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 implementa 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

L'obiettivo cardine, che è stato prefissato, è quello di fornire un supporto che andasse oltre la semplice gestione atomica delle funzionalità, garantendo agli utenti la possibilità di tenere delle conversazioni complesse con il Bot per governare le varie casistiche e diramazioni delle scelte e applicare gli effetti stabiliti dal regolamento in modo quanto più automatico possibile.
Si è deciso comunque di lasciare libertà di azione all'utente, fornendo la possibilità di aggiustare e regolare a piacimento i parametri di ogni azione prima di effettuarla.

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

A supporto dell'applicazione si è deciso di impiegare una **base di dati** per contenere le informazioni del gioco e i dati dei singoli utenti e partite, utilizzando la libreria **sqlite3** di Python. Il database immagazzina i necessari oggetti di modello sotto forma di file JSON, che vengono aggiornati dal Controller ogniqualvolta la relativa istanza subisce una modifica.

Per la visualizzazione delle **schede dei personaggi e della crew** è stata realizzata un'appendice del programma che si occupa della **composizione dinamica di file** di tipo **PNG**, che utilizza le informazioni salvate negli oggetti del Modello.
La realizzazione di questa "factory" ha impiegato la libreria **Pillow** (PIL).

Il **diario di gioco** è stato implementato con un **file HTML**, che viene aggiornato dal Controller ogni volta che viene effettuata un'azione complessa, aggiungendo specifici tag precostruiti per organizzare le azioni e le descrizioni fornite dagli utenti. Per questa parte si è fatto uso della libreria **BeautifulSoup**.

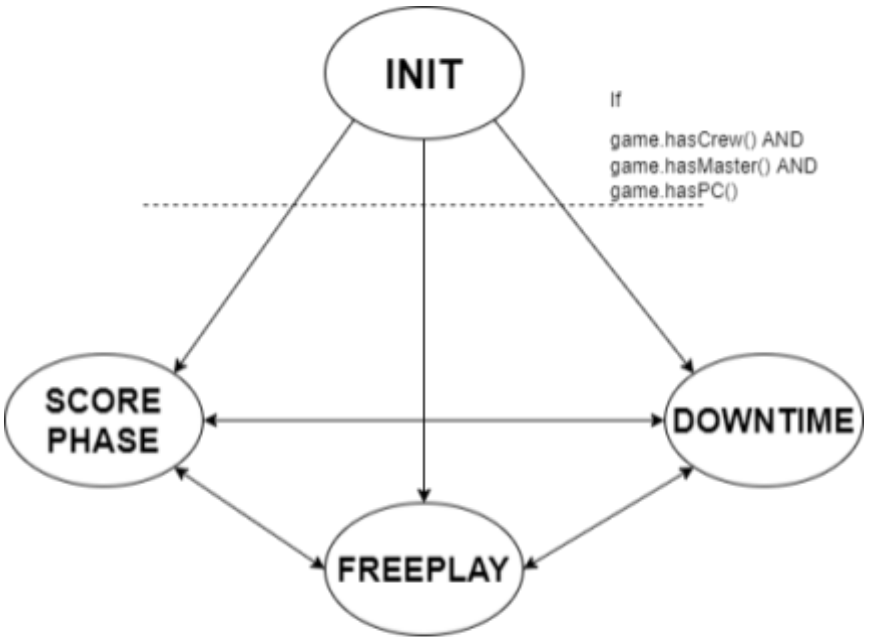
È stata anche realizzata una **mappa interattiva** del mondo di gioco in cui è possibile piazzare diversi segnaposto, utilizzando **HTML5**, immagini **SVG** e codice **Javascript** (e libreria **jQuery**)

SVILUPPO

SCELTE IMPLEMENTATIVE

Avendo deciso di lasciare all'utente quanta più libertà d'azione possibile (pur tuttavia fornendo dei "binari" per la gestione guidata degli eventi), si è reso necessario l'utilizzo di controlli all'interno del modello per impedire la manomissione di valori o il passaggio di dati non ben formati. 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. Alcuni comandi possono essere eseguiti solamente in determinati stati, ma è comunque sempre possibile cambiare stato tramite l'apposito comando.



