Rasanhal

PROGETTO DI TECNOLOGIE WEB

Realizzazione di una webapp per il gioco di ruolo via chat

Lorenzo Balugani 128991

Sommario

Sommario

Introduzione

Dungeons and Dragons è un gioco di ruolo fantasy nato nel 1974, nel quale i giocatori prendono parte alle avventure (o campagne) create da un membro particolare del gruppo dei player, ovvero il dungeon master.

Tipicamente, DnD viene giocato di persona, anche se in tempi recenti la cosa si è resa difficile, a causa della situazione attuale. Sebbene rimanga, anche in questo contesto, preferibile avere una comunicazione vocale tra i giocatori e il master, negli anni si è affermato – all'interno di piccole comunità – la tendenza al giocare attraverso chat testuale: è il play-by-chat (PbC). In questo scenario, le partite proseguono in modo asincrono, con il master che tira i dadi per conto dei suoi giocatori.

Lo scopo di questo progetto è di creare una webapp per il gioco di ruolo in PbC, che fornisca ai giocatori gli strumenti per comunicare, creare campagne e personaggi. Rasanhal è pensato per essere un sistema installabile all'interno di piccole comunità di giocatori, quindi facilmente gestibili da un solo amministratore.

Requisiti

Di conseguenza, l'applicazione deve essere in grado di:

1. Permettere agli amministratori di creare e gestire utenti;
2. Permettere agli amministratori di aggiungere materiale utilizzabile dai giocatori, come:

    2.1 Razze;
    2.2 Armi;
    2.3 Abilità;
    2.4 Oggetti;
    2.5 Incantesimi;

3. Permettere agli utenti di creare e gestire personaggi;
4. Permettere agli utenti di creare e gestire campagne ricoprendo il ruolo di dungeon master, in particolare:

    4.1 Aggiungere giocatori alla campagna;
    4.2 Creare e gestire gruppi di giocatori;

5. Permettere agli utenti di partecipare alle campagne, scrivendo messaggi;

Temi

Il sistema, al fine di poter rispondere ai requisiti, deve essere in grado di gestire i temi che seguono.

Utenti

Gli utenti sono le persone che prendono parte alle campagne, in veste di giocatore oppure di dungeon master: questo ruolo non è però legato all'utente, in quanto un utente può essere dungeon master di una campagna e giocatore di un'altra. Un utente deve essere in grado di creare personaggi e campagne.

Campagne

Una campagna è un'avventura creata (o almeno narrata) da parte di un utente che, relativamente a quella campagna, è il dungeon master. In alcuni casi, più dungeon master possono portare avanti la stessa campagna. Alla campagna sono legati, oltre che gli utenti, i personaggi che partecipano e i gruppi.

Gruppi

Un gruppo è un insieme di utenti che partecipano alla stessa campagna e viene creato da un dungeon master. Possono venire visti come delle sotto-divisioni stagne della chat generale, e servono per permettere al party (l'insieme dei giocatori) di dividersi. Ad ogni gruppo sono quindi collegati gli utenti che lo compongono, oltre che ai messaggi che vengono inviati. Un utente non può cambiare o meno gruppo, solo il dungeon master può farlo.

Personaggi

Un personaggio è il tramite di un giocatore all'interno della campagna, mediante il quale il giocatore interagisce con il mondo e i personaggi che il dungeon master ha creato. Un personaggio può essere presente in più campagne (ad esempio, nel caso la campagna X sia il seguito della campagna Y), ma non può comparire più volte nella stessa campagna. Un personaggio ha diverse caratteristiche di cui tenere conto, le quali sono specificate nella sezione apposita di questa tesi.

Oggetti

Un oggetto è un asset di gioco che viene inserito dall'amministratore della piattaforma. Può essere posseduto dai personaggi in diverse quantità.

Specie

Una specie è un asset di gioco che viene inserito dall'amministratore della piattaforma. Un personaggio può appartenere ad una certa specie, e in questo modo acquisire determinati tratti, come la capacità di vedere al buio oppure certi bonus alle abilità.

Classi

Una classe è un asset di gioco che viene inserito dall'amministratore della piattaforma, e rappresenta l'addestramento che un personaggio ha ricevuto: più l'addestramento è stato approfondito, più il livello relativo a quella classe è alto e più elementi vengono sbloccati relativamente a quella classe.

