

TRIPLANNER: Documentazione

A.A 2024/2025

Progetto di Ingegneria della conoscenza

Progetto a cura di:

- Di Terlizzi Federica, 774492 : f.diterlizzi12@studenti.uniba.it

Professore del corso: Fanizzi Nicola

Github del progetto: <https://github.com/Federica-Di-Terlizzi/ICON-24-25>

Indice dei contenuti documentazione:

0. Decisioni di sviluppo;
1. Introduzione;
2. Struttura dell'interfaccia e caso d'uso;
3. Architettura;
4. Conclusioni;

Elenco argomenti di interesse:

- **Agenti e Sistemi intelligenti;**
- **Ontologie: Web semantico**
- **Rappresentazione e interrogazione su basi di conoscenza;**

0. Decisioni di sviluppo

L'applicazione in questione è stata sviluppata in python 3.12.10 e come IDE si è scelto di utilizzare "Intellij IDEA".

Di seguito si riportano le **librerie** principali presenti nel progetto:

- **tkinter**: usata per creare l'interfaccia grafica (GUI), dove `tk` è il modulo principale, mentre `ttk` (Themed Tkinter) è usato per un widget con uno stile semplice, `messagebox` è usato per mostrare finestre di avviso o errore.
- **SPARQLWrapper**: usata per effettuare interrogazioni su endpoint SPARQL al fine di richiedere al database remoto (Wikidata) informazioni sui monumenti o luoghi turistici delle città.
- **requests**: usata per effettuare richieste http al fine specifico di recuperare dati da API e scaricare risorse come le immagini.
- **rdflib**: serve per lavorare con i grafi RDF (Resource Description Framework), il modo in cui i dati sono rappresentati in Wikidata.
- **langdetect**: in questo progetto è utilizzato per riconoscere la lingua in cui è scritto il nome della città inserita nell'input, dato che le città potrebbero essere anche straniere.
- **Pillow**: usata per gestire e visualizzare le immagini dei monumenti o luoghi turistici nell'interfaccia utente.
- **Unidecode** e **unicodedata**: entrambe utili nel progetto per trattare accenti, simboli, maiuscole e minuscole nella compilazione dei campi di input.

Moduli presenti:

- **gui**: gestisce l'interfaccia utente dell'applicazione ed è il punto di avvio del programma.
- **itinerario**: è responsabile della generazione degli itinerari secondo un criterio di qualità
- **info_monumento**: comprende tutta la logica relativa all'acquisizione dei dati cioè delle immagini e delle descrizioni dei monumenti
- **qualita**: si occupa di calcolare i punteggi di qualità dei monumenti, assegnando il valore di 1 all'immagine e al documento (massimo 2 punti per monumento, se presenta sia immagine che descrizione)

Per poter eseguire correttamente il programma, si richiede di installare le librerie SPARQLWrapper, Pillow e requests nelle versioni indicate nel file "requirements. txt".

Ontologia utilizzata:

Il dataset principale del progetto è Wikidata, che segue il modello RDF (Resource Description Framework). Come risorse alternative, per compensare la possibile mancanza dei dati in Wikidata, vengono utilizzate Wikipedia e Wikimedia Commons.