Stati di Gioco

Per realizzare il salvataggio degli oggetti di gioco, è stata creata una classe-interfaccia che fornisce il metodo necessario per la scrittura di stringhe in formato JSON. Gli oggetti che implementano questa interfaccia possono ridefinire tale metodo per serializzare i propri attributi più complessi.

In maniera analoga, una classe-interfaccia è implementata dagli oggetti che possono essere “disegnati”, permettendo a questi ultimi di chiamare i metodi della “factory” di immagini di cui necessitano per costruire la propria scheda di gioco.

La gestione della persistenza del Bot è stata realizzata utilizzando la “*PicklePersistence*” fornita dalla libreria, che produce un file di ridotte dimensioni contenente tutte le informazioni che sono necessarie ad associare utenti a relative conversazioni. In questo modo è possibile riprendere una conversazione da dove era stata lasciata anche in caso di malfunzionamento o spegnimento volontario del Bot.

La maggior parte dei comandi sono gestiti tramite i “*ConversationHandler*”. I comandi che richiedono un’interazione tra utenti fanno uso di conversazioni annidate. In ogni caso, l’invocazione del comando */cancel* (un fallback comune a tutti i “*ConversationHandler*”) provoca la chiusura di una conversazione e 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 (un esempio è /assist che consente ad un personaggio di assistere un compagno durante un Tiro Azione).

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; un esempio è la realizzazione della tastiera per richiedere all'utente eventuali dadi bonus che possono derivare da un oggetto, un'abilità o più in generale una situazione non controllabile dal programma.

Per gestire la comunicazione con l'utente sono stati utilizzati i *telegram data* per simulare una sessione.
Nel dettaglio, quando un utente invoca un comando complesso, viene creato un dizionario atto a contenere le informazioni per la gestione della conversazione e i dati necessari al Controller per l'applicazione degli effetti e scrittura del diario di gioco.
Vengono impiegati gli "*user_data"* per le conversazioni che riguardano il singolo utente, mentre i "*chat_data"* per le conversazioni che includono (o che possono includere) più di un utente.

Organizzazione dei Telegram Data

Infine, per quanto riguarda la localizzazione linguistica, sono stati prodotti due file json, che supportano rispettivamente la lingua inglese e la lingua italiana. All'avvio della conversazione con il Bot, viene selezionata di default la lingua inglese. È possibile modificare questa impostazione con il relativo comando.
Analogamente si può anche selezionare la lingua con la quale verrà scritto il diario di gioco.

Le due opzioni sono state separate per consentire una personalizzazione completa del Bot, che può in questo modo tenere conversazioni in lingue diverse con due utenti appartenenti allo stesso gioco.

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. Questo consente all'applicazione di salvare nella base di dati l'identificativo univoco dell'account Telegram dell'utente e relazionare ogni utente alle proprie partite.

- ➔ Le regole sanciscono che un giocatore possa controllare più di un personaggio durante una partita. Per comodità d'uso e per non obbligare l'utente a specificare quale dei suoi personaggi sta eseguendo l'azione, 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 indirizzare correttamente una richiesta di un utente verso la relativa istanza di gioco, il Controller ha la necessità di inferire dalla richiesta l'identificativo della partita da modificare e, per fare questo, vengono utilizzati l'identificativo univoco dell'utente e l'identificativo univoco della chat da cui proviene la richiesta.

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

- ➔ Le regole del gioco si riferiscono al *Game Master* come "arbitro" della partita e quindi come supervisore e non come giocatore. Tuttavia si è deciso di lasciare anche a questo tipo di utente la possibilità di controllare uno o più personaggi. Questo rende possibile anche una partita in solitaria nella chat privata, nella quale l'utente è contemporaneamente *Master* e giocatore.

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. L'applicazione legge il token dal file e il Bot associato diventa in grado di ascoltare e gestire comandi e conversazioni.

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 ed è consigliabile nominarlo amministratore.
Nel caso in cui un utente venga inserito in un gruppo in cui è già presente il Bot, è previsto che la conversazione privata possa essere avviata mediante il comando */login*, che, mascherando il comando */start* con associato un payload, innesca la conversazione per gestire la registrazione dell'utente.
Una volta che tutti gli utenti sono registrati è possibile avviare la partita vera e propria.