Abilità

Un'abilità è un asset di gioco che viene inserito dall'amministratore della piattaforma, e indica una certa area di competenza di un personaggio, come ad esempio l'essere in grado di intuire le intenzioni di qualcuno oppure di arrampicarsi sui muri. Di base, un personaggio è competente in alcune abilità, mentre in altre no: questo non vuol dire che non sia in grado di – ad esempio – addomesticare un cavallo, ma semplicemente non avrà bonus aggiuntivi quando dovrà tirare il dado.

Incantesimi

Un incantesimo è un asset di gioco che viene inserito dall'amministratore della piattaforma. Un incantesimo è qualcosa che solo personaggi addestrati all'uso della magia possono usare.

Glossario

Dal momento che verranno spesso utilizzati termini alternativi a quelli usati finora per indicare alcuni componenti del software, segue un glossario:

Termine	**Sinonimi**
Utente	Giocatore, player, dungeon master, DM
Campagna	Avventura, partita, sessione
Personaggio	PG
Gruppo	Party, chat
Oggetto	Item, loot
Incantesimo	Spell, magia
Classe	Ruolo, specializzazione
Abilità	Skill
Dungeons And Dragons	DnD, D&D

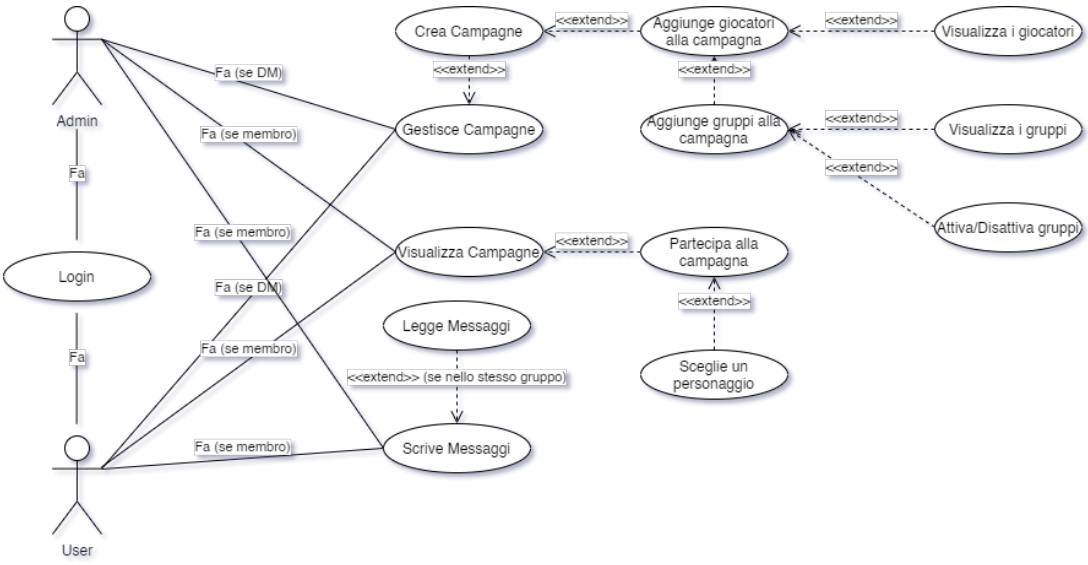
Descrizione del progetto

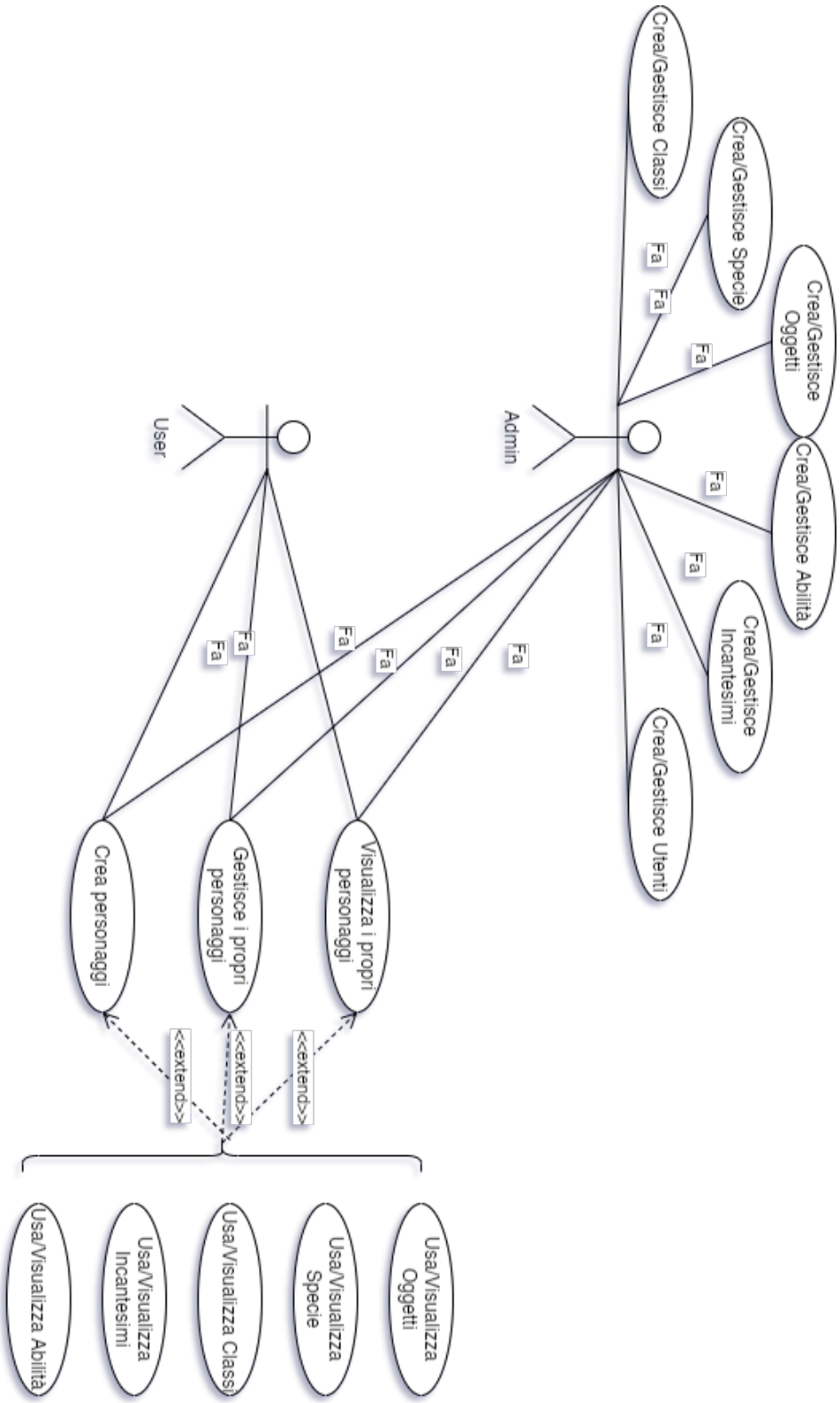
Tipologie di utenti e permessi

Gli utenti all'interno di Rasanhal sono divisi in due categorie:

- Admin, ovvero gli utenti che possono accedere al pannello amministrativo. Possono, oltre a quello che possono fare gli utenti normali:
  - Creare Oggetti, Classi, Specie, Abilità, Incantesimi, Utenti;