1. INTRODUZIONE:

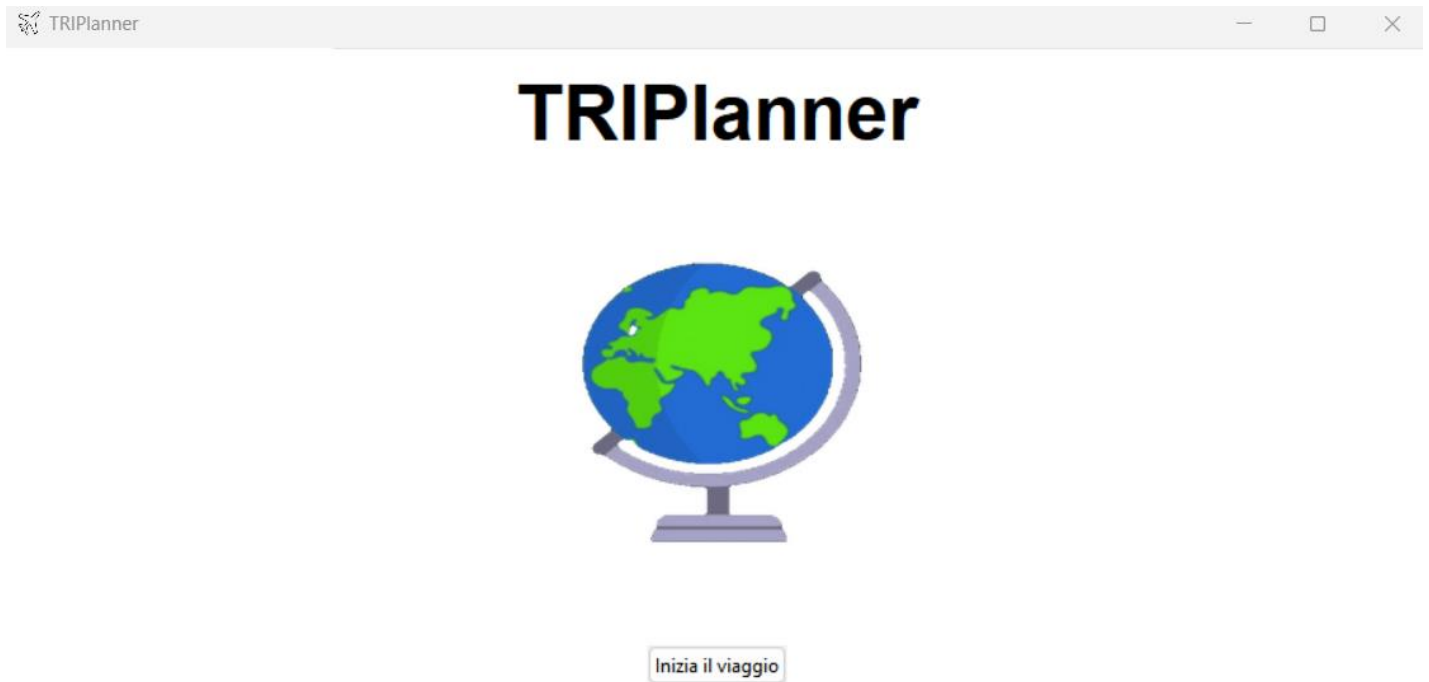
TRIPPlanner è un'applicazione progettata per semplificare l'organizzazione di un viaggio, offrendo agli utenti suggerimenti su cosa visitare in base alla città selezionata. Grazie a un'interfaccia semplice e intuitiva, l'applicazione permette di creare un itinerario giorno per giorno, con un massimo di 4 monumenti proposti per ciascun giorno. Tuttavia, non si limita a restituire per ogni città una lista di monumenti (tra i quali sono compresi teatri, chiese, musei, siti culturali ecc.), ma li distribuisce nei vari giorni secondo un criterio di qualità. Tale criterio stabilisce che i monumenti che presentano sia immagine e descrizione siano quelli distribuiti per prima: partendo dal primo giorno se ne inserisce uno per volta fino ad arrivare all'ultimo giorno per poi ricominciare il ciclo; successivamente, si distribuiscono i monumenti che hanno solo immagine o descrizione, a partire dal giorno successivo a quello in cui è stato inserito l'ultimo monumento completo; si completa poi l'itinerario con i monumenti che non hanno né immagine né descrizione, seguendo lo stesso ragionamento.

L'obiettivo principale di TRIPPlanner è rendere più veloce la ricerca e l'organizzazione dei luoghi da visitare in una città, eliminando la noia dell'attività di planning e migliorando l'esperienza complessiva di chi pianifica un viaggio.

