SVILUPPO DI BOT TELEGRAM IN PYTHON

Leandro Bognanni, Gabriele Brenna, Niccolò Bergamaschi

Professore di Riferimento: Giovanni Agosta

INDICE

INTRODUZIONE

SCOPO DEL PROGETTO

Lo scopo del progetto è lo sviluppo di un Bot di Telegram per supportare il regolamento del Gioco di Ruolo ***Blades in the Dark.***
L'applicazione supporta svariati comandi per consentire agli utenti di svolgere una partita completa sull'applicazione Telegram ed è stata realizzata con l'utilizzo della API **python-telegram-bot.**

OBIETTIVI

Le funzionalità dell'applicazione stabilite comprendono:

- Gestione dei diversi tiri di dadi presenti nel gioco;
- Creazione e modifica delle schede dei personaggi e della crew;
- Creazione e gestione degli orologi e delle loro meccaniche;
- Gestione delle Fazioni e relative relazioni con i personaggi;
- Organizzazione delle attività di Score e Downtime;
- Redazione del diario di gioco per tenere traccia dell'avanzamento della storia e delle attività nel mondo di gioco;
- Assistenza dell'utente con suggerimenti o presentazione di possibili scelte in accordo con le regole del gioco;
- Persistenza della partita e di ogni singola conversazione e resilienza alle disconnessioni.

PROGETTAZIONE

SCELTE PROGETTUALI E ARCHITETTURA

L'architettura dell'applicazione si ispira al pattern Model-View-Controller e ne implementa le principali caratteristiche.

La prima fase di progettazione si è svolta seguendo un approccio Top-Down, inizialmente con la stesura del diagramma delle classi UML per la definizione del Modello dell'applicazione, seguito poi dalla stesura vera e propria del codice.
La seconda fase di progettazione, per la definizione del Controllore e della Vista, ha seguito un approccio misto, con una stesura iniziale dei diagrammi di sequenza per tenere traccia delle conversazioni più complesse, seguita da alcuni adattamenti successivi per sfruttare appieno le potenzialità della libreria impiegata.

COMPONENTI

- Una **base di dati** per contenere le informazioni del gioco e i dati dei singoli utenti e partite (è stata utilizzata la libreria **sqlite3** di Python).

- Appendice del programma che si occupa della **composizione dinamica di file** di tipo **PNG (**è stata utilizzata la libreria **Pillow** (PIL)).

- Costruttore del **diario di gioco** in formato **HTML** (è stata utilizzata la libreria **BeautifulSoup)**.

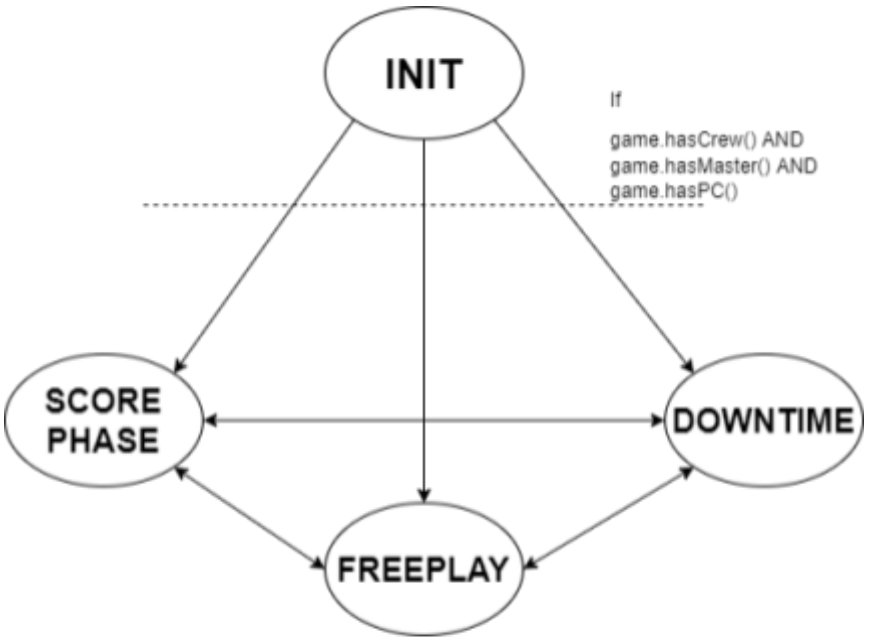
- Costruttore di una **mappa interattiva** (utilizzando **HTML5**, immagini **SVG,** codice **Javascript** con relativa libreria **jQuery**).