COMANDI

Di seguito è riportata una lista dei comandi supportati con una descrizione sintetica.
La descrizione di ciascun comando è comunque disponibile tramite il comando */help* direttamente sul Bot.

	Comando	Area	Descrizione
1	/start	Generale	Avvia la conversazione con il Bot, mostra il messaggio di benvenuto, i crediti e le istruzioni per l'avvio.
2	/login	Generale	Registra un account Telegram e associa uno username all'utente.
3	/createPC	Creazione	Avvia la conversazione per creare un nuovo personaggio. Se la creazione viene terminata il risultato è salvato negli *user_data* dell'utente.
4	/createGame	Creazione	Crea una nuova partita nella chat.
5	/join	Generale	Permette all'utente di prendere parte ad una partita, sia come *Game Master* che come giocatore. Nel caso in cui si entri in partita con un personaggio, gestisce anche l'attribuzione degli ultimi attributi.
6	/createCrew	Creazione	Avvia la conversazione per creare la crew per la partita corrente.
7	/changeState	Generale	Permette di cambiare la fase di gioco. Le opzioni disponibili sono *FREEPLAY*, *SCORE PHASE* e *DOWNTIME.*
8	/myPC	Generale	Invia all'utente il file PNG con la scheda del personaggio attivo.

9	/myCrew	Generale	Invia all'utente il file PNG con la scheda della crew della partita.
10	/journal	Generale	Invia all'utente il file HTML con il diario di gioco della partita.
11	/map	Generale	Invia all'utente il file HTML con la mappa interattiva di Doskvol.
12	/changePC	Generale	Permette all'utente di cambiare il personaggio attivo della partita.
13	/createClock	Creazione	Avvia la conversazione per creare un nuovo orologio.
14	/tickClock	Generale	Permette di selezionare un orologio e aumentare il livello di completamento.
15	/addStress	Generale	Aggiunge (o rimuove) la quantità specificata di stress al personaggio attivo.
16	/addTrauma	Generale	Permette di selezionare o scrivere un nuovo trauma da aggiungere al personaggio attivo.
17	/useArmor	Generale	Marca uno slot armatura del personaggio attivo.
18	/coin	Generale	Consente di gestire i propri possedimenti, modificando le quantità di monete presenti nel borsello e nella riserva del personaggio e nel vault della crew.
19	/vaultCapacity	Generale	Permette di aumentare o ridurre manualmente le dimensioni del vault della crew.
20	/upgradeCrew	Generale	Aumenta il rango della crew se il dominio è "Forte", altrimenti riporta il valore di dominio a "Forte".
21	/factionStatus	Generale	Mostra le relazioni della crew con le altre fazioni di gioco e permette la modifica.
22	/addExp	Generale	Permette di attribuire punti esperienza agli attributi del personaggio e alla crew.
23	/addActionDots	Generale	Consente di spendere i punti azione guadagnati e assegnarli alle azioni dei rispettivi attributi.
24	/addUpgrade	Generale	Permette di modificare la lista dei potenziamenti della crew.
25	/addSpecalAbility	Generale	Aggiunge una nuova abilità speciale al personaggio attivo o alla crew.
26	/addHarm	Generale	Gestisce l'aggiunta di un nuovo danno al personaggio, permettendo di selezionare anche il livello di gravità.
27	/addHarmCohort	Generale	Gestisce l'aggiunta di un livello di danno ad una delle cohort della crew.
28	/changePurveyor	Generale	Seleziona un nuovo fornitore del vizio del personaggio attivo o sostituisce quello già presente.

