UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA
Dipartimento di Scienze Fisiche, Informatiche e Matematiche
Progetto per l'esame di Algoritmi Distribuiti

TofuFilm: check when your favourites movies are in the theatre

Lorenzo Stigliano
Matricola: 1885534

Contents

1 Introduzione

Questo progetto prende spunto da un programma che avevo costruito anni fa la cui idea era quella di avere un bot Telegram in python per controllare che film fossero disponibili al cinema Victoria di Modena. Questo aveva funzionato egregiamente, ma avevo dismesso il progetto dopo qualche tempo perché impegnato con altre attività universitarie e non. Durante il corso di questo semestre, dovendo immaginare di creare una applicazione distribuita, ho pensato di rispolverare questa idea e di trasformarla in qualcosa di più artigianale e compliant con i requisiti del progetto. Ho eliminato completamente la parte del bot di telegram sostituendola con un server locale creato ad hoc.

Per portare a compimento il progetto ho fatto riferimento ad una libreria che permette di creare applicazioni distribuite o con modello client-server: Pyro4. Pyro è una libreria scritta interamente in Python che consente di costruire applicazioni in cui gli oggetti possono comunicare tra loro sulla rete ed è possibile farlo utilizzando i metodi python come siamo abituati a farlo, con quasi ogni tipo di parametro e valore di ritorno, mentre Pyro si occupa di individuare l'oggetto giusto sul computer corretto al fine di eseguire il metodo.

Durante lo sviluppo di questa applicazione ho deciso di sostituire l'interfaccia di Telegram con una libreria grafica python: tkinter. Del vecchio progetto sono stati mantenuti i metodi di scraping, ma sono stati aggiornati: prima questi salvavano tutto su file statici testuali sfruttando la libreria pickle, mentre ora utilizzo un database dedicato ai film per poter salvare tutte le informazioni raccolte durante lo scraping.

I requisiti del progetto sono i sequenti:

- Client
  - visualizzazione della programmazione settimanale dei film in sala o in specifici giorni scelti
  - consultazione listino prezzi ed informazioni rispetto ad eventuali scontistiche
  - visualizzazione sinossi e trailer, ove presenti, rispetto ai film in sala
  - possibilita' di scaricare un documento locale con all'interno tutta la programmazione settimanale delle proiezioni
  - interazione con il programma tramite GUI
  - richiesta di update delle informazioni contenute nel server: il client deve poter chiedere al server di effettuare un aggiornamento delle informazioni
  - registrazioni utenti al server
  - creazione di lista dei film preferiti (per utente)

- Server
  - fornire al client tutte le informazioni necessarie rispetto alla programmazione dei film e loro sinossi/trailer
  - operazioni di scraping al fine di ottenere tutte le informazioni utili al client
  - mantenimento di una copia locale delle informazioni che si sono ottenute più di recente. Questo perche' qualora il sito da cui fare scraping non fosse piu' disponibile ci sarebbe comunque una copia consultabile delle ultime informazioni raccolte.
  - storage degli utenti registrati/liste di film preferiti

Durante la fase finale di sviluppo è stata inoltre aggiunta una chat che permette agli utenti che hanno aperta una istanza del programma client, di poter comunicare con gli altri utenti che utilizzano l'applicazione. Il nome con cui ci si riferisce al programma è Tofu[1] o TofuFilm, con tutte le sue declinazioni: tofu_client, tofu_server, ...

[1] Questo in onore del defunto criceto dorato della mia ragazza, chiamato Tofu. Mi sembrava un modo carino per ricordarlo e rendere la scrittura del codice più dolce

Figure 1: Tofu: il criceto dorato

2 Avvio e Uso

Per poter far funzionare correttamente l'applicazione è necessario lanciare in sequenza i seguenti comandi:

```
python -m Pyro4.naming
python tofu_server.py
python tofu_client.py
```

Quasi sicuramente sarà necessario installare dei componenti aggiuntivi quali BeatifoulSoup, Pyro4, sqlite3, ... attraverso pip. Non si segnalano particolari incompatibilità tra le diverse versioni delle librerie che vi verranno suggerite di installare. Qualora si dovessero riscontrare problemi si consiglia di utilizzare conda come envirorment e i binari di python e pip forniti. Il primo comando ci permette di poter far comunicare tofu_client con tofu_server senza andare a copiare l'indirizzo URI che viene rilasciato come primo messaggio da `tofu_server.py` e dato in input a `tofu_client.py`. Tutto quanto il programma funziona da interfaccia grafica, ma sui terminali client e server vedrete su stdout alcune stampe che mi sono state utili per capire lo stato attuale del programma (in particolare del server).