2. Struttura dell'interfaccia e caso d'uso

L'interfaccia utente è semplice ed essenziale:

Tramite il pulsante «*Inizia il viaggio*», l'utente viene portato dalla pagina di intro alla pagina iniziale dove è richiesto l'inserimento degli input (nome della città e numero di giorni di soggiorno), all'interno degli appositi campi.



Attenzione! Nel caso in cui l'utente inserisca un valore numerico o un carattere, rispettivamente nel campo del nome della città e in quello del numero di giorni di soggiorno, appare un messaggio di errore in cui si specifica all'utente di inserire solo caratteri per la città e un numero di giorni maggiore o uguale a 1.


Una volta cliccato sul pulsante «*Crea itinerario*», appare la lista di città con nome uguale o simile a quella inserita. Selezionata la città desiderata e cliccato su «*Conferma*», si avvia la fase di caricamento. Al termine di questa, verrà mostrato all'utente la schermata con i monumenti o luoghi turistici divisi per giorno e classificati secondo un criterio di qualità.

The screenshot displays the TRIPlanner web application interface. The main heading is "Organizza il tuo viaggio". Below it, there are two input fields: "Nome città:" with the value "bari" and "Numero giorni di soggiorno:" with the value "3". A modal dialog box titled "Seleziona la città corretta" is open in the center. Inside the modal, it says "Seleziona la città desiderata:" followed by a dropdown menu showing "Nabari (Giappone)". Below the dropdown is a "Conferma" button. At the bottom of the main form, there is a "Crea itinerario" button.

N.B. Prima della conferma, l'utente ha la possibilità anche di chiudere il riquadro e cambiare gli input, per poi procedere come prima.


TRIPlanner

Giorno 1:




Villa Longo de Bellis

Villa Longo de Bellis è una dimora storica di Palese Macchie, sita sul lungomare Tenente Novello e con retrostante accesso in via Nazionale.



Teatro Margherita

Teatro Margherita is a former theatre in the city of Bari, Apulia on the east coast of Italy. Its predecessor, a wooden structure called Varietà Margherita opened on September 5, 1910. From 1912–1914, a new theatre was erected by architect Francesco De Giglio. It opened in 1914.



cattedrale di San Sabino

Il duomo di Bari, il cui nome ufficiale è Basilica cattedrale metropolitana di San Sabino, è la cattedrale di Bari, in Puglia, sede vescovile dell'arcidiocesi cattolica di Bari-Bitonto e monumento nazionale italiano.

Giorno 2:

Torna indietro

L'utente deve utilizzare lo scroll laterale per visualizzare l'intera pagina ma ha sempre a disposizione la barra con il bottone «*Torna indietro*» nel caso in cui voglia tornare indietro per generare un nuovo itinerario.

4. Architettura

TRIPanner integra i principi della rappresentazione della conoscenza e della ricerca, per generare itinerari turistici ottimali. È progettato per raccogliere, organizzare e suggerire luoghi di interesse da visitare, in base alla città selezionata. A partire da una base di conoscenza (Knowledge Base) che contiene dati e informazioni strutturate, il sistema esegue una serie di operazioni definite per generare un itinerario ottimale secondo un criterio di ordinamento prestabilito. I monumenti nel dataset sono descritti attraverso un'ontologia in formato RDF e pertanto rappresentati da un insieme di triplette RDF(individuo-proprietà-valore) che ne formalizzano le relazioni e le caratteristiche. Si utilizza così il linguaggio SPARQL per interrogare l'endpoint di riferimento. I monumenti vengono poi restituiti nel formato JSON, una struttura più semplice, facilmente integrabile e adatta alla comunicazione con l'utente.