- User, i normali utenti della piattaforma. Possono:
  - Creare/gestire Campagne di loro proprietà. Questo implica che:
    - Possono creare/isolare gruppi, e aggiungere utenti ad essi;
    - Possono aggiungere utenti alla campagna;
  - Visualizzare e partecipare alle campagne condivise con loro (o di cui sono proprietari);
  - Creare/gestire personaggi. Questo implica che:
    - Possono aggiungere specie, classi, abilità, oggetti, incantesimi ai loro personaggi;
    - Possono aggiungerli alle campagne a cui partecipano;
  - Possono scrivere messaggi all'interno del gruppo attivo all'interno di una campagna a cui partecipano;

Quanto specificato in questo paragrafo viene riassunto all'interno dello schema UML che segue.



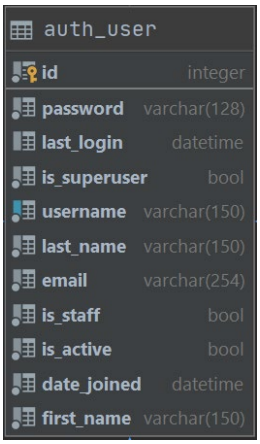

Crea/Gestisce Classi
Crea/Gestisce Specie
Crea/Gestisce Oggetti
Crea/Gestisce Abilità
Crea/Gestisce Incantesimi
Crea/Gestisce Utenti
Admin
User
Fa
Crea personaggi
Gestisce i propri personaggi
Visualizza i propri personaggi
<<extend>>
Usa/Visualizza Abilità
Usa/Visualizza Incantesimi
Usa/Visualizza Classi
Usa/Visualizza Specie
Usa/Visualizza Oggetti

Database e classi di modelli

E'stato scelto di impiegare come database sqlite, principalmente per rendere più semplice la consegna del progetto. Detto questo, niente vieta, in futuro, di spostarsi su Postgres o altro DBMS. Partendo dai temi trattati, è stata eseguita la progettazione della base di dati, che ha portato alcune modifiche rispetto a quella ipotizzata nel progetto di Basi di Dati.

