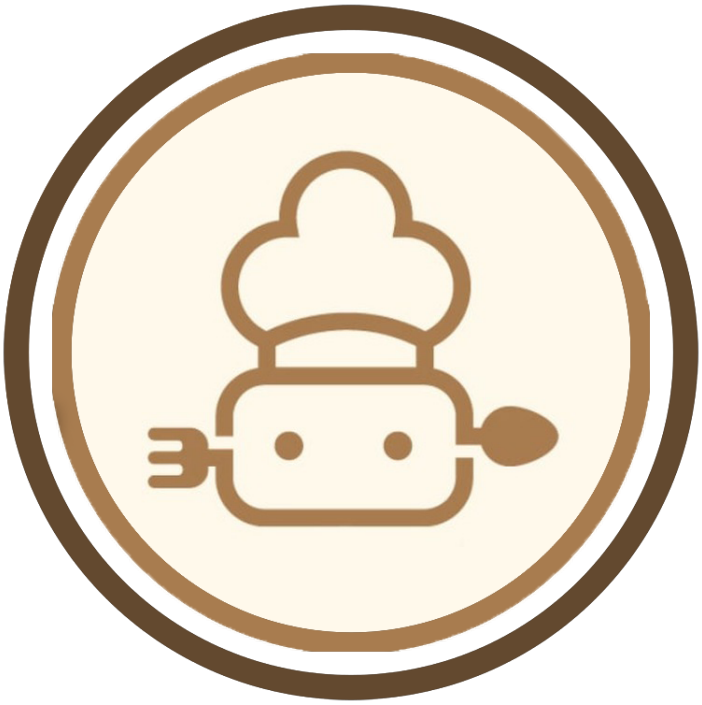
TasteIt - Progetto di Ingegneria Informatica

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

29 aprile 2022



Indice

1 Introduzione **2**
1.1 Installazione . . . 2
1.1.1 Configurazione Telegram . . . 2
1.1.2 Configurazione Google . . . 2

2 Funzionalità TasteIt **3**
2.1 /lang - Modifica lingua . . . 3
2.2 /cerca - Ricerca dei ristoranti . . . 3
2.3 /preferiti - Lista dei preferiti . . . 3

3 Architettura **4**
3.1 Database . . . 4
3.2 Class Diagram . . . 5
3.3 Conversazioni - Macchine a stati . . . 6
3.3.1 Modifica della lingua . . . 6
3.3.2 Ricerca ristorante . . . 6