29	/migrate	Generale	Modifica la natura del personaggio attivo. Consente agli Umani, ai Costrutti e ai Vampiri di diventare Fantasmi e ai Fantasmi di diventare Costrutti o Vampiri.
30	/changePCclass	Generale	Permette al personaggio attivo di cambiare classe (disponibile solo se il personaggio è Umano).
31	/addRep	Generale	Aggiunge la quantità specificata di *rep* alla crew.
32	/addArmorCohort	Generale	Aggiunge (o rimuove) un livello di armatura ad una delle cohort della crew.
33	/changeLife	Generale	Gestisce il ritiro del personaggio attivo. L'utente può decidere se il personaggio muore o se semplicemente si ritira per condurre una vita migliore.
34	/addNote	Generale	Permette all'utente di aggiungere una nota personalizzata nel diario di gioco.
35	/editNote	Generale	Consente di modificare una descrizione nel diario di gioco precedentemente inserita.
36	/changeFrameSize	Generale	Permette al personaggio attivo di modificare la dimensione della propria scocca (disponibile solo per i Costrutti).
37	/addServant	Generale	Permette al personaggio attivo di aggiungere un nuovo servitore oscuro (disponibile solo per i Vampiri).
38	/addCohort	Creazione	Aggiunge una nuova cohort alla crew e consente di sceglierne il tipo, i pregi e i difetti.
39	/addTypeCohort	Generale	Aggiunge un nuovo tipo ad una delle cohort della crew.
40	/addClaim	Creazione	Aggiunge alla crew una nuova rivendicazione, relativa al covo o alla prigione.
41	/score	Score	Inizia la preparazione di un nuovo colpo. Permette ai giocatori di unirsi con i propri personaggi e di gestire il carico.
42	/useItem	Score	Permette a un giocatore di marcare come trasportato un oggetto.
43	/flashback	Score	Avvia la conversazione del flashback per consenti
44	/endScore	Score	Chiude l'ultimo colpo attivo.
45	/payoff	Downtime	Gestisce la distribuzione delle ricchezze ottenute durante l'ultima fase di *Score.*
46	/heat	Downtime	Gestisce le conseguenze delle attività passate, calcola e aggiunge la quantità di sospetto generate alla crew.
47	/entanglement	Downtime	Tira i dadi per decidere i coinvolgimenti della crew.

48	/secretEntanglement	Downtime	Agisce come un normale /entanglement ma la descrizione non viene mostrata agli altri giocatori.
49	/downtimeActivity	Downtime	Avvia la conversazione per selezionare e svolgere un'attività di downtime per il personaggio.
50	/roll	Dadi	Lancia il numero specificato di dadi e restituisce il risultato.
51	/actionRoll	Dadi	Avvia la conversazione per eseguire un Tiro Azione.
52	/fortuneRoll	Dadi	Avvia la conversazione per eseguire un Tiro Fortuna.
53	/resistanceRoll	Dadi	Avvia la conversazione per eseguire il Tiro Resistenza.
54	/groupAction	Dadi	Avvia la conversazione per eseguire un'Azione di Gruppo.
55	/groupActionCohort	Dadi	Avvia la conversazione per eseguire un'Azione di Gruppo con una cohort della crew.
56	/incarcerationRoll	Dadi	Avvia la conversazione per eseguire un Tiro Incarcerazione.
57	/endGame	Generale	Conclude la partita e invia nella chat tutte le schede di gioco e la versione finale del diario di gioco.
58	/createItem	Personalizzazione	Consente la creazione di un nuovo oggetto
59	/createSpecialAbility	Personalizzazione	Consente la creazione di una nuova abilità speciale, sia per i personaggi che per le crew.
60	/createXpTrigger	Personalizzazione	Consente la creazione di un nuovo trigger per l'esperienza
61	/createHuntingGround	Personalizzazione	Consente la creazione di un nuovo terreno di caccia per le crew.
62	/createUpgrade	Personalizzazione	Consente la creazione di un nuovo potenziamento specifico delle crew.
63	/createNPC	Personalizzazione	Consente la creazione di un nuovo NPC. È possibile anche associarlo ad una fazione esistente
64	/createClaim	Personalizzazione	Consente la creazione di una nuova rivendicazione della crew, sia legata al covo che alla progione.
65	/createCharacterSheet	Personalizzazione	Permette la definizione di un nuovo archetipo di personaggio, selezionando tutte le caratteristiche che ne definiscono la natura. É possibile associare al nuovo personaggio anche oggetti di gioco creati dagli utenti
66	/createCrewSheet	Personalizzazione	Permette la definizione di un nuovo archetipo di crew, selezionando tutte le caratteristiche che ne definiscono la particolarità. É possibile associare allla nuova crew anche oggetti di gioco creati dagli utenti