Le tabelle/modelli che sono risultati sono riportati di seguito.

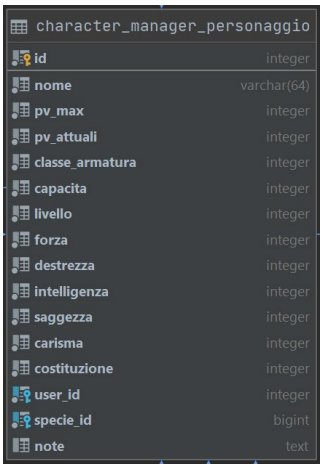
Elementi principali



User

Per il modello user è stato utilizzato quello presente all'interno dell'App django Auth. User è collegato a Campagna tramite la tabella ponte Partecipa (n:m), a Personaggio (1:n), a Messaggio (1:n) e Gruppo passando per la tabella ponte Gruppo_Users (n:m)

Personaggio

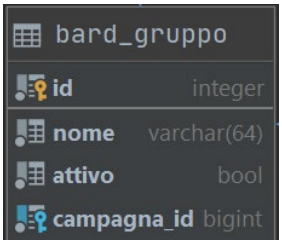


Il modello Personaggio contiene tutte le caratteristiche generali che un personaggio all'interno della 5° edizione di DnD, che sono riportati qua a lato. Personaggio è legato allo User che l'ha creato (1:n lato User), alla sua specie (1:n lato Specie) e ad altri elementi tramite tabelle ponte, come Incantesimi (tramite Lancia), Oggetti (tramite Possiede), Classi (tramite IstruitoA) e Abilità (tramite SaFare). Personaggio è inoltre legato a Campagna tramite una tabella ponte.

Campagna

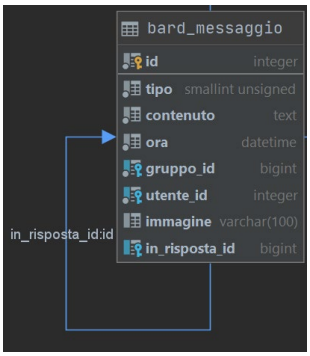
Il modello campagna contiene l'id, il titolo e la descrizione della campagna. Le informazioni sull'appartenenza di un utente o di un personaggio ad una campagna è salvata all'interno delle tabelle ponte Partecipa e Campagna_Personaggi. Campagna è inoltre collegata a Gruppo (1:n lato Campagna).

Gruppo



Il modello gruppo contiene l'id, il titolo del gruppo e un flag booleano che indica se il gruppo è "attivato" o meno. Se il gruppo risulta attivato, gli utenti che ne fanno parte verranno automaticamente inseriti al suo

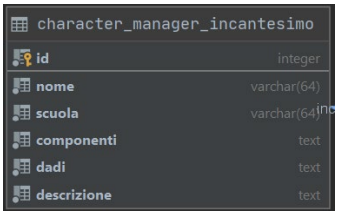
interno quando si collegano alla chat. Gruppo è collegato a Campagna, a User (tramite tabella ponte) e a Messaggi (1:n dal lato di Gruppo).

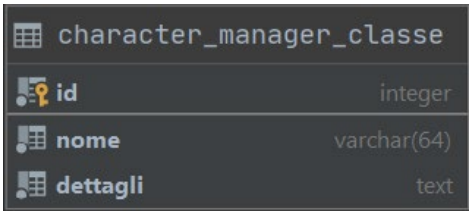


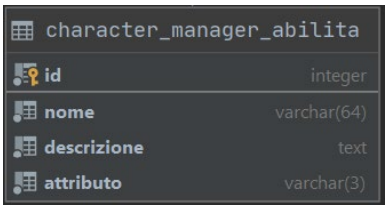
Messaggio

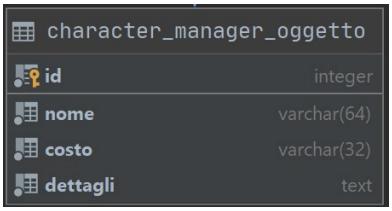
Il modello messaggio contiene l'id del messaggio, il tipo di messaggio (per sviluppi futuri), il contenuto, l'ora, riferimenti al gruppo di appartenenza, l'autore e un eventuale messaggio a cui fa riferimento e un campo di tipo ImageField. Al suo interno è contenuto il percorso delle immagini che vengono inviate.