SVILUPPO

SCELTE IMPLEMENTATIVE

La solidità del Modello è stata comprovata attraverso un testing completo di tutti i componenti, che verifica il comportamento anche nei casi limite.

La gestione delle fasi di gioco è regolata da un attributo dell'oggetto Game che memorizza la situazione della partita. È possibile cambiare stato tramite l'apposito comando.



Stati di Gioco

Sono state definite delle classi-interfaccia per realizzare il salvataggio degli oggetti di gioco come stringhe di formato JSON e per "disegnare" le schede di gioco come immagini PNG.

La gestione della persistenza del Bot è stata realizzata utilizzando la "*PicklePersistence*" fornita dalla libreria principale, che produce un file di ridotte

dimensioni contenente tutte le informazioni che sono necessarie ad associare utenti a relative conversazioni.

I comandi sono gestiti tramite i "*ConversationHandler*", i quali possono sviluppare ulteriori conversazioni annidate se richiedono un'interazione tra utenti. è stato definito un comando */cancel* (un fallback comune a tutti i "*ConversationHandler*"), che provoca la chiusura di una conversazione, di tutte le conversazioni figlie e la cancellazione dei dati memorizzati fino a quel momento.

Alcune conversazioni fanno uso di "fallbacks" per consentire ad altri utenti di "partecipare all'azione" in modo interattivo.

Sono state ampiamente utilizzate le tastiere virtuali. In linea generale, si è deciso di impiegare la "*ReplyKeyboard*" per fornire suggerimenti di risposta all'utente, mentre le "*InlineKeyboard*" sono state usate per gestire scelte più complesse oppure con minor libertà di azione.

Per gestire la comunicazione con l'utente sono stati utilizzati i *telegram data* per simulare una sessione.
Vengono impiegati gli "*user_data"* per le conversazioni che riguardano il singolo utente, mentre i "*chat_data"* per le conversazioni che includono più di un utente.

Organizzazione dei Telegram Data

Il bot offre anche la possibilità di modificare la lingua. Sono stati prodotti due file Json per localizzare il Bot e il diario di gioco sia in inglese che in italiano.
Il bot è in grado di conversare con diversi utenti dello stesso gioco in lingue diverse.

ASSUNZIONI E COMPLETAMENTI DELLE SPECIFICHE

- ➔ Prima di poter prendere parte ad un gioco ed utilizzare le funzionalità offerte dal Bot, viene richiesto all'utente di effettuare una registrazione.

- ➔ Viene aggiunto negli *user_data* un dizionario che ha per chiavi gli identificativi delle chat nelle quali l'utente partecipa ad una partita e per valori i nomi dei personaggi attivi in ciascuna partita. In ogni momento è possibile cambiare il personaggio attivo, invocando il comando relativo nella giusta chat.

- ➔ Viene consentita la creazione di più partite nella medesima chat, tuttavia non è consentito che un utente acceda a più di una partita per chat.

- ➔ Per ragioni di riconoscimento pratico (e logico) non è consentito che due utenti partecipino ad una partita usando lo stesso personaggio.

- ➔ Si è deciso di lasciare anche al *Game Master* la possibilità di controllare uno o più personaggi. Questo rende possibile anche una partita in solitaria nella chat privata.

APPLICAZIONE

ESEGUIRE IL BOT

Il Bot può essere eseguito dall'interprete di Python (consigliata versione 3.9 o superiore).
È necessario possedere un token per Bot di Telegram e aggiungere un file token.txt all'interno della cartella resources del Progetto. La prima volta che si intende interagire con il bot è necessario avviarlo tramite il comando */start*.
A questo punto è possibile aggiungere il bot ad un gruppo o avviare una conversazione nella propria chat privata.
È necessario concedere al Bot i tutti i privilegi nell'apposita sezione nel gruppo di Telegram e nominarlo amministratore.

CONCLUSIONI

L’applicazione implementa le specifiche richieste. Si è cercato di espandere quanto più possibile le funzionalità dell’applicazione, sfruttando appieno le potenzialità offerte dalle strutture adottate.
La natura dell’operazione rispecchia la volontà di fornire un’applicazione ingegnerizzata, che possa integrare i dati di partenza con nuovi dati dall’esterno ed evolvere senza la necessità di continue modifiche o revisioni del codice.
A tal proposito, si è deciso di dare all’utente la possibilità di “personalizzare” la propria esperienza, con la possibilità di definire nuovi archetipi degli oggetti di gioco, come, ad esempio, le schede dei personaggi e della crew.
L’opportunità di fare è stata affiancata a quella di apprendere, sfruttando ogni problema per imparare a superare un ostacolo, anche con mezzi non richiesti nella fase di definizione del progetto.
L’attività di gruppo è stata sicuramente più efficace rispetto a un lavoro individuale, poiché ha permesso di confrontare, confutare e sintetizzare opinioni divergenti, con l’obiettivo comune di ottenere il miglior risultato.