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

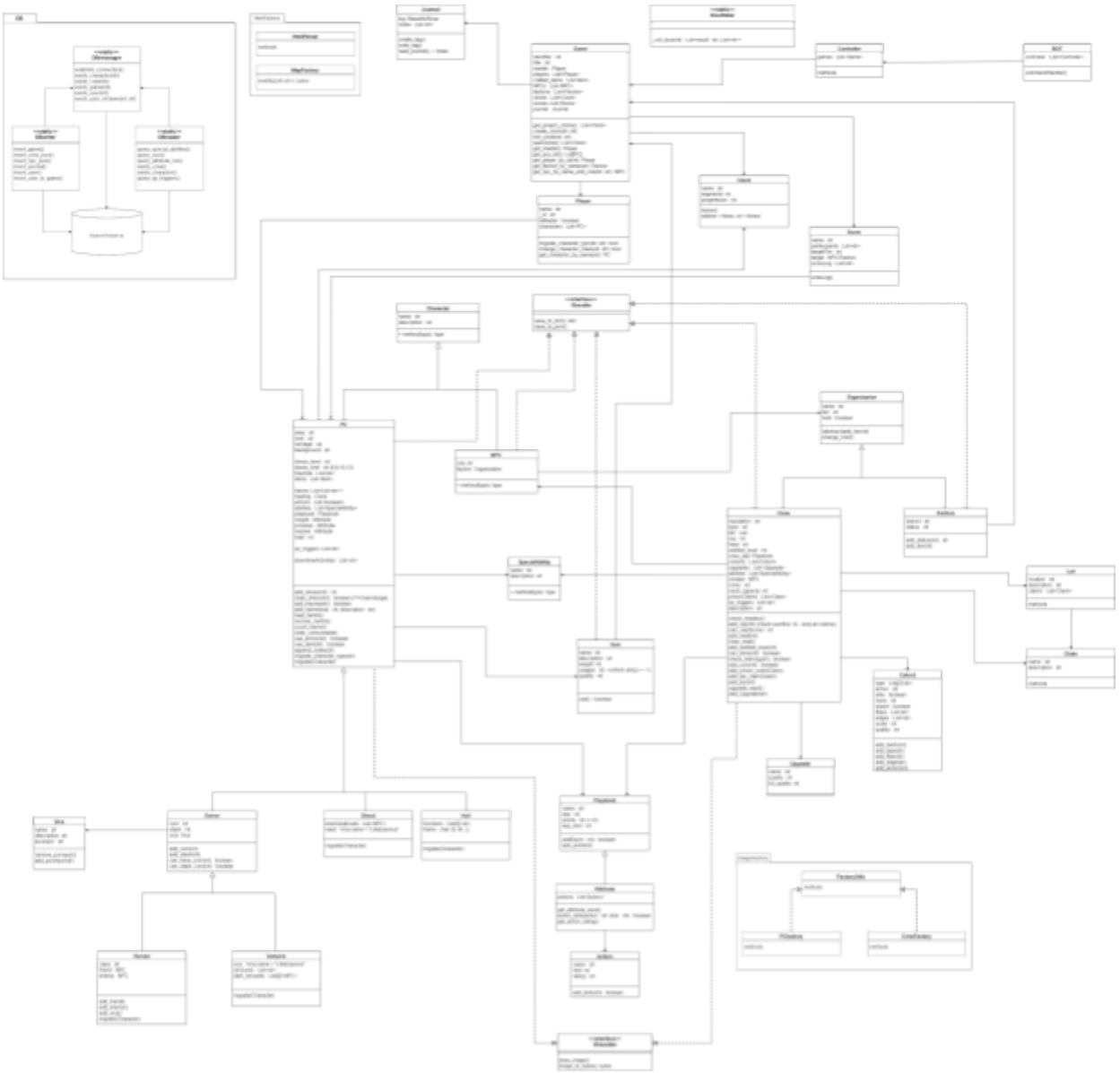


Diagramma delle classi UML

Schema Entità-Relazione