s last name
`username`	`String`	Optional. User‘s or bot’s username

Tabella 4.3: User object [1]

Chat

This object represents a chat.

Field	Type	Description
`id`	`Integer`	Unique identifier for this chat
`type`	`String`	Type of chat, can be either “private”, “group”, “supergroup” or “channel”
`username`	`String`	Optional. Username, for private chats, supergroups and channels if available
`first_name`	`String`	Optional. First name of the other party in a private chat
`last_name`	`String`	Optional. Last name of the other party in a private chat

Tabella 4.4: Chat object [1]

CallbackQuery

This object represents an incoming callback query from a callback button in an inline keyboard. If the button that originated the query was attached to a message sent by the bot, the field message will be presented.

[1] https://core.telegram.org/bots/api

Field	Type	Description
`id`	`String`	Unique identifier for this query
`message`	`Message`	Optional. Message with the callback button that originated the query
`data`	`String`	Data associated with the callback button. Be aware that a bad client can send arbitrary data in this field

Tabella 4.5: CallbackQuery object [1]

InlineKeyboardMarkup

This object represents an inline keyboard that appears right next to the message it belongs to.

Field	Type	Description
`inline_keyboard`	`[[InlineKeyboardButton]]`	Array of button rows, each represented by an Array of `InlineKeyboardButton`

Tabella 4.6: InlineKeyboardMarkup object [1]

InlineKeyboardButton

This object represents one button of an inline keyboard. You must use exactly one of the optional fields.

Field	Type	Description
`text`	`String`	Label text on the button
`callback_data`	`String`	Optional. Data to be sent in a callback query to the bot when button is pressed

Tabella 4.7: InlineKeyboardButton object [1]

[1] https://core.telegram.org/bots/api

4.2.4 Method

sendMessage

Use this method to send text messages. On success, the sent Message is returned.

Parameters	Type	Required	Description
`chat_id`	`Integer or String`	Yes	Unique identifier for the target chat
`text`	`String`	Yes	Text of the message to be sent
`reply_markup`	`InlineKeyboardMarkup`	Optional	A JSON object for an inline keyboard

Tabella 4.8: sendMessage method [1]

4.3 Capire il testo

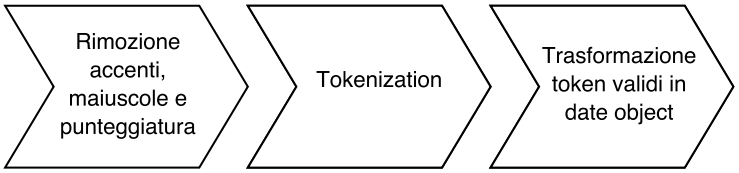

Figura 4.3: Processo di estrazione delle date dal testo

Dopo aver ricevuto l'`Update` da Telegram, viene estratto il testo del messaggio e, da questo, si cercano le date nelle quali il cliente richiede un appuntamento.

1. Il testo viene sottoposto ad un processo di standardizzazione. Vengono eliminati tutti gli accenti molto presenti nella lingua italiana: questo permette di non fare distinzione tra "lunedì" corretto e "lunedi" non corretto, errore molto frequente nei messaggi. Inoltre tutte le lettere maiuscole vengono trasformate in minuscole e, infine, viene eliminata la punteggiatura.

[1] https://core.telegram.org/bots/api

2. A questo punto, la stringa deve essere suddivisa in token, ovvero viene divisa ad ogni spazio in una sotto stringa e viene poi creata una lista con tutti i token ottenuti.

3. Tutti i token della lista vengono analizzati. Quelli che rappresentano dei numeri vengono trasformati in orari, mentre quelli rappresentanti delle keyword vengono trasformati nei giorni corrispondenti, a seconda della data di partenza. Le keyword sono token che rappresentano i giorni di una data, come per esempio oggi, domani o lunedì. Infine i giorni e gli orari vengono messi insieme per formare le date complete.