4 Strumenti utilizzati **7**

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 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
```

1.1.1 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.

1.1.2 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 le 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.

2 Funzionalità TasteIt

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

- */start* - Avvia il bot mostrando un messaggio di benvenuto all'utente.
- */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 1: Risposta al comando /lang

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

2.2 /cerca - Ricerca dei ristoranti

TODO: Inserire ricerca ristoranti

2.3 /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 2 è stato riportato il diagramma ER rappresentante la semplice struttura del database.

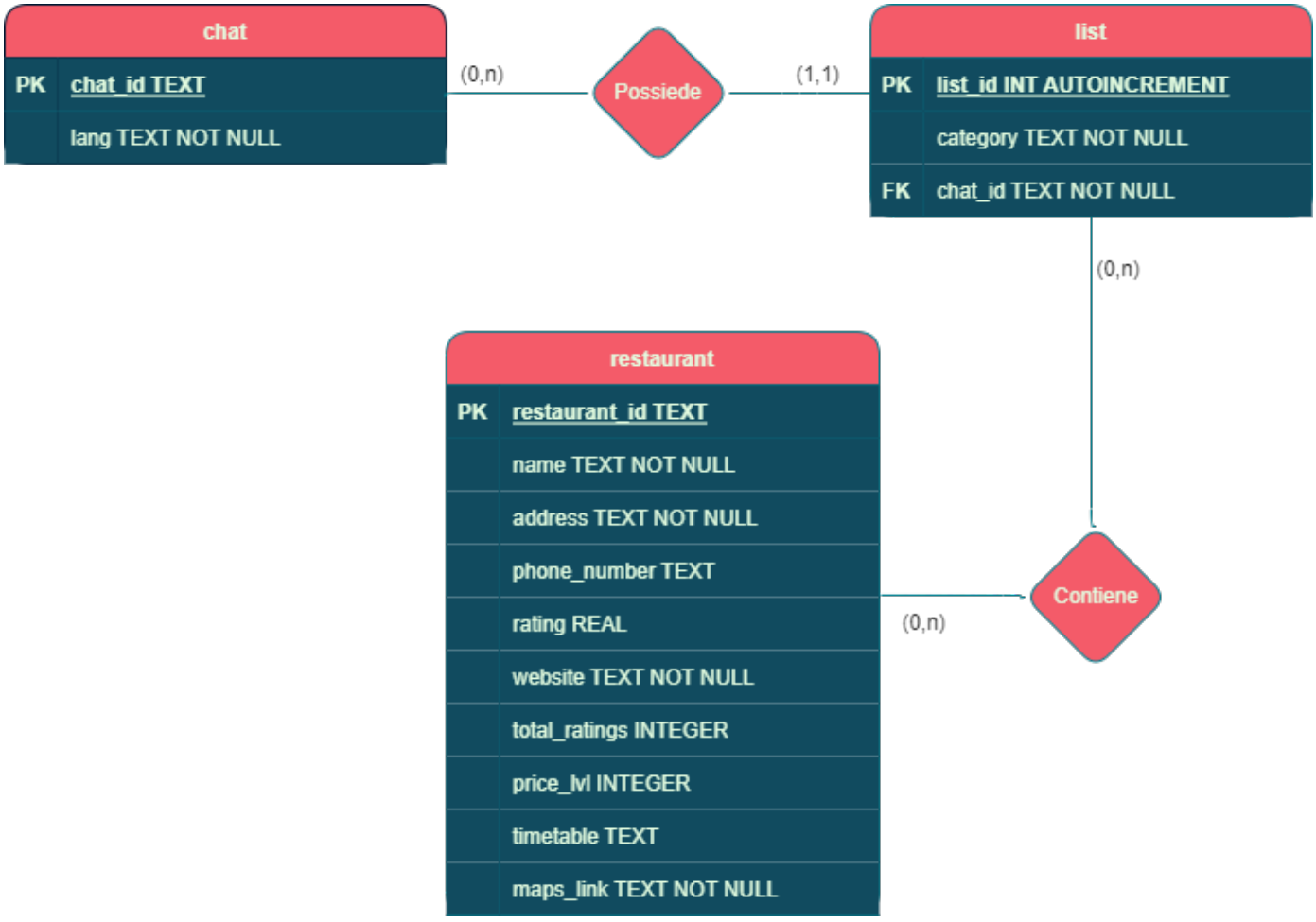


Figura 2: Schema ER Database TasteIt

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)
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.

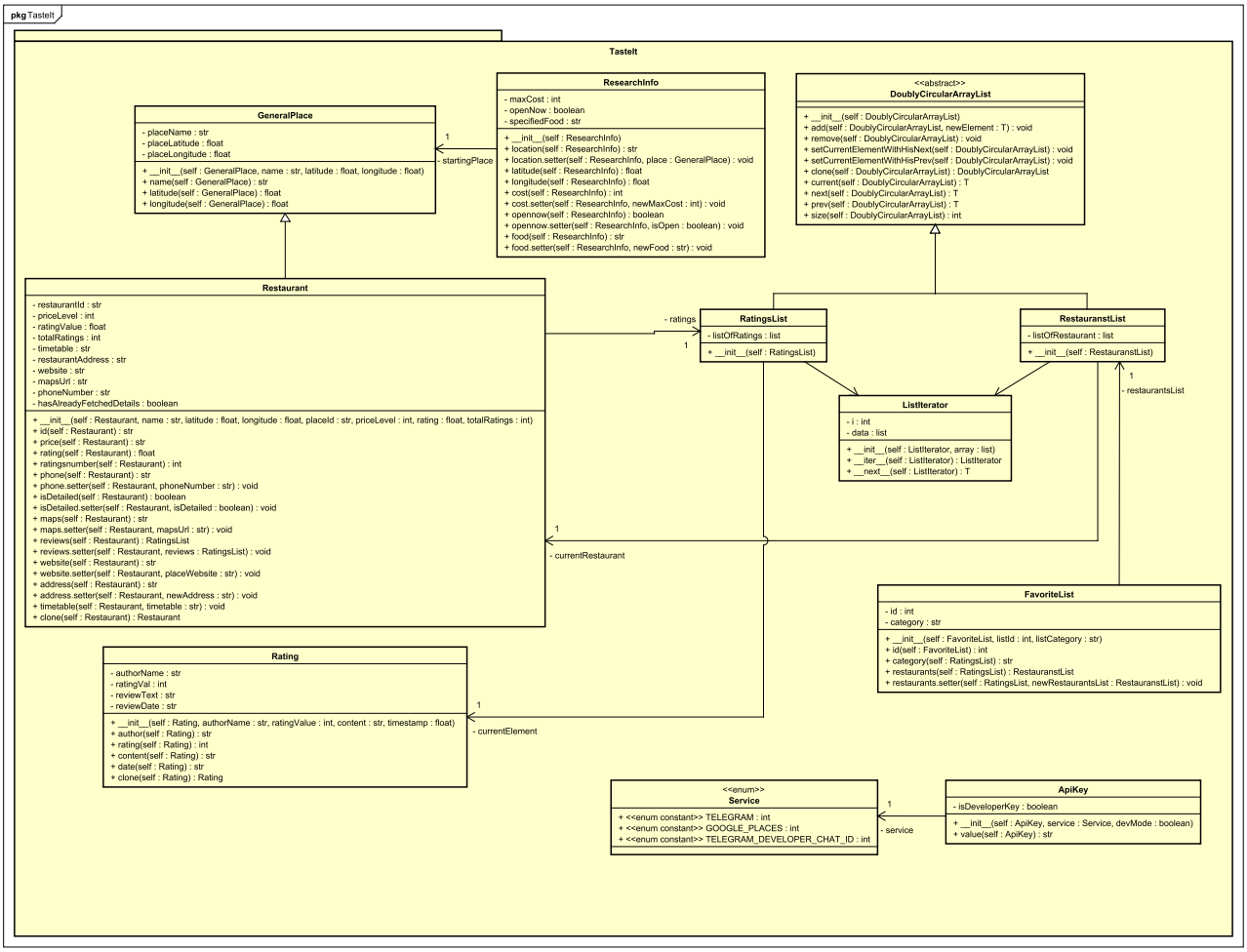


Figura 3: TasteIt - Class Diagram

Segue una breve descrizione del contenuto della Figura 3:

- *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.

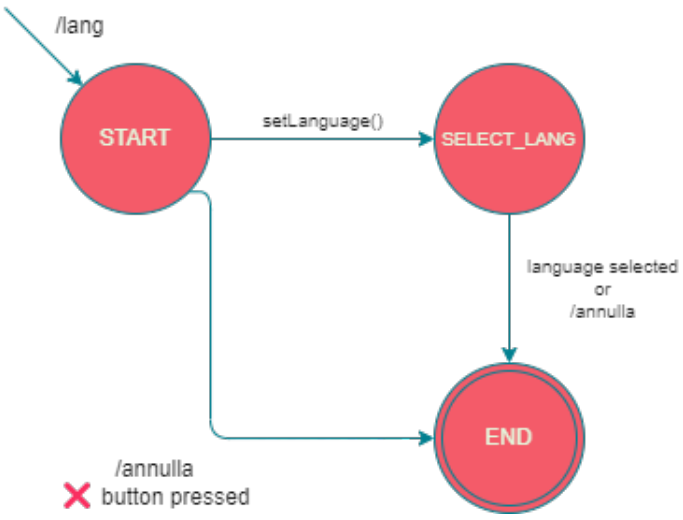


Figura 4: /lang - State Machine

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.

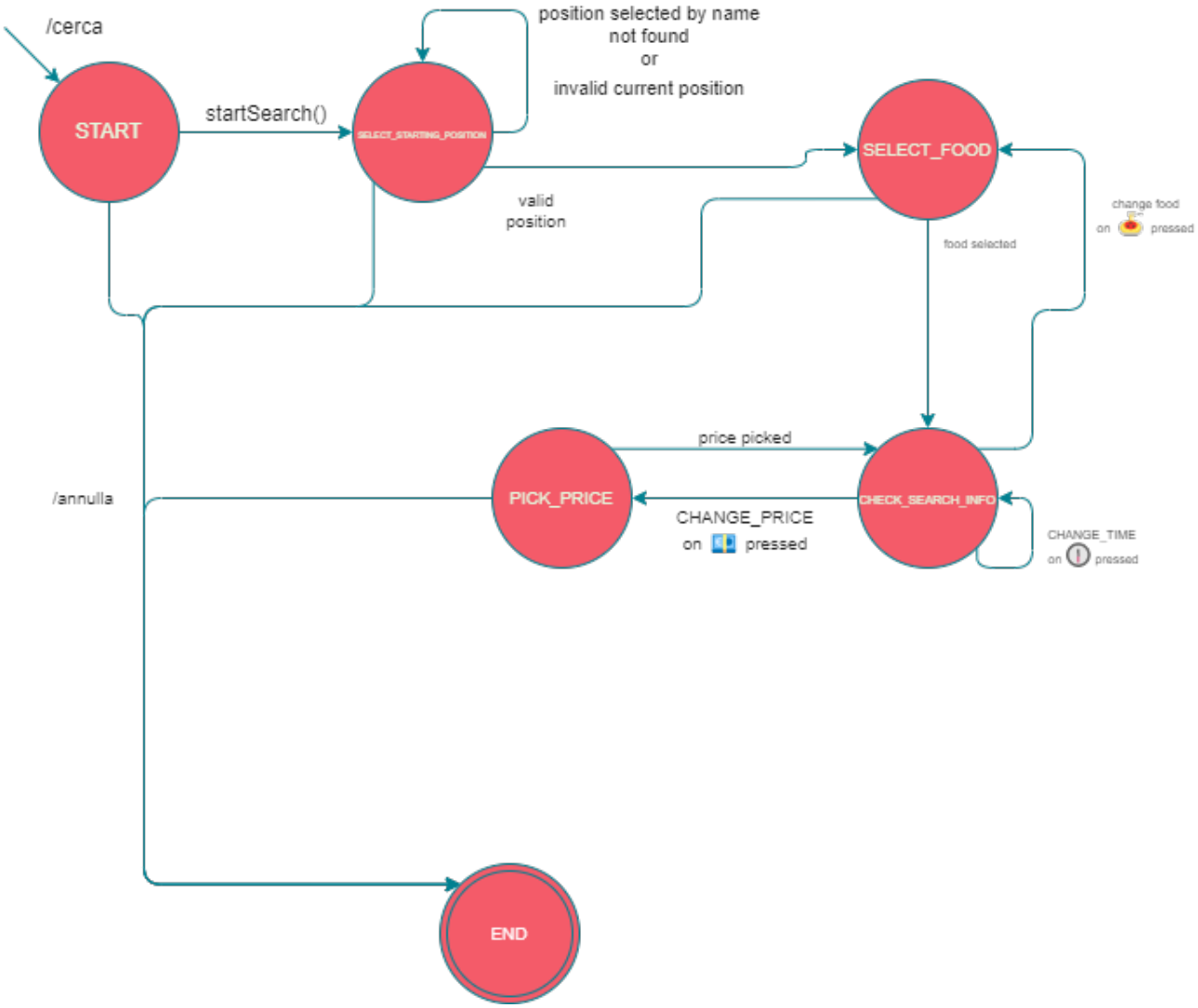


Figura 5: /cerca - State Machine

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