Tabelle degli Asset di gioco



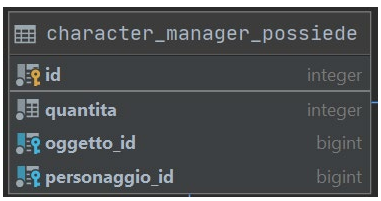


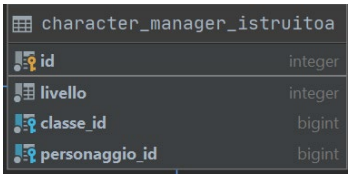


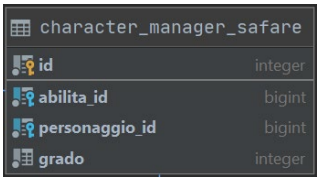


Questi modelli costituiscono il set di regole/asset che i giocatori possono assegnare ai loro personaggi. Per questa ragione, possono essere modificati, aggiunti o eliminati soltanto dagli amministratori della piattaforma.

Tabelle ponte con attributi

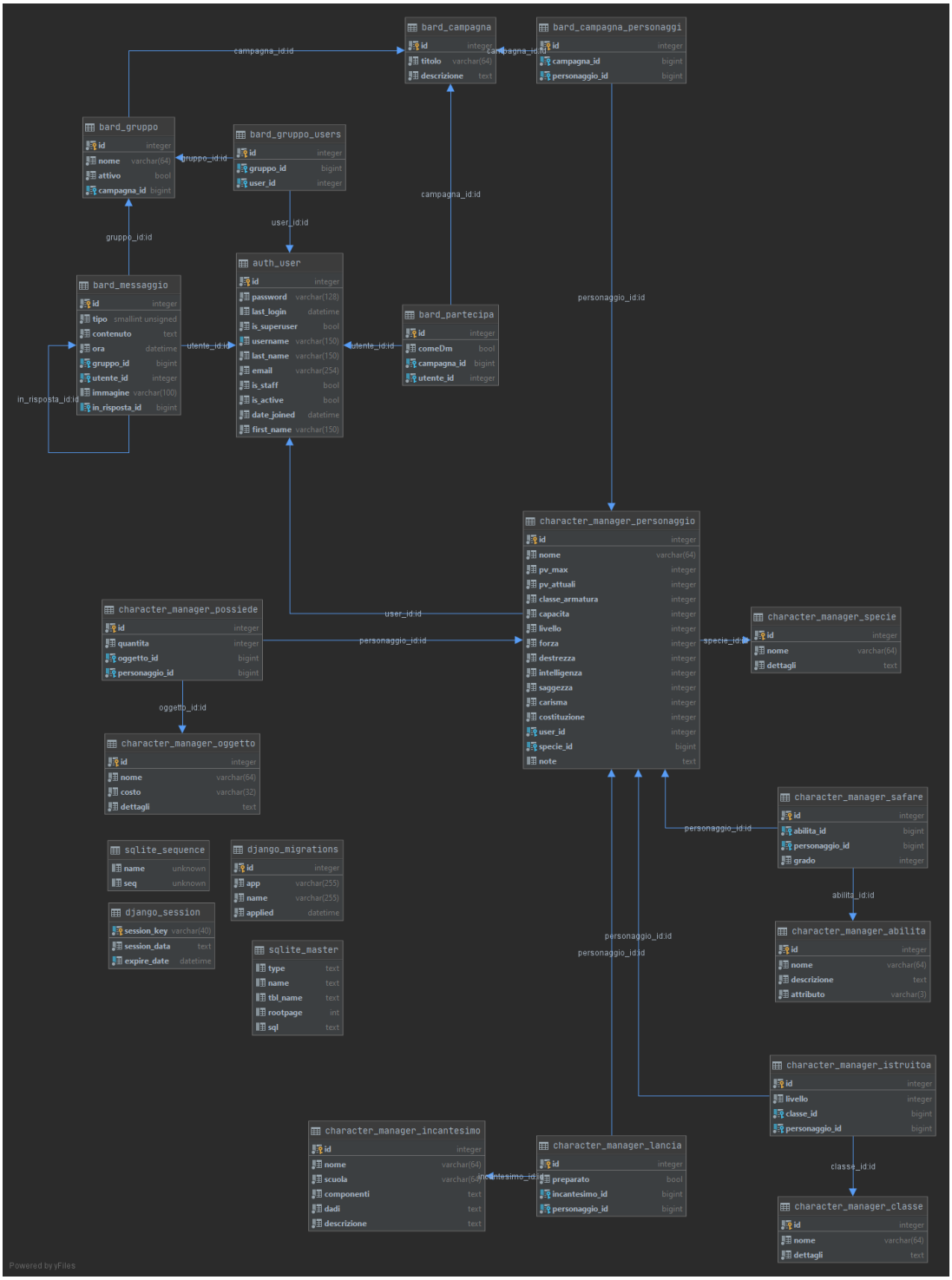






Descrizione del progetto

Schema completo



Questo schema contiene solo le tabelle usate dal progetto, omettendo quelle non utilizzate dell'App Auth.

Flussi d'uso dell'applicazione

Sono di seguito elencati i flussi che l'utente segue durante le sue attività all'interno dell'applicazione:

- Aggiunta personaggio
  - Accesso all'applicazione;
  - Click su "+" di fianco a Lista Personaggi;
  - Inserimento delle generalità del personaggio e degli elementi legati al personaggio (Abilità, Oggetti, Incantesimi…);
  - Click su salva;
- Modifica personaggio
  - Accesso all'applicazione;
  - Click sull'icona di modifica del personaggio;
  - Modifica del personaggio tramite i controlli appositi;
  - Click su salva o elimina (nel caso in cui la modifica consista nel cancellare il personaggio);
- Creazione campagna
  - Accesso all'applicazione;
  - Click su "+" di fianco a Lista Campagne;
  - Inserimento delle generalità della campagna, degli utenti e dei gruppi;
  - Click su salva;
- Modifica campagna
  - Accesso all'applicazione;
  - Click sulla campagna desiderata di cui si è DM;
  - Click su "Modifica campagna";
  - Modifica della campagna tramite i controlli appositi;
  - Click su salva o elimina (nel caso in cui la modifica consista nel cancellare la campagna);
- Partecipare alla campagna
  - Accesso all'applicazione;
  - Click sulla campagna desiderata. L'applicazione connetterà automaticamente l'utente al websocket della campagna, sul quale vengono trasmesse le comunicazioni di cambio canale della chat, e se è definito un canale predefinito a cui l'utente può accedere viene collegato il websocket appropriato;
  - Se l'utente è DM:

        - Può scrivere messaggi, spostare/spostarsi tra gruppi e consultare le schede dei personaggi dei giocatori;
    - Se l'utente non è DM:
        - Può scrivere messaggi, consultare le proprie schede dei personaggi e assegnare un personaggio alla campagna;

Tecnologie utilizzate

Backend

Per il backend si è scelto di utilizzare il Framework Web visto durante il corso di Tecnologie Web, in particolare Rasanhal si basa sul modulo REST di Django, Django Rest Framework. La scelta di questo modulo, vista la scelta del framework, è stata praticamente obbligata, in quanto è la soluzione go-to per realizzare una restapi con Django.
Lato python, altri moduli di significativa importanza sono simple-jwt, che offre maggior controllo sui token JWT (che vengono utilizzati dal Backend per l'autenticazione) rispetto a quello fornito da DRF, e channels, che permette di creare websocket all'interno di un'applicazione Django.

Comunicazione

Per il momento, le comunicazioni tra frontend e backend avvengono tramite http e non https, il che non rende ideale il sistema dal punto di vista della sicurezza. Nulla impedisce, in fase di deployment, di incapsulare le comunicazioni tramite https.
La funzionalità di chat fa uso dei websocket, e anche in questo caso la scelta della tecnologia è stata obbligata: i websocket permettono un aggiornamento continuo delle informazioni tra frontend e backend, e per i messaggi di chat e le comunicazioni di cambio canale questa caratteristica è fondamentale.

Frontend

Il frontend è stato sviluppato utilizzando React. React è stato scelto in quanto permette di creare frontend responsivi ed altamente modulari, ed è un'estensione ottima al linguaggio Javascript che lo rende estremamente flessibile e facile da utilizzare. Il modulo npm più utilizzato è quello di React-Bootstrap, che permette di portare gli elementi di Bootstrap all'interno della propria webapp.

Organizzazione del software e scelte

Organizzazione del software