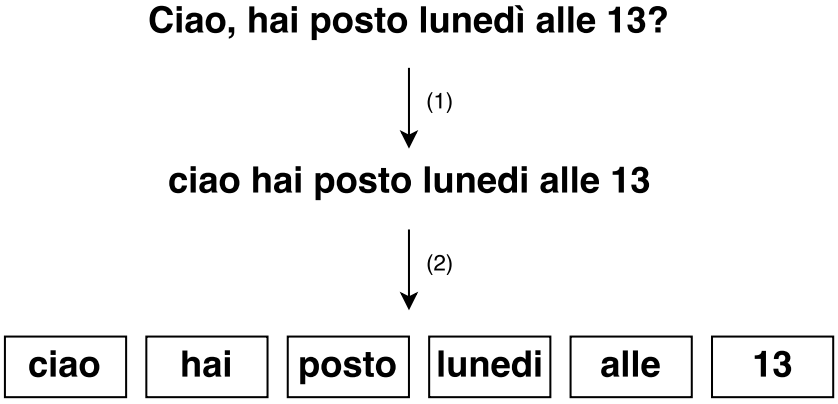

Figura 4.4: Tokenization

In questo modo, viene analizzata tutta la stringa. Il programma è così capace di trovare tutte le date presenti nelle stringhe (quindi anche più di una). Nel caso non sia specificato un giorno della settimana, viene preso quello della data stessa. Un esempio di stringa nella quale il giorno viene sottinteso è: "Ciao, hai posto alle 15?". Attualmente ci sono alcuni limiti, però, in quanto stringhe troppo articolate potrebbero non venire comprese correttamente.

4.4 Google Calendar

4.4.1 Ricerca appuntamenti

Si cercano adesso gli orari disponibili vicino alle date richieste dal cliente. Per fare ciò, come prima cosa si scaricano da Google Calendar tutti gli appuntamenti già presi nel giorno della data richiesta.

```
events = calendar_service.events().list(
    calendarId="primary", timeMin=orario_apertura,
    timeMax=orario_chiusura,
    singleEvents=True, orderBy="startTime").execute()
```

A questo punto, si cercano gli orari disponibili vicino a quello richiesto. La ricerca viene svolta in un preciso ordine: prima quelli possibili fino a due ore avanti, poi quelli possibili fino a trenta minuti indietro e, infine, quelli avanti fino alla chiusura dell'attività. Viene lasciato, tra un possibile orario e l'altro, un intervallo temporale minimo di trenta minuti e se ne cercano fino a quattro. Nel caso in cui la richiesta non specifichi nessun orario, ne vengono presi quattro distribuiti il più possibile equamente su tutto l'arco della giornata. Il tutto viene ripetuto per ogni data presente nel messaggio di richiesta. Durante la ricerca, si tiene conto degli orari dell'attività, con le eventuali pause e chiusure.

4.4.2 Annotazione appuntamenti

Prima di tutto, si controlla che l'appuntamento richiesto sia ancora realmente disponibile, poi si procede a segnarlo sul calendario:

```
event = {
    "summary": name, # Nome delle persona che richiede l'appuntamento
    "start": {
        "dateTime": inizio_appuntamento,
        "timeZone": zone, # Fuso orario della località dell'attività
    },
    "end": {
```

```
        "dateTime": fine_appuntamento,
        "timeZone": zone, # Fuso orario della località dell'attività
    }
}
calendar_service.events().insert(calendarId='primary', body=event).execute()
```

Il nome che viene segnato sul calendario è quello corrispondente all'account Telegram del cliente.

Capitolo 5

Esempio d'uso

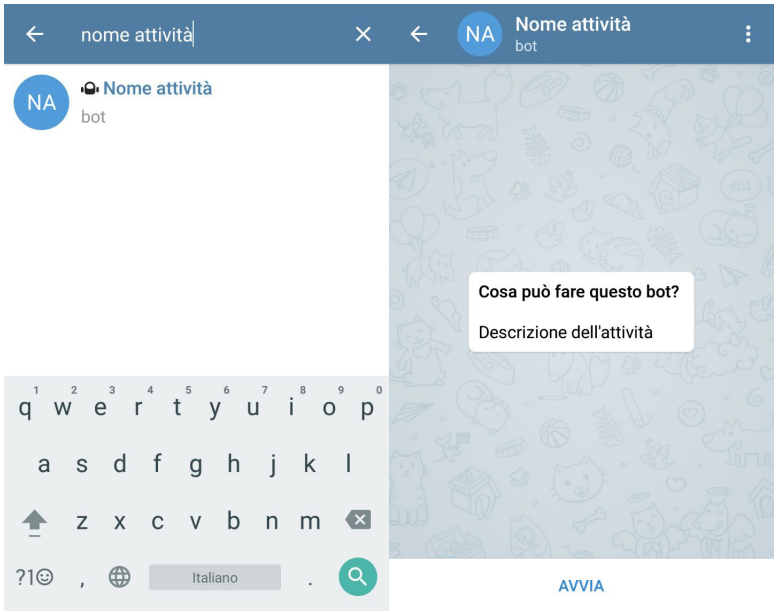

Figura 5.1: Ricerca dell'attività digitandone il nome e avvio della chat. Alla prima apertura della chat, compare la descrizione dell'attività in questione.