3 Descrizione del Sistema

Il sistema basato su un paradigma client-server. Il client in questo caso ottiene dal server l'accesso a metodi che non possiede localmente e che gli permettono di poter ottenere le risorse necessarie al suo corretto funzionamento. IL server gestisce altri programmi in parte grazie threading di python. I programmi in questione sono relativi al funzionamento della chat presente all'interno del programma. Questa necessita di un server differente per poter funzionare, che viene pilotato dal server principale che effettua le operazioni di scraping sui film e che offre i servizi principali per il funzionamento dell'applicazione (login, registrazione, visualizzazione film presenti in sala, ...).

Per la memorizzazione dei contenuti relativi ai film e agli utenti che si sono registrati sul serve si fa affidamento a due databases distinti che vengono gestiti da un modulo preposto alle funzioni di inserimento e aggiornamento dati. Questo database viene interrogato per verificare che le credenziali immesse dall'utente all'atto del login siano corrette. Le password degli utenti **non vengono salvate in chiaro**, ma sono *hashate* tramite la funzione `sha256`. Il database degli utenti è composto da una singola tabella e le seguenti colonne: name (primary key), password, favourites list. La favourites list è la lista dei film preferiti da un determinato utente.

L'aggiornamento dei contenuti relativi ai film viene effettuato tramite una modulo che effettua diverse operazioni di scraping all'interno della pagina web del maggiore cinema di Modena. Tali operazioni si avvalgono di librerie quali: BeautifulSoup, requests, re (per effettuare regex e effettuare una milgiore pulizia del codice html da cui estrapolare informazioni utili). Una volta terminate tali operazioni viene aggiornato il database utile al mantenimento dei dati relativi ai film. Questo è composto da un unica table composta dalle seguente colonne: title, direction, genere, duration, cast, time_slots, reservation, trailer. Il campo time_slots si riferisce a giorni e orari in cui è possibile vedere un determinato film, mentre reservation è il link utile alla prenotazione del film.

Figure 2: Menu Principale dell'applicazione con utente loggato con nome *prova*

3.1 Architettura del Sistema

Lo schema che segue mostra i legami che intercorrono tra le varie parti del programma nella sua interezza. tofu_client funge principalmente da interfaccia grafica per tutti i metodi che sono forniti da tofu_server e gestisce il thread legato a chat_client (anch'esso interfaccia grafica basata per la chat basata su tkinter). In fase di sviluppo non è stato possibile integrare diversamente chat_client all'interno dell'applicativo poiché questo avrebbe portato a degli errori del tipo `main thread not in the main loop`. Essenzialmente chat_client ha un metodo per cui deve stare sempre in ascolto in attesa di messaggi provenienti dal server. Questo comportamento è bloccante rispetto al corretto funzionamento delle altre parti di tofu_client e per questo si è fatto uso di threads per poter invocare il client della chat senza compromettere il funzionamento dell'applicazione principale.

Lo stesso meccanismo avviene all'interno del modulo di tofu_server, in cui un thread si occupa della gestione di chat_server. Entrambi poi questi set vengono terminati una volta chiusa l'applicazione.

