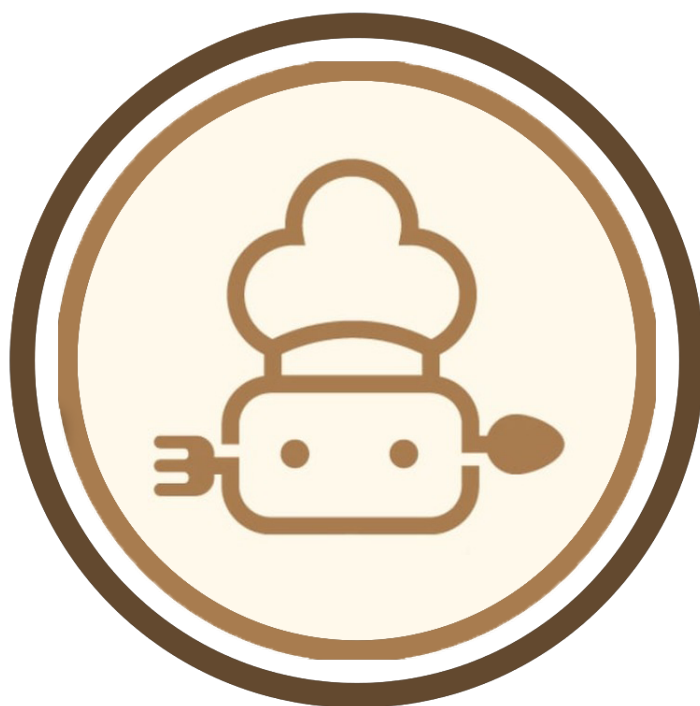


TasteIt - Progetto di Ingegneria Informatica

Paolo Pertino [10729600] paolo.pertino@mail.polimi.it

11 maggio 2022



Indice

1	Introduzione	2
1.1	Obiettivi	2
1.2	Requisiti	3
1.3	Installazione	4
1.3.1	Configurazione Telegram	4
1.3.2	Configurazione Google	4
1.3.3	Permessi per i gruppi	4
2	Funzionalità TasteIt	6
2.1	/lang - Modifica lingua	6
2.2	/settings - Modifica dei filtri sul raggio di ricerca	6
2.3	/cerca - Ricerca dei ristoranti	7
2.4	/preferiti - Lista dei preferiti	10
3	Architettura	11
3.1	Database	11
3.2	Class Diagram	12
3.3	Conversazioni - Macchine a stati	13
3.3.1	Modifica della lingua	13
3.3.2	Ricerca ristorante	14
3.3.3	Visualizza lista preferiti	15
3.3.4	Modifica impostazioni	15
4	Strumenti utilizzati	17

1 Introduzione

TasteIt è un bot Telegram che si propone di aiutare gli utenti a effettuare una ricerca del ristorante in cui consumare un pasto, in base alle loro esigenze in termini di ciò che vogliono consumare, di orario e di prezzo. La ricerca può aver inizio a partire dalla propria posizione, oppure dal nome di una località, piazza o via di interesse.

Infine, la semplice operazione di ricerca è stata corredata dalla possibilità di creare liste dei preferiti, interattività della conversazione nei gruppi con possibilità di creare sondaggi ed è inoltre fornito un supporto per diverse lingue (attualmente sono implementate solo Italiano e Inglese).

1.1 Obiettivi

Come anticipato nel paragrafo precedente, TasteIt si propone di essere una valida interfaccia per i servizi offerti da Google Maps in termini di ricerca di ristoranti. Per questo motivo, di seguito elenchiamo i principali obiettivi preposti:

- **Ricerca Personalizzata**

Offrire una ricerca personalizzata del ristorante in cui consumare il proprio pasto, in termini di cibo desiderato, orario di apertura, fascia di prezzo, posizione iniziale di ricerca e preferenze relative alla distanza massima che si vuole percorrere (2 filtri di default per indicare la volontà di sopraggiungere al locale a piedi o in macchina).

- **Informazioni del ristorante**

Fornire inizialmente delle informazioni riassuntive che permettano all'utente di scegliere un ristorante (che possiamo assumere essere valutazione complessiva e numero di recensioni) e successivamente, sotto richiesta dello stesso, mostrare informazioni più dettagliate, comprendenti anche indirizzo, numero di telefono, sito web (se presente) e recensioni degli utenti.

- **Gestione preferiti**

Offrire la possibilità ad utenti e gruppi di salvare i ristoranti di interesse in liste dei preferiti, consultabili in autonomia in qualsiasi momento.

- **Interattività nei gruppi**

Rendere interattivo ed utilizzabile il bot nei gruppi, in modo tale da essere uno strumento effettivamente utile per coordinare l'organizzazione di più persone. A tal proposito, solamente nei gruppi, a valle di una ricerca effettuata, permettere la decisione attraverso sondaggi.