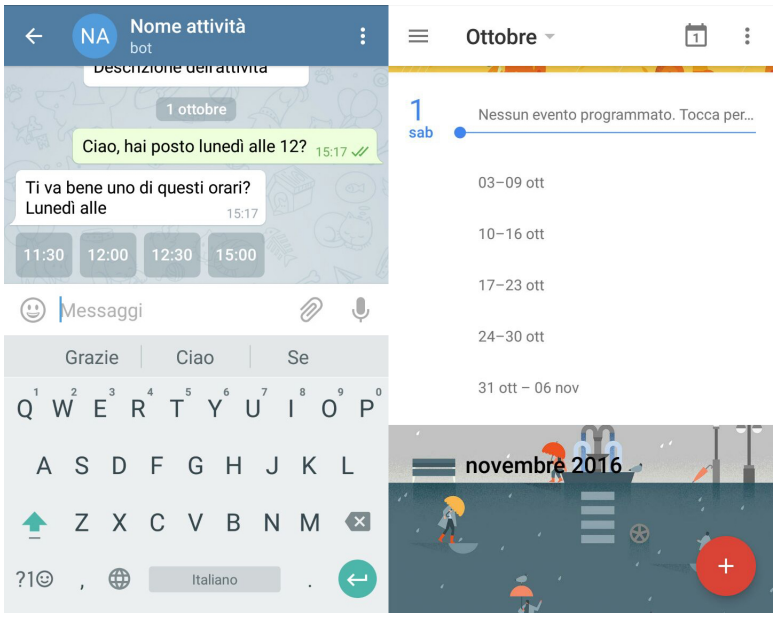

Figura 5.2: Richiesta di un appuntamento. Il bot offre varie opzioni al cliente, negli orari vicini a quello da lui indicato. In questo esempio, il salto dalle 12.30 alle 15.00 è dovuto alla pausa pranzo dell'attività.

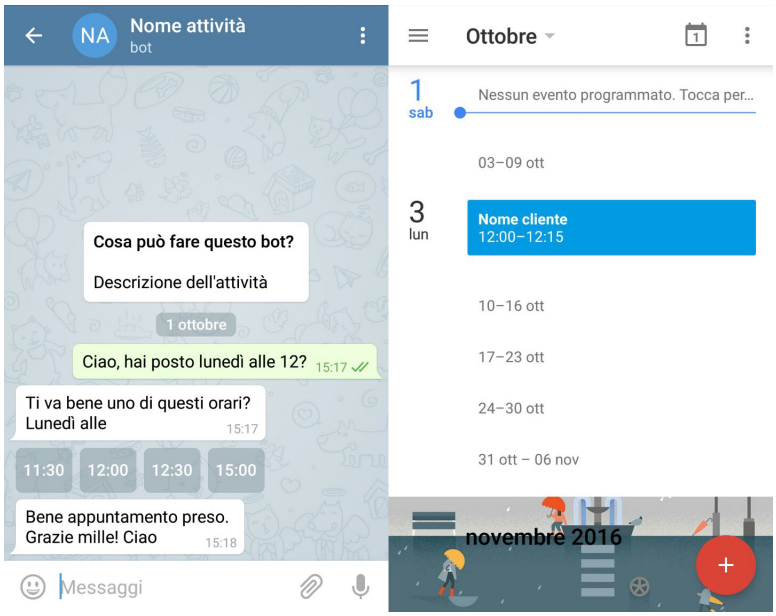

Figura 5.3: Conferma dell'appuntamento scelto dal cliente, con la conseguente annotazione sul calendario del professionista.