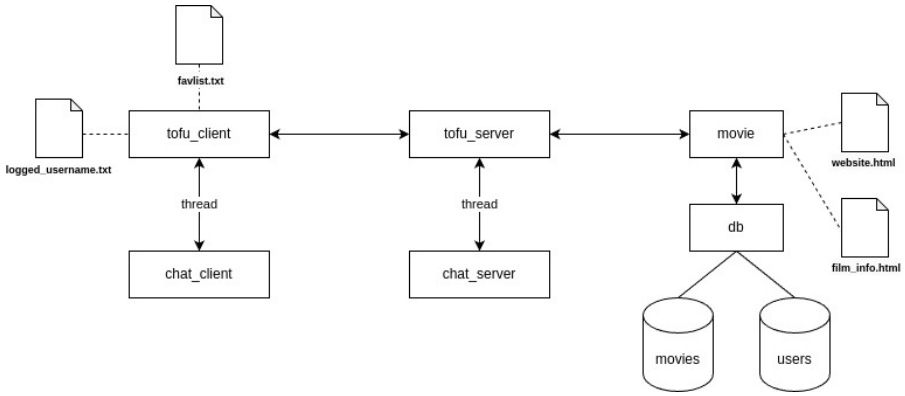

Figure 3: Schema architettura applicazione

4 Metodi e Interfacce

In questa sezione andremo a definire i principali metodi componenti Client e Server. Verranno esclusi metodi ridondanti o di testing. Si tratta di metodi usati per effettuare controllo di flusso all'interno dell'applicazione. Nel codice sorgente li potrete trovare commentati e con nomi molto precisi rispetto alla loro funzione.

4.1 Client

4.1.1 Tofu_Client

show_all Il metodo show_all permette di andare a visualizzare all'interno della finestra principale dell'applicazione, la lista dei film che sono proiettati attualmente all'interno delle sale del cinema. Per poter effettuare questa operazione invoca un metodo appartenente a tofu_client che gli ritorna il risultato della query sul database dei film che prende in considerazioni tutti quanti gli elementi presenti.

show_info All'interno della schermate principale non sono presenti tutti i campi (colonne) che ogni film possiede. Per poter visualizzare ulteriori informazioni risulta necessario premere il pulsante info che attiva il metodo show_info. Questo permette di andare a visualizzare informazioni aggiuntive rispetto al film che è stato selezionato all'interno della tabella dei risultati. Queste informazioni risultano utili al fine di consultare gli orari di proiezione, link di prenotazione, cast del film e trailer.

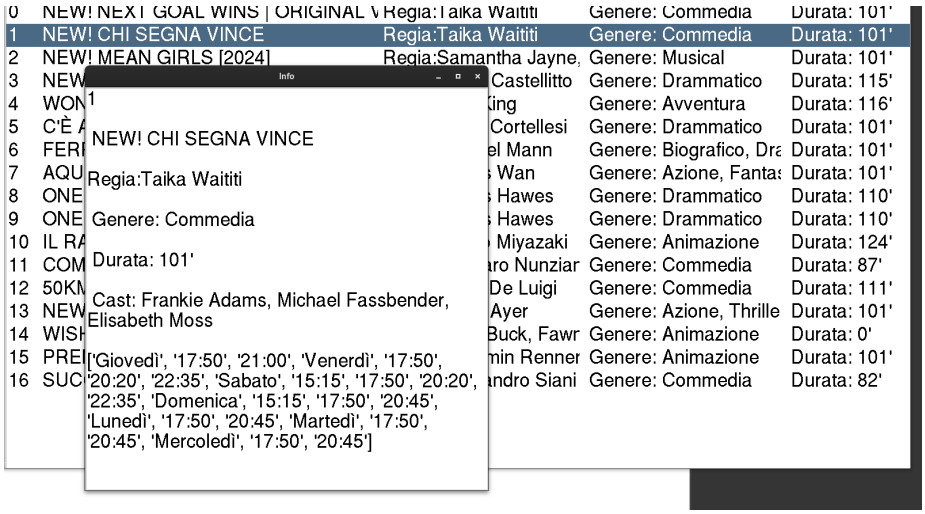

Figure 4: Film Info

kill_threads Tale metodo permette al client di poter terminare correttamente i threads relativi al funzionamento di chat_client e chat_server. Il primo è gestito come variabile della classe principale del programma tofu_client, mentre il secondo viene gestito da tofu_server.

db_update db_update si occupa di eseguire l'aggiornamento del database relativo ai film invocando il metodo preposto a tale operazione (esposto da tofu_server). Il metodo esposto dal server a sua volta invoca altri metodi appartenenti al file `movie.py` per effettuare le operazioni di scraping dei film.

dump_treeview Questo metodo si occupa del download di un file locale contenente la lista dei film che sono visualizzati all'interno della schermata principale. L'operazione viene effettuata utilizzando lo stesso metodo esposto dal server che si utilizza per visualizzare i film, solo che in questo caso il risultato della interrogazione sul database non viene visualizzata a schermo, ma salvata su un file.

register Il metodo register si occupa della registrazioni di un nuovo utente e funziona invocando il relativo metodo esposto dal server. Questo metodo del server interagisce con `db.py` per andare a inserire correttamente il nuovo utente. Non è possibile inserire nuovamente un utente che è già registrato, poiché prima dell'inserimento avviene un controllo sul campo username (che viene utilizzato come *Primary Key*).

login Il metodo login permette ad un utente che è già registrato di loggarsi correttamente. Questo funziona invocando un metodo che viene esposto dal server. Se la procedura avviene correttamente, all'interno del client sarà possibile visualizzare il nome dell'utente loggato.

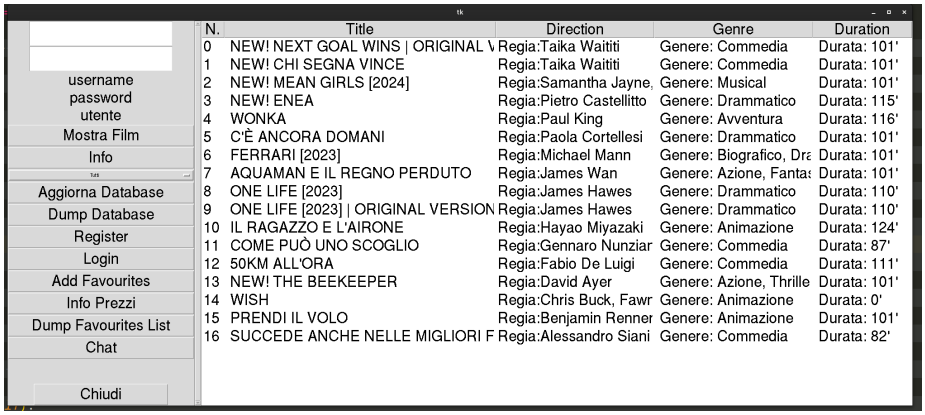

Figure 5: Enter Caption

add_favourites add_favourites consente di collezionare, all'interno di una variabile, la lista di film preferiti che poi verranno salvati all'interno del database all'atto della registrazione dell'utente. Cliccando su un film e poi premendo sull'omologo bottone è possibile aggiungerlo alla suddetta lista.

show_info_prices Questo metodo si occupa dell'ottenimento delle informazioni relative ai prezzi e convenzioni attive del cinema. Questo invoca un metodo del server che invoca il metodo preposto alle operazioni di scraping rispetto alle informazioni suddette. I dati raccolti vengono poi visualizzati dall'utente finale all'interno di una nuova finestra.

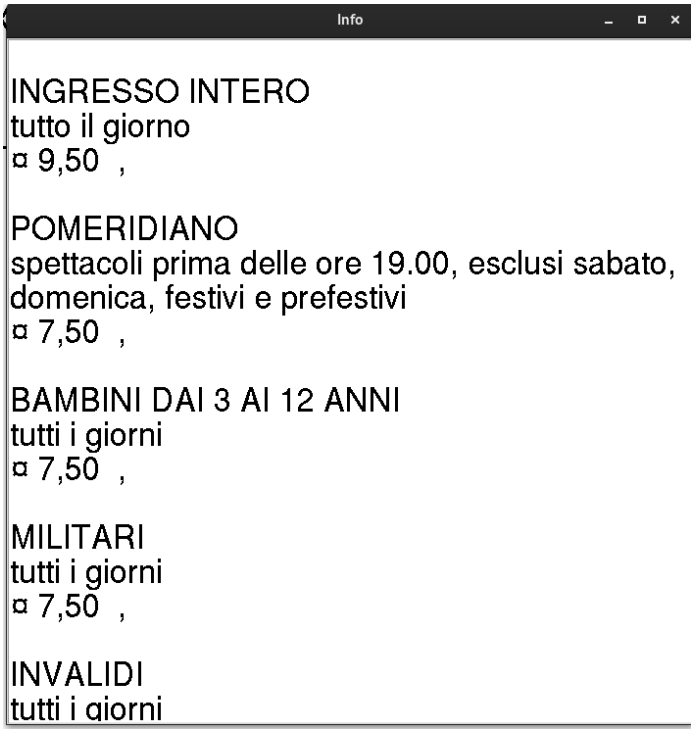

Figure 6: Info Prezzi e Convenzioni

dump_favourites_list Questo metodo permette di scaricare la lista dei film preferiti dell'utente che è attualmente loggato all'interno dell'applicazione. Questa operazione viene effettuata invocando un metodo esposto dal server che interroga il database e ritorna la lista dei film preferiti dell'utente su cui si è indagato. La lista dei film è salvata dal client all'interno di un file testuale.

film_chat film_chat invoca il metodo del server che gestisce il chat_server e successivamente viene esegue l'applicazione client relativa alla chat. La chat è

unica e onnicomprensiva, non vi sono chat specifiche per ogni film presente in sala. Ogni utente può accedervi e se non è loggato può comunque comunicare con gli altri utenti in maniera anonima.

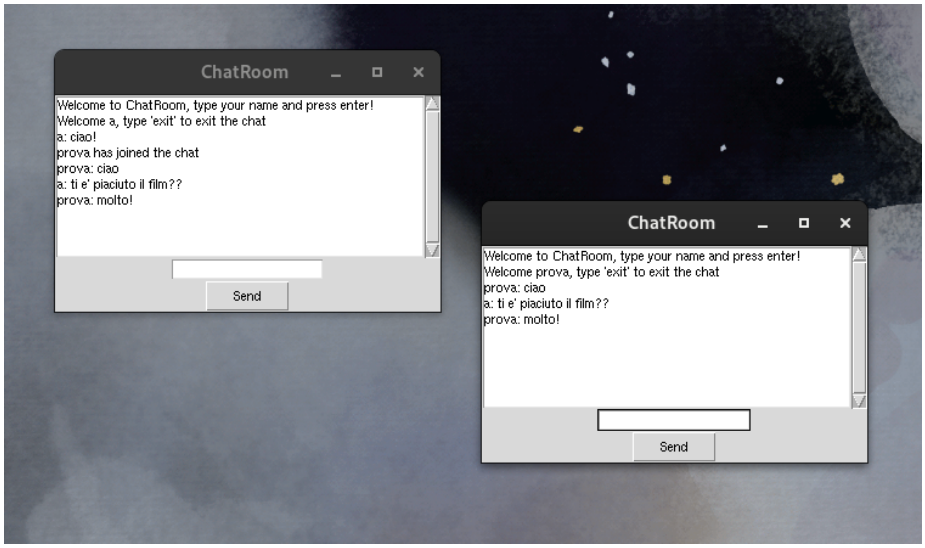

Figure 7: Chat tra utente "a" e utente "prova"

4.2 Server

4.2.1 Tofu_Server

get_film Questo metodo ritorna la lista di tutti quanti i film che sono attualmente presenti all'interno del database. Viene eseguita una query SQL e il risultato viene poi ritornato al client che procederà alla sua corretta visualizzazione. Non viene eseguito alcun tipo di aggiornamento del database.

db_update Questo meotdo si occupa dell'esecuzione delle operazioni di scraping necessario al corretto aggiornamento del database relativo ai film. Queste operazioni vengono effettuate da metodi importati dal file `movie.py`.

db_dump Il seguente metodo permette di scaricare tutto il contenuto del database relativo ai film all'interno di un file. Questa operazione viene effettuata passando al client il risultato di una interrogazione del database che prende in esame tutti quanti i film presenti al suo interno. Sarà poi il client ad occuparsi della scrittura su file dei dati raccolti.

user_registration user_registration si occupa della registrazione degli utenti all'interno del database. Sfrutta un metodo appartenente al file `db.py` per

poter registrare correttamente gli utenti, passandogli dati relativi a username, password (hashed) e lista dei film preferiti.

user_login Questo metodo si occupa del login dell'utente. Viene effettuato un controllo rispetto allo username e hash della password che sono argomenti del metodo. Qualora vi sia un match su entrambi i campi allora si ritiene che l'utente sia correttamente loggato all'interno dell'applicazione e si ritorna un valore True al client, il quale poi sarà in grado di cambiare l'interfaccia del client e considerarlo come utente loggato correttamente.

get_info_price Questo metodo invoca un metodo presente all'interno di `movie.py` che ritorna la lista delle informazioni relativi ai prezzi ed eventuali sconti o convenzioni del cinema. Tali informazioni sono raccolte in due variabili che vengono ritornate al client. Questo procederà alla loro corretta visualizzazione.

dump_fav In questo metodo si esegue una interrogazione sul database degli utenti per poter ottenere la lista dei film preferiti di un determinato utente. Il risultato di questa interrogazione viene poi trasferito al client che procederà alla scrittura su file del suddetto dato.

chat_server Questo metodo fa uso delle librerie Threading e subprocess per la gestione del server necessario al corretto funzionamento della chat. Lancia un thread che gestice il server relativo alla chat.

4.3 Metodi Aggiuntivi

In questa sezione verranno descritti i file o metodi che sono stati citati in più parti della sezione precedente. Questi non presentano una quantità di metodi tale da richiederne un elenco e verrà quindi effettuata una analisi macroscopica sul loro funzionamento.

4.3.1 Movie

Il file `movie.py` è proposto alle operazioni di scraping al fine di ottenere informazioni rispetto ai film che sono presenti in sala (oltre che informazioni rispetto a prezzi e sconti). Al suo si trovano diversi metodi che utilizzano la libreria BeatifulSoup per analizzare il codice html contenente i dati rispetto ai film che sono proiettati . Il codice html contiene le informazioni divise in due diversi gruppi:

- even
- odd

Di conseguenza ci sono due metodi differenti che effettuano lo scraping sul gruppo *even* e *odd*. Alla fine di ciascun metodo vengono svolte le operazioni di pulizia dei dati raccolti e infine di inserimento di tutti i dati all'interno del database.

4.3.2 DB

Il file `db.py` invece si occupa di tutte le operazioni relative all'inserimento di dati all'interno del database. In fase di progettazione ho ritenuto che potesse essere comodo gestire tali operazioni su un file differente rispetto a quelle del server o di scraping, al fine da mantenere una maggiore modularità in caso di sviluppi futuri dell'applicazioni o di modifica di specifiche funzionalità.

4.3.3 chat_client

`chat_client.py` altro non è che una applicazione basata su tkinter che permette di poter comunicare con tutti gli utenti che sono connessi al server chat_server.py. I messaggi vengono mandati su una porta virtuale, sui cui è in ascolto anche il server, che non fa altro che prendere i messaggi delle varie istanze di chat_client e mandarle in broadcast a tutti quelli che sono in ascolto su quella porta (con in aggiunta il relativo username del mittente).

4.3.4 chat_server

`chat_server.py` permette di poter visualizzare correttamente i messaggi che vengono inviati da chat_client per tutti quanti gli utenti che sono connessi alla chat. Il server ascolta continuamente le connessioni in ingresso, avvia un nuovo thread per ogni client connesso e gestisce la comunicazione con i client in thread

separati. I client possono inviare messaggi al server, e il server diffonde questi messaggi a tutti i client connessi. Se un client invia il messaggio di uscita (*'exit'*), viene rimosso dalla chat.

5 Sviluppi Futuri e Problemi da Risolvere

Un problema affligge ancora l'applicazione è la gestione dei threads. E' stata migliorata, ma penso si possa ancora lavorare per renderla più sicura garantendo un corretto funzionamento di ogni parte dell'applicazione. Nel momento in cui sto scrivendo questa relazione la chiusura del thread che gestisce il server della chat potrebbe essere migliorata e resa più thread-safe. Al momento non sono pienamente soddisfatto della implementazione fatta.

Un altro limite è la parte di user experience e non mi sembra di poter raggiungere risultati soddisfacenti utilizzando la libreria tkinter. Sarebbe interessante andare a sfruttare un'altra libreria, magari non basata su python. Durante il corso degli ultimi giorni ho effettuato alcuni test con GTK e sembra poter essere un candidato interessante con cui provare a implementare la GUI del programma.