- **Persistenza dei dati**

I dati principali dell'utente o di un gruppo, quali lingua ed impostazioni relative ai filtri di distanza massima, devono essere persistenti e non relegati alla singola conversazione.

- **Multilingua**

Permettere la personalizzazione della lingua *"parlata"* dal bot. Offrire di default Italiano e Inglese.

1.2 Requisiti

Per far fronte agli obiettivi dichiarati nella sezione precedente, TasteIt deve soddisfare i seguenti requisiti:

- **Parametrizzazione della ricerca**

Permettere all'utente di personalizzare la propria ricerca, fornendogli la possibilità di modificare tutti i parametri che caratterizzano la stessa (cibo desiderato, fascia di prezzo, orario, metodo di raggiungimento) nella fase che la precede.

- **Richiesta e gestione delle informazioni**

Per limitare i costi delle Places API è necessario fare richiesta, **nel momento opportuno**, delle sole informazioni di interesse. É pertanto necessario prevedere fasi distinte tra visualizzazione della lista dei ristoranti della zona di ricerca ed informazioni dettagliate di un singolo locale.

- **Persistenza e gestione dei dati**

Costruire un'apposita base di dati che permetta di salvare le informazioni relative agli utenti e ai gruppi, come lingua desiderata e liste dei preferiti, in modo tale che esse siano consultabili e modificabili in qualsiasi momento.

- **Compatibilità con i gruppi**

Fornire la compatibilità delle conversazioni, di default disponibili in chat privata, anche nei gruppi, prevedendo l'interazione del bot con molteplici agenti in una stessa conversazione. Per agevolare l'operazione di scelta di un ristorante, permettere la creazione di sondaggi tra i ristoranti appartenenti alla lista che si sta visualizzando.

1.3 Installazione

Per poter avviare il bot è necessario avere Python 3.x [1] installato sul proprio dispositivo. Attraverso il packet manager *pip* installare i requirements elencati nel file *requirements.txt*, recandosi nella directory principale del progetto (*./TasteIt*) e digitando il comando:

```
$ pip install -r requirements.txt
```

1.3.1 Configurazione Telegram

Successivamente è necessario creare un bot attraverso *@BotFather* ed ottenere la API key associata [2].

Creare dunque un file *.env* all'interno della cartella */src* in cui inserire la chiave appena generata con il seguente formato:

```
TELEGRAM_KEY=la_tua_chiave
```

Inoltre, collegarsi nella chat di *@chatIDrobot* e premere *Avvia* per ottenere informazioni riguardanti la propria chat. Copiare il numero che segue la dicitura *chat_id* ed inserirlo nel file *.env* con il formato seguente:

```
TELEGRAM_DEVELOPER_CHAT_ID_KEY=chat_id_copiato
```

In questo modo quando un errore inaspettato verrà riscontrato da un utente, il bot provvederà in automatico a segnalare lo sviluppatore.

La configurazione dei parametri inerenti Telegram è ora completata.

1.3.2 Configurazione Google

Infine, siccome il funzionamento del bot è relegato all'utilizzo dei servizi offerti da Google, attraverso le *Places API*, è necessario registrare un account su *Google Cloud Services* e fare richiesta per l'abilitazione di una key per le suddette API [3]. Una volta ottenuta la chiave, inserirla all'interno del file *.env* con il formato che segue:

```
GOOGLE_PLACES_KEY=la_tua_google_api_key
```

Terminato questo passaggio, il bot è pronto per essere avviato. Digitare dunque il comando:


```
$ py ./main.py
```

per avviare l'applicativo.

1.3.3 Permessi per i gruppi

Se si vuole aggiungere TasteIt ad un gruppo, assicurarsi di fornirgli i seguenti permessi da amministratore:

Aggiungi amministratore



TasteIt
non ha accesso ai messaggi

Cosa può fare questo amministratore?

- ☒ Cambiare le info del gruppo
- ☒ Eliminare messaggi
- ☐ Rimuovere utenti
- ☐ Invitare utenti tramite link
- ☒ Fissare messaggi
- ☒ Gestire le chat video
- ☐ Rimanere anonimo
- ☐ Aggiungere amministratori

Questo amministratore non potrà aggiungere nuovi amministratori.

Titolo personalizzato

Helper

Sarà mostrato un titolo personalizzato a tutti i membri invece di 'amministratore'.

Annulla Salva

Figura 1: Permessi da assegnare a TasteIt nei gruppi

2 Funzionalità TasteIt

Di seguito elenchiamo le funzionalità offerte dal bot ed i comandi necessari per l'interazione.

- */start* - Avvia il bot mostrando un messaggio di benvenuto all'utente.
- */help* - Permette la visualizzazione di tutti i comandi disponibili.
- */settings* - Permette la modifica di alcuni parametri di ricerca.
- */lang* - Permette il cambio di lingua. Esso impatterà sia sui messaggi di servizio sia sugli effettivi risultati di ricerca.
- */cerca* - Introduce la conversazione per iniziare la ricerca di un ristorante.
- */preferiti* - Mostra all'utente (o al gruppo) le sue liste preferiti.
- */annulla* - Annulla l'operazione corrente.