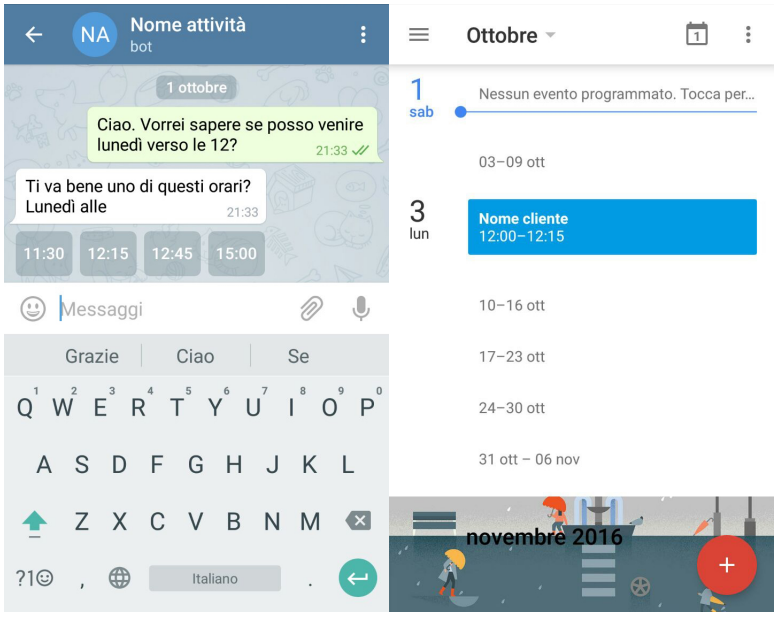

Figura 5.4: Richiesta di un appuntamento ad un orario già occupato. In questo caso, il bot offre diverse alternative in orari vicini a quello richiesto, ma tra questi non rientra l'orario esatto.

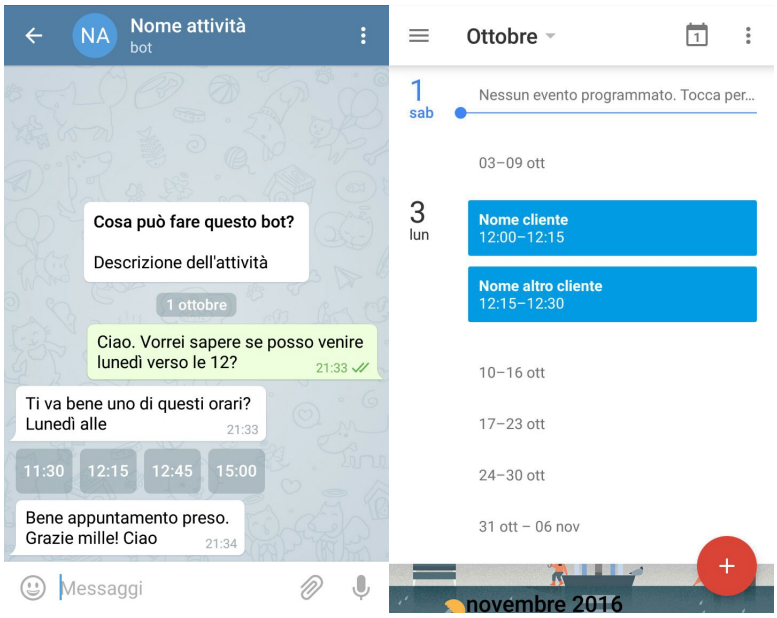

Figura 5.5: Conferma dell'appuntamento scelto da un secondo cliente, con la conseguente annotazione sul calendario del professionista.

Figura 5.6: Riconoscimento di keyword speciali come oggi, domani, dopodomani.

Figura 5.7: Riconoscimento di messaggi articolati: orari in varie forme (ad esempio 15,30 e 15.30), giorni sottintesi, date multiple, giorni senza accento (in questa immagine "lunedi", "martedi").

Figura 5.8: Errori nel riconoscimento di alcune date. Questo avviene quando il cliente sotto intende troppo nel messaggio o usa frasi troppo complesse. In entrambi i casi, per un giorno vengono restituiti gli orari di default (possibili orari durante tutto l'arco della giornata), come se non fosse stato specificato un orario nella richiesta.

Capitolo 6

Features e sviluppi futuri

Per rendere completamente funzionale e commercializzabile l'applicazione fin qui descritta, è necessario che in futuro vengano introdotte nuove caratteristiche e funzionalità.

Messaggi vocali

Oggi l'invio di messaggi vocali è molto in voga. Come la maggior parte delle applicazioni di messaggistica, anche Telegram supporta i messaggi vocali e, tramite le Telegram bot API, è possibile accedervi, come avviene per i normali messaggi di testo. Infatti ogni nuovo messaggio vocale è inoltrato al server esattamente come succede per quelli di testo, con l'unica eccezione che l'oggetto `Message`, invece di avere un campo `text`, ha un campo `voice`. A questo punto, l'unico passaggio necessario è la trasformazione dell'audio in testo, per poi avviarlo, così, agli stessi procedimenti dei messaggi normali. Per fare ciò, vengono utilizzate le Google Cloud Speech API, che, dato un audio contenente un messaggio vocale, lo convertono in testo. Queste ultime offrono una notevole affidabilità sulla conversione in testo, in quanto sono le stesse che Google utilizza per i suoi prodotti. Inoltre supportano il riconoscimento di più di ottanta lingue.

Traduzione in altre lingue

Per internazionalizzare i bot, sono due gli aspetti principali da dover prendere in considerazione. Il primo e anche più importante, consiste nella traduzione delle keyword. Queste devono comprendere i vari modi con cui si prende un appuntamento nelle diverse

lingue, quindi non solo le traduzioni alla lettera. Il secondo, invece, riguarda le date, che devono essere impostate con il giusto fuso orario e presentate con la corretta formattazione. Come ultima cosa, deve essere tradotto il front end usato per l'iscrizione, durante la quale il professionista deve impostare la propria nazionalità per avere la versione giusta del bot.

Miglioramento dell'intelligenza del bot

Uno degli aspetti più importanti consiste nell'intelligenza del bot, ovvero il modo in cui il programma è in grado di capire il testo inserito dal cliente, fornendogli così una risposta adeguata. Da questo punto di vista, c'è bisogno di miglioramenti che rendano il bot in grado di capire anche i messaggi molto complessi. Inoltre è necessario insegnargli a distinguere le varie situazioni: il cliente cerca informazioni ma non vuole prenotare, il cliente ha difficoltà durante la fase di prenotazione e cerca chiarimenti o, semplicemente, vuole venire a conoscenza dell'indirizzo dell'attività e degli orari di apertura.

Personalizzazione degli appuntamenti

In molti casi, l'attività offre ai clienti numerosi servizi che, molto probabilmente, hanno tempistiche diverse. Per gestire tutto questo, in fase di registrazione il professionista deve inserire i propri trattamenti, con le rispettive durate. In questo modo, durante la prenotazione il cliente può specificare anche il trattamento desiderato. A volte, inoltre, può essere presente più di un operatore. Prendendo l'esempio specifico di una parrucchiera, sono in genere presenti più persone a tagliare i capelli nel salone. Se il cliente ne predilige una in particolare e vuole specificarlo, durante la fase di prenotazione gli può essere fornita anche l'opzione di scelta.

Modifica o annullamento di un appuntamento

Bisogna integrare la possibilità per il cliente di disdire o spostare l'appuntamento preso direttamente dalla chat, senza dover così chiamare il professionista. Inoltre, se il cliente lo richiede, il giorno stesso dell'appuntamento l'applicazione può inviargli un messaggio per ricordarglielo.

Troll

Senza alcuna protezione, qualcuno potrebbe iniziare a prendere appuntamenti a caso, solo per divertimento. Per limitare comportamenti di questo genere, vengono introdotte due funzioni. La prima consiste nella richiesta del numero di telefono. Infatti, tramite le Telegram bot API, è possibile richiedere al cliente il suo numero, semplicemente facendogli cliccare un tasto per il consenso. Dopodiché il numero viene salvato: il professionista può così contattare il cliente anche telefonicamente in caso di emergenza. La seconda, invece, consiste nel limitare ad ogni cliente (account Telegram) il numero di prenotazioni simultanee a disposizione. Una volta raggiunta una certa soglia, non gli è possibile effettuarne altre.

Admin panel

Per il professionista, viene aggiunta una pagina di amministrazione dove gli è possibile vedere tutte le statistiche e gestire le varie impostazioni. Inoltre, invece di salvare gli appuntamenti sul calendario principale, se egli lo utilizza già per motivi diversi da quelli lavorativi, gli viene data la possibilità di salvarli su un calendario secondario creato appositamente.

Invio di messaggi informativi e promozionali

Il professionista può inviare, tramite l'admin panel, sia dei messaggi informativi, come cambi di orari dell'attività, sia promozionali ai clienti che lo hanno contattato in precedenza. È inoltre possibile impostare il bot in modo tale che, quando un cliente richiede un appuntamento, gli vengano proposti orari in cui l'agenda è più vuota, con un eventuale sconto nel caso della scelta di uno di quelli.

Altre app di messaggistica

Fino a questo punto, si è sempre parlato di Telegram, ma cosa ne è delle altre app di messaggistica? Con poco lavoro aggiuntivo sul programma che gestisce i bot, quelle aventi le API, possono essere supportate, in modo che il cliente utilizzi la sua preferita.

Ad oggi, Facebook Messenger è l'altra app diffusa in larga scala che li supporta. In un futuro è auspicabile che anche WhatsApp converga su questa strada.

Conclusioni

In questo lavoro, si è analizzato un possibile modo per migliorare la comunicazione tra professionisti e clienti, estrapolando dai metodi già esistenti gli aspetti migliori e cercando di eliminare gli svantaggi che questi comportano. Il caso più negativo, si verificherebbe nel momento in cui la maggior parte dei clienti non utilizzasse questa applicazione. Ciò comporterebbe sì un mancato miglioramento della fluidità lavorativa del professionista, ma, in fin dei conti, neanche un peggioramento. Egli è solo portato ad effettuare un cambio dell'agenda.

Si è visto anche come questa soluzione sia favorevole sia al cliente, in quanto non solo mantiene inalterati, ma cerca anche di migliorare i paradigmi di comunicazione a cui è abituato, sia al professionista. Quest'ultimo, in particolare, è colui che ottiene i maggiori vantaggi. Con questa applicazione, presupponendo che tale metodo di prenotazione venga adottato da una buona parte della sua clientela, egli può infatti constatare un notevole miglioramento nella qualità del suo lavoro, con una spesa sempre molto contenuta.

Bibliografia

[1] Bot, https://it.wikipedia.org/wiki/Bot, ultima consultazione: 30/09/2016

[2] Chat bot, https://it.wikipedia.org/wiki/Chat_bot, ultima consultazione: 30/09/2016

[3] Messenger Platform, https://developers.facebook.com/docs/messenger-platform, ultima consultazione: 30/09/2016

[4] Google App Engine, https://cloud.google.com/appengine, ultima consultazione: 30/09/2016

[5] Google Calendar API, https://developers.google.com/google-apps/calendar, ultima consultazione: 30/09/2016

[6] Google Cloud Datastore, https://cloud.google.com/datastore, ultima consultazione: 30/09/2016

[7] Google Cloud Speech API, https://cloud.google.com/speech, ultima consultazione: 30/09/2016

[8] Google Identity Platform, https://developers.google.com/identity, ultima consultazione: 30/09/2016

[9] Library httplib2, https://github.com/httplib2/httplib2, ultima consultazione: 30/09/2016

[10] OAuth 2.0, https://oauth.net/2, ultima consultazione: 30/09/2016

[11] Python `datetime`, `https://docs.python.org/2/library/datetime.html`, ultima consultazione: 30/09/2016

[12] Python `string`, `https://docs.python.org/2/library/string.html`, ultima consultazione: 30/09/2016

[13] Telegram bots, `https://core.telegram.org/bots`, ultima consultazione: 30/09/2016

Ringraziamenti

Desidero innanzitutto ringraziare il Professore Luciano Bononi per avermi dato l'opportunità di svolgere questo progetto.

Inoltre i miei più preziosi ringraziamenti vanno ai miei genitori, che mi sono sempre stati vicini, permettendomi pazientemente di raggiungere questo traguardo e a tutta la mia famiglia per il sostegno che, consapevolmente o no, mi ha dato. Spero di riuscire a ripagare tutti i sacrifici e il supporto che mi hanno sempre dimostrato.

Un ringraziamento speciale va anche alla mia ragazza Ilaria che mi ha supportato e sopportato in tutti i momenti durante quest'ultimo anno. Spero di riuscire a ricambiare almeno in parte tutto quello che mi ha dimostrato.

Infine vorrei ringraziare i miei amici di sempre che, in un modo o nell'altro, hanno condiviso con me tutti questi anni. Grazie per essere sempre i migliori.