DOCUMENTAZIONE AGGIUNTIVA E ALLEGATI

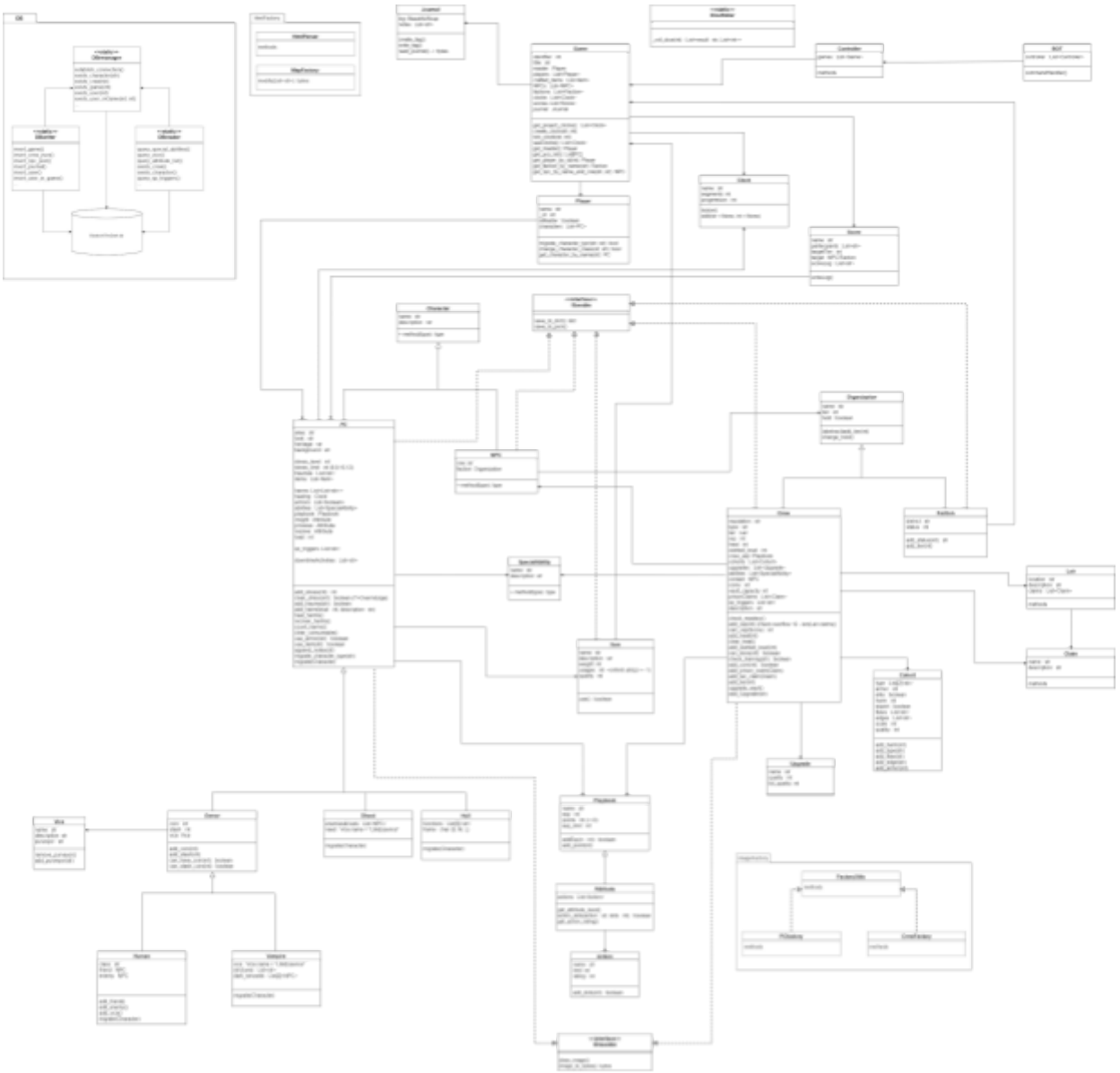


Diagramma delle classi UML

Schema Entità-Relazione