2.1 */lang* - Modifica lingua

Digitando il comando */lang* è possibile modificare la lingua con cui interagire con il bot. Quest'ultimo invierà un messaggio con allegato una tastiera contenente le lingue disponibili (di default soltanto Italiano ed Inglese).



Figura 2: Risposta al comando */lang*

Selezionando una delle bandiere, la lingua muterà in quella selezionata ed i dati verranno anche aggiornati nel database.

2.2 */settings* - Modifica dei filtri sul raggio di ricerca

Attraverso il comando */settings* è possibile modificare i parametri di default relativi alla massima distanza percorribile sia a piedi sia in macchina (di default a piedi 1.5 km e in macchina 20 km).

Tali nuove informazioni saranno salvate nel database in modo tale da essere utilizzate come raggio quando si effettua un'operazione di ricerca di un ristorante.

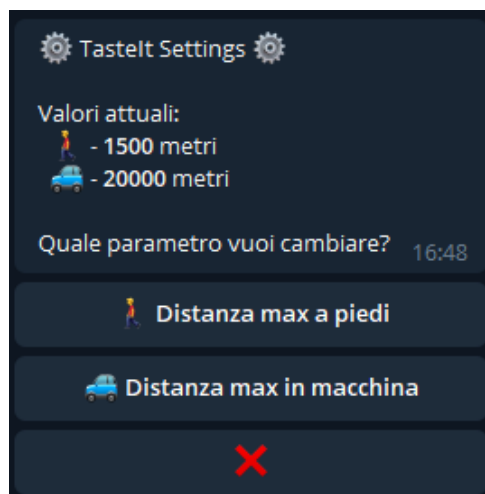


Figura 3: Risposta al comando /settings

Cliccando il bottone *Distanza max a piedi* o *Distanza max in macchina* sarà successivamente possibile scegliere un valore compreso tra 1 e 50'000 per indicare il nuovo raggio di ricerca da settare come valore di default (in metri) per gli spostamenti a piedi o in macchina.

2.3 /cerca - Ricerca dei ristoranti

A seguito del comando */cerca* inizierà la conversazione che gestisce l'intera fase di ricerca di un ristorante. La procedura è interamente guidata ed è articolata in diversi passi, di cui di seguito presentiamo una breve descrizione e delle immagini prese dalla conversazione stessa.

1. Richiesta della posizione

Il primo step richiede all'utente di inserire la posizione da cui far iniziare la ricerca. Essa può essere specificata manualmente fornendo il nome di una località, via, piazza, etc... oppure inviando la propria posizione attuale utilizzando la funzionalità nativa di telegram.

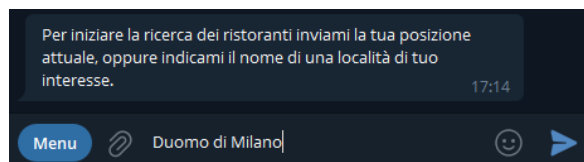


Figura 4: Posizione iniziale da nome



Figura 5: Posizione iniziale da posizione attuale

2. Richiesta del tipo di cibo

Il secondo step prevede l'inserimento da parte dell'utente di ciò che vuole mangiare.

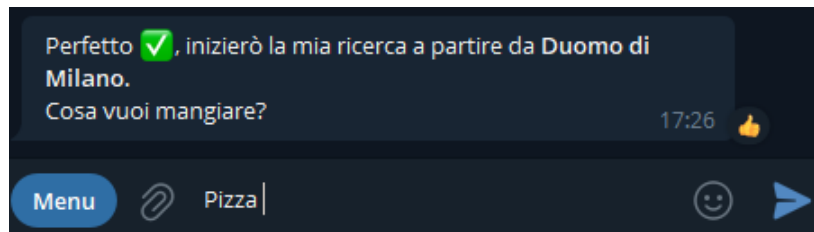


Figura 6: Scelta del cibo

3. Messaggio di recap

Il terzo step consiste nella revisione di tutti i parametri con cui verrà effettuata la ricerca. Al messaggio di recap vengono affissi dei bottoni che permettono di modificare tali parametri. Nello specifico è possibile:

- (a) modificare la preferenza riguardante il cibo;
- (b) includere nella ricerca i ristoranti che da orario sono al momento della stessa chiusi oppure escluderli e visualizzare soltanto quelli aperti;
- (c) modificare la fascia di prezzo;
- (d) scegliere se sopraggiungere al locale a piedi oppure in macchina;

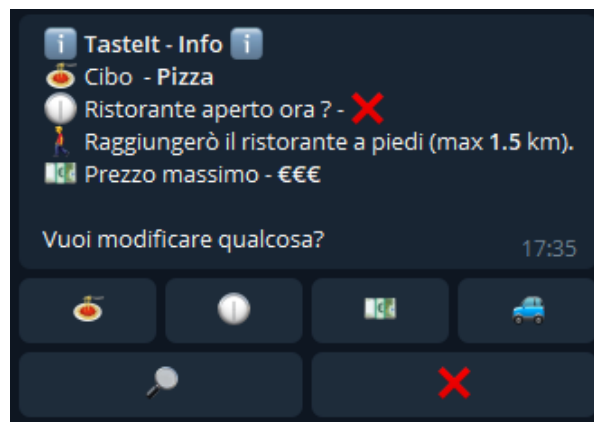


Figura 7: Messaggio di recap

Premendo la lente di ingrandimento avrà dunque inizio la ricerca con i parametri specificati.

4. Visualizzazione lista dei ristoranti trovati

Dopo una breve attesa in cui vengono interrogate le Places API, verrà inviato un messaggio a cui è affissa una tastiera per scorrere i ristoranti estratti che soddisfano le caratteristiche specificate negli stage precedenti. Per ogni locale sono mostrate poche informazioni, quali nome, valutazione media degli utenti e numero totale di recensioni.



Figura 8: Lista di ristoranti

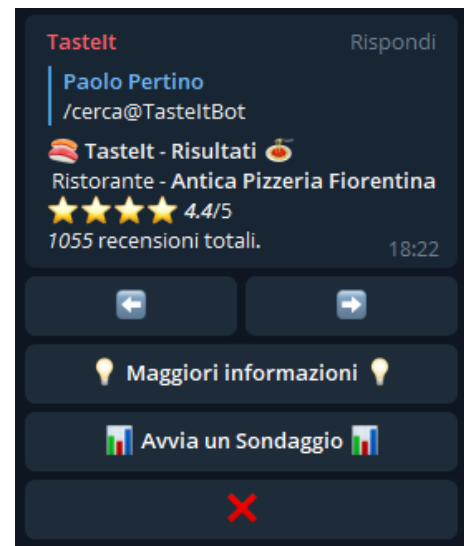


Figura 9: Lista dei ristoranti nei gruppi

5. Informazioni dettagliate

Cliccando il pulsante *"Maggiori informazioni"* verranno effettivamente mostrate info più dettagliate sul ristorante scelto.



Figura 10: Lista di ristoranti

I bottoni legati al messaggio permettono rispettivamente di visitare il sito web (se presente), aprire Google Maps alla posizione del ristorante per ottenere indicazioni, visualizzare le recensioni e aggiungere il locale ad una lista dei preferiti.

6. **Recensioni** TODO: visualizzare le recensioni

2.4 /preferiti - Lista dei preferiti

TODO: Inserire lista dei preferiti

3 Architettura

Nella sezione corrente viene resa esplicita l'architettura dell'applicativo. Nello specifico vengono presentate ed analizzate:

- Struttura del database.
- Oggetti custom creati per gestire le liste di ristoranti.
- Macchine a stati rappresentanti le conversazioni.

3.1 Database

Nella Figura 11 è stato riportato il diagramma ER rappresentante la semplice struttura del database.

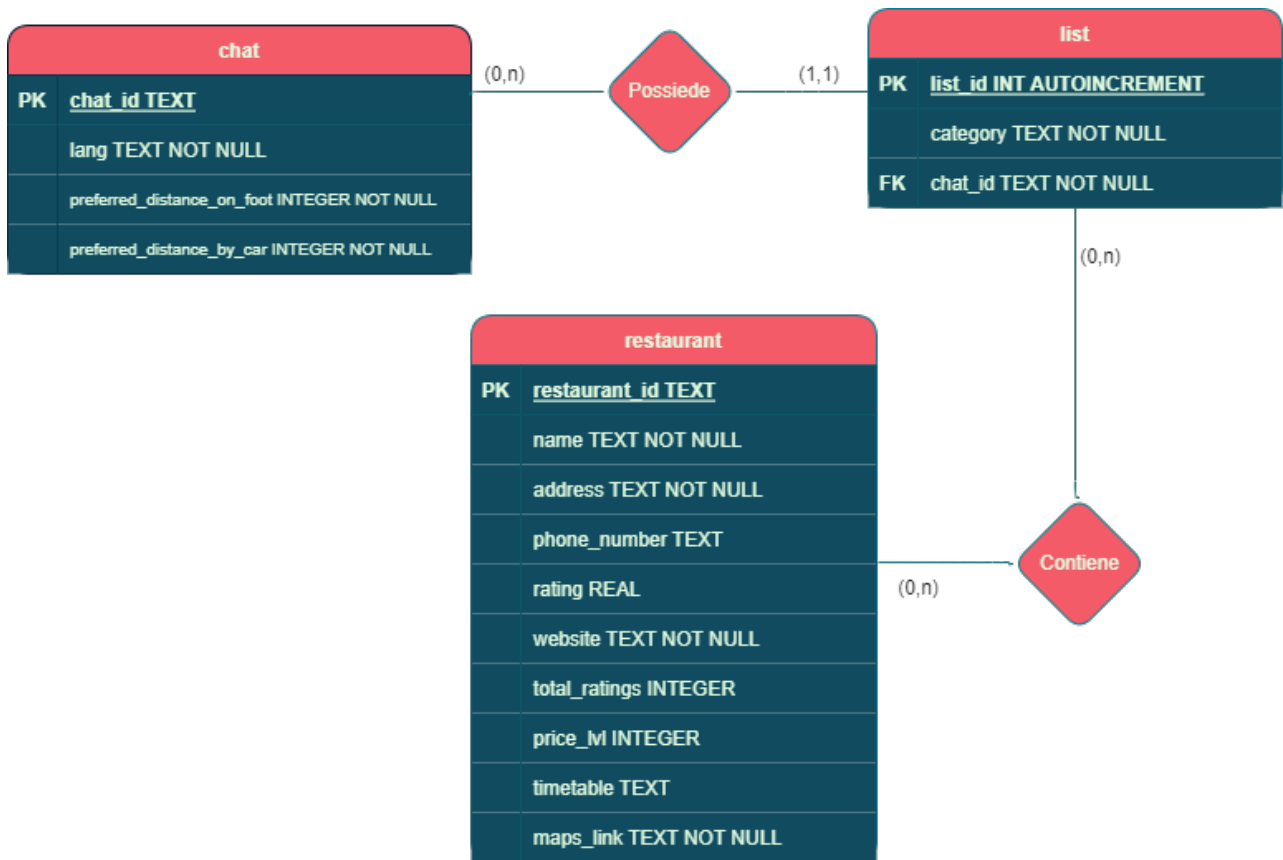


Figura 11: Schema ER Database TasteIt

La relazione 'Contiene' è stata successivamente trasformata in una tabella ponte chiamata *restaurant_for_list*. Pertanto, riportiamo di seguito lo schema logico dedotto dalla progettazione concettuale effettuata:

chat(**chat_id**, lang, preferred_distance_on_foot, preferred_distance_by_car)

restaurant(**restaurant_id**, name, address, phone_number, rating, website, total_ratings, price_lvl, timetable, maps_link)

list(**list_id**, category, chat_id)

restaurant_for_list(**list_id**, **chat_id**)

list.chat_id → chat.chat_id

restaurant_for_list.list_id → list.list_id

restaurant_for_list.chat_id → chat.chat_id

3.2 Class Diagram

Di seguito riportiamo inoltre un class diagram degli oggetti ausiliari creati ed utilizzati ai fini del progetto.

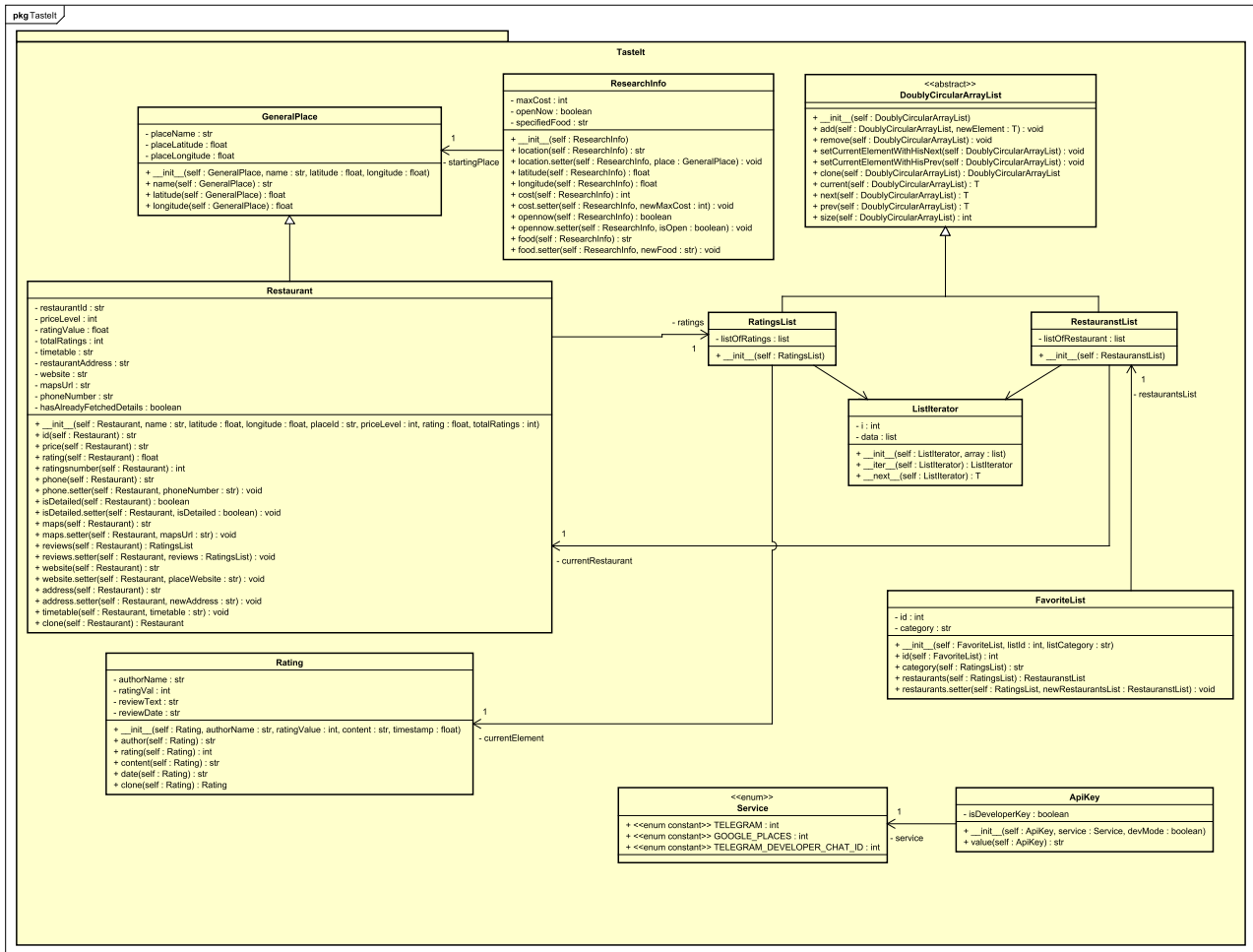


Figura 12: TasteIt - Class Diagram

Segue una breve descrizione del contenuto della Figura 12:

- *Service* enumerazione rappresentante i servizi per cui è disponibile una API key;
- *ApiKey* rappresenta la chiave stessa. Il metodo *value* cerca la key del servizio specificato all'interno del file *.env* o tra le variabili d'ambiente del sistema;
- *ResearchInfo* rappresenta le informazioni preliminari utili per effettuare una ricerca di un ristorante;
- *GeneralPlace* contiene le informazioni relative alla posizione di un determinato luogo;
- *DoublyCircularArrayList* classe astratta rappresentante una doppia lista circolare^[4] (differente dalla versione linked, in quanto i nodi non sono effettivamente linkati tra loro);
- *Restaurant* rappresenta un ristorante e tutte le sue informazioni (nome, latitudine, longitudine, 'costosità', valutazione media degli utenti, numero totale di recensioni, orario settimanale, indirizzo, sito web, link a google maps e numero di telefono);

- *Rating* rappresenta una recensione (autore, valutazione, contenuto e data);
- *RestaurantsList* implementazione di `DoublyCircularArrayList` rappresentante una lista di ristoranti;
- *RatingsList* implementazione di `DoublyCircularArrayList` rappresentante una lista di recensioni;
- *ListIterator* iteratore per le liste sopra descritte, il quale le rende iterabili da inizio a fine;
- *FavoriteList* rappresenta una lista preferiti (id, categoria e ristoranti in essa contenuti);

3.3 Conversazioni - Macchine a stati

In questa sezione verranno presentate le macchine a stati schematizzanti le conversazioni offerte dal bot e gli eventi che scatenano i cambiamenti di stato.

3.3.1 Modifica della lingua

La conversazione più semplice: in risposta al comando `/lang` viene invocata la funzione `setLanguage()` che presenta all'utente il messaggio con la tastiera per scegliere la lingua da utilizzare.

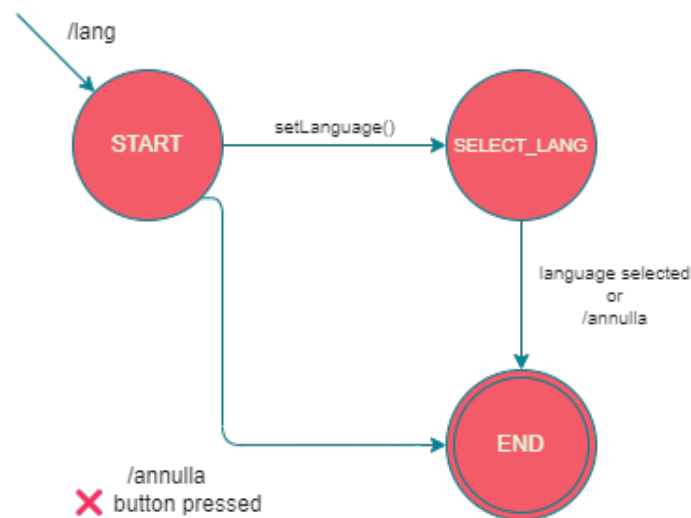


Figura 13: `/lang` - State Machine

3.3.2 Ricerca ristorante

Risponde alla richiesta effettuata con il comando `/cerca` e gestisce l'intera conversazione di ricerca dei ristoranti e visualizzazione dei risultati.

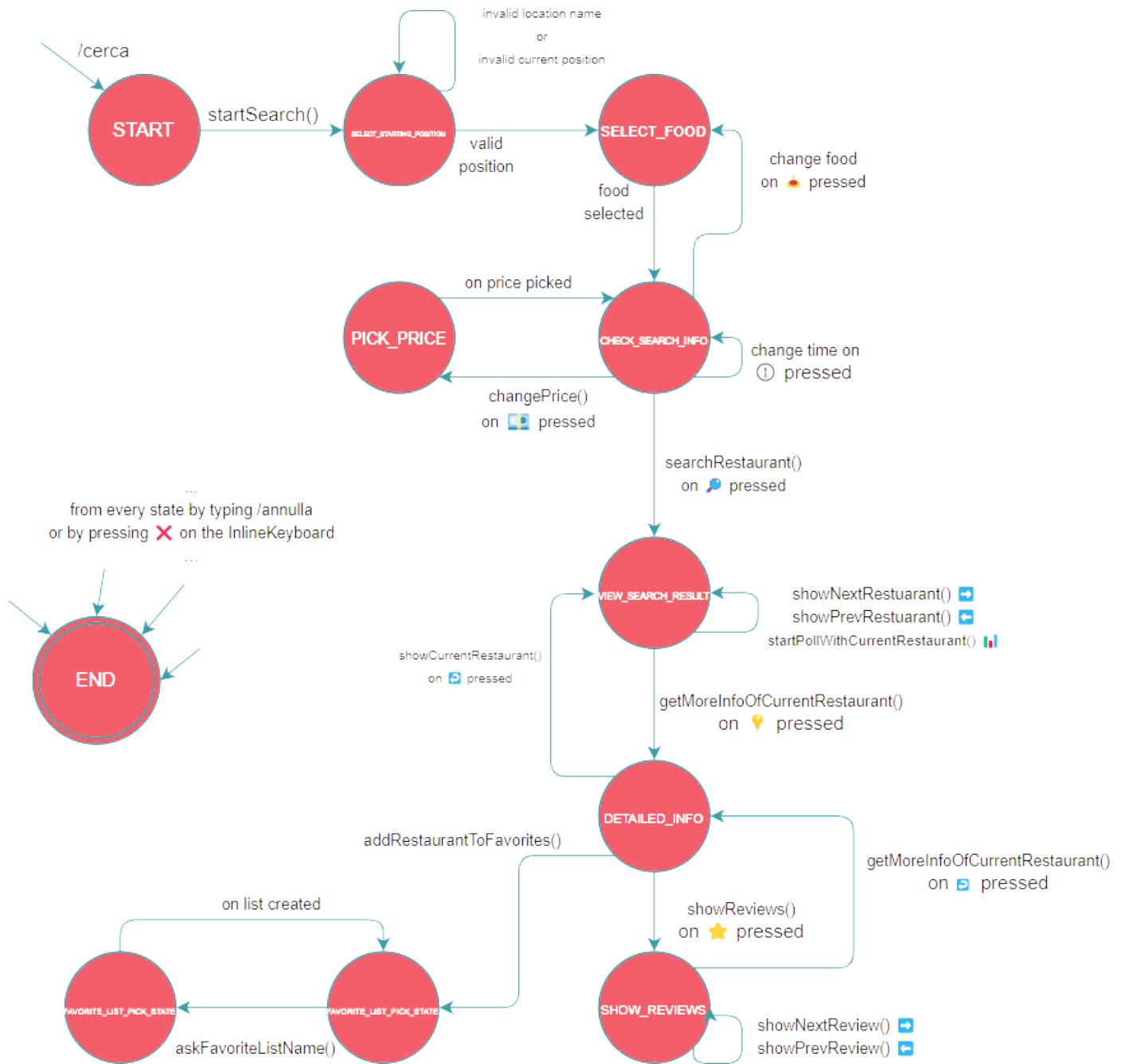


Figura 14: `/cerca` - State Machine

3.3.3 Visualizza lista preferiti

Risponde al comando */preferiti*. Gestisce l'intera conversazione che permette di visualizzare e modificare le proprie liste dei preferiti.

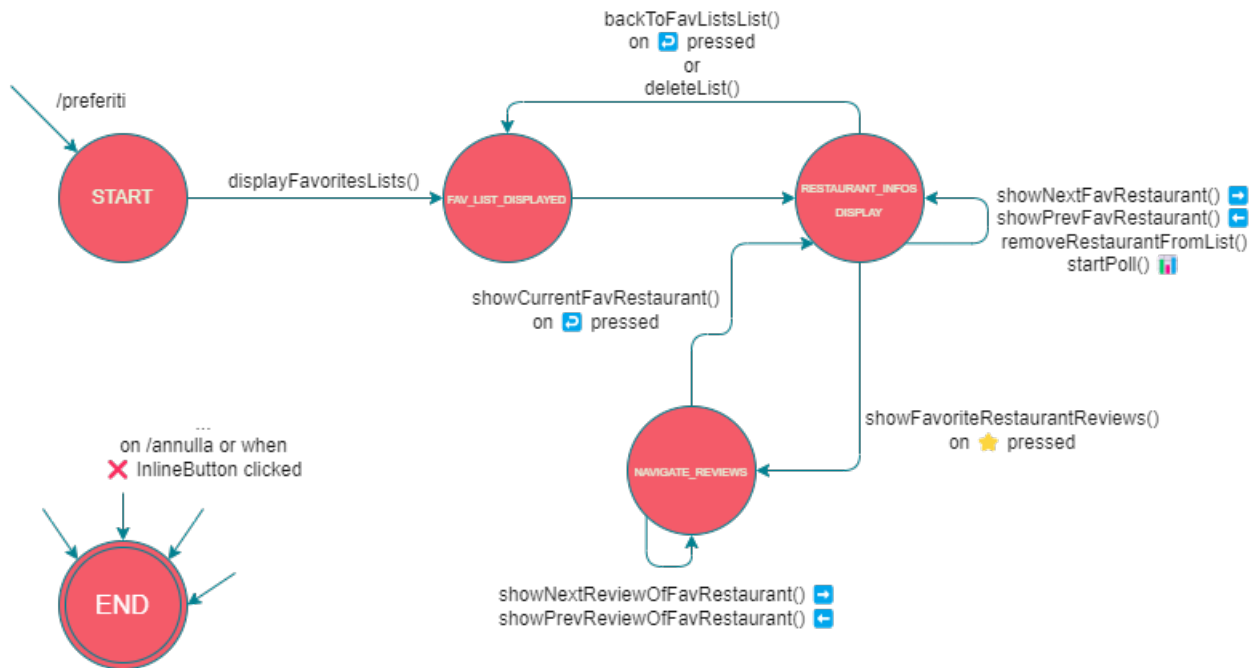
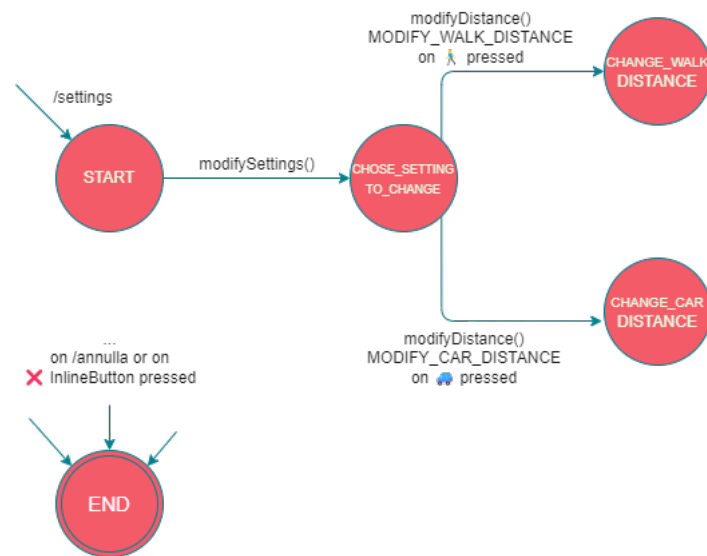


Figura 15: */preferiti* - State Machine

3.3.4 Modifica impostazioni

Risponde al comando */settings*. Gestisce la conversazione che permette all'utente di modificare i suoi parametri di default relativi alla distanza massima percorribile quando viene effettuata una ricerca.

Figura 16: `/settings` - State Machine

4 Strumenti utilizzati

Nella seguente sezione verranno indicati i principali strumenti di sviluppo utilizzati:

- *Visual Studio Code* - Principale IDE utilizzato.
- *python-telegram-bot* - Wrapper python usato per interfacciarsi con le API di Telegram.
- *Places API* - Google Maps API; gestiscono l'operazione di ricerca dei ristoranti.
- *SQLite3* - modulo per interagire con le API dell'omonima libreria SQLite, utilizzata per creare e gestire databases in locale.
- *Raspberry PI 4* - Hosting del bot e gestione del database.
- *AstahUML* - Creazione di diagrammi UML.
- *GitKraken* - Git GUI per visualizzare il workflow di sviluppo ed utilizzare efficientemente Git.
- *TEXStudio* - Gestione e aggiornamento del report.

Riferimenti bibliografici

- [1] [Python Download & Installation](#)
- [2] [Create a Telegram bot](#)
- [3] [Google Cloud Platform - Places API Key](#)
- [4] [Circular Doubly Linked List](#)