Analizziamo nel dettaglio i moduli del progetto:

- nel file **gui.py** sono presenti 5 classi:

1. StartPage = racchiude solo l'animazione della pagina di intro
2. InputPage = in questa rientra la grafica della schermata con i campi di input e i controlli del corretto inserimento di questi. Inoltre, chiama la funzione `find_city_candidates` di `itinerario.py` per cercare la città ed eventualmente mostrarla all'interno di una lista con città di ugual nome (o contenenti la stringa di input) tra cui scegliere. Altrimenti, salva il qid e il label della città e lascia l'esecuzione alla pagina di caricamento.
3. LoadingPage = presenta la struttura della pagina di caricamento con l'avvio della progress bar.
4. ResultPage = mostra la pagina dei risultati con lo scroll verticale e la divisione per giorni. Per ogni monumento, tenta il caricamento immagine da URL attraverso la funzione globale `load_image_from_url` e se non la trova, la rimpiazza con un'immagine placeholder contenuta nella cartella `assets`.
5. TriPlannerApp = è la classe principale; si occupa della gestione delle relazioni tra le altre classi e lavora con le funzioni di `itinerario.py`. Dunque, è essenziale per attivare l'app.

- nel file **info_monumento.py** abbiamo:

- La variabile globale "headers" per definire lo user-agent usato nelle richieste http e rispettare così le API di Wikipedia/Wikimedia.
- La funzione `extract_description` che prende in input il risultato della query e controlla se c'è una descrizione, dando priorità a quella in italiano e, se manca, prendendo quella in inglese.
- La funzione `assign_source` assegna la fonte alle immagini o descrizioni recuperate.

- Le funzioni `_fetch_wikipedia_summary` e `_search_and_fetch_wikipedia` fungono da ricerca dei dati (immagine e descrizione) nelle fonti secondarie.
 - La funzione `_update_if_missing` serve a completare i dati di un monumento con informazioni trovate dalle fonti secondarie, senza sovrascrivere quello che già c'è.
 - La funzione `get_monument_data` è quella principale, il cui scopo è di arricchire i dati di un monumento combinando più fonti.
- Nel file **itinerario.py** è racchiusa la logica del progetto:
 - La funzione `unique_by_label` assicura di non mostrare duplicati di un monumento nella lista dei risultati.
 - La funzione `fetch_monuments` ricava, a partire dal nome della città, il qid corrispondente, attraverso la chiamata a `get_city_qid`; poi chiama `fetch_monuments_by_qid` che, preso in input il qid, cerca i monumenti e li restituisce.
 - La funzione `get_city_qid` non restituisce nulla nel caso in cui a un nome di città corrispondano più qid, lasciando che sia l'utente, attraverso l'interfaccia, a chiarire la sua scelta. Questa si basa sulla funzione `find_city_candidates` che cerca possibili città con il nome inserito dall'utente, recuperando qid, label, country e lang.
 - La funzione `plan_itinerary_by_popularity` si occupa infine di organizzare i monumenti nei diversi giorni a seconda del criterio di qualità stabilito e realizzato nella funzione del modulo `qualita.py`.
 - Nel file **qualita.py** è prevista una sola funzione: `quality_score`, che valuta la completezza delle informazioni di un monumento e assegna un punteggio da 0 a 2.

5. Conclusioni

In conclusione, il progetto è uno strumento utile per la ricerca e la distribuzione dei monumenti (inclusi chiese, teatri, musei, siti culturali ecc.) associati alle città. Per ottenere una distribuzione equa, si dividono i monumenti nei vari giorni di soggiorno, sulla base di un criterio di qualità che mostra in primis gli elementi che possiedono immagine e descrizione.

Fondata su un dataset online, l'applicazione non è limitata a un'area geografica specifica, permettendo di esplorare liberamente un ampio archivio di informazioni internazionali. Tuttavia, è importante notare che il programma non garantisce risultati per ogni città e potrebbe non riuscire a generare itinerari completi per viaggi molto lunghi.

Come idea di sviluppo futuro, si potrebbe arricchire l'applicazione con ulteriori opzioni di personalizzazione dell'itinerario, consentendo agli utenti di inserire più parametri di ricerca. Un ulteriore miglioramento futuro potrebbe consistere nell'integrazione di fonti dati più precise e complete, in modo da generare itinerari più dettagliati, anche per viaggi più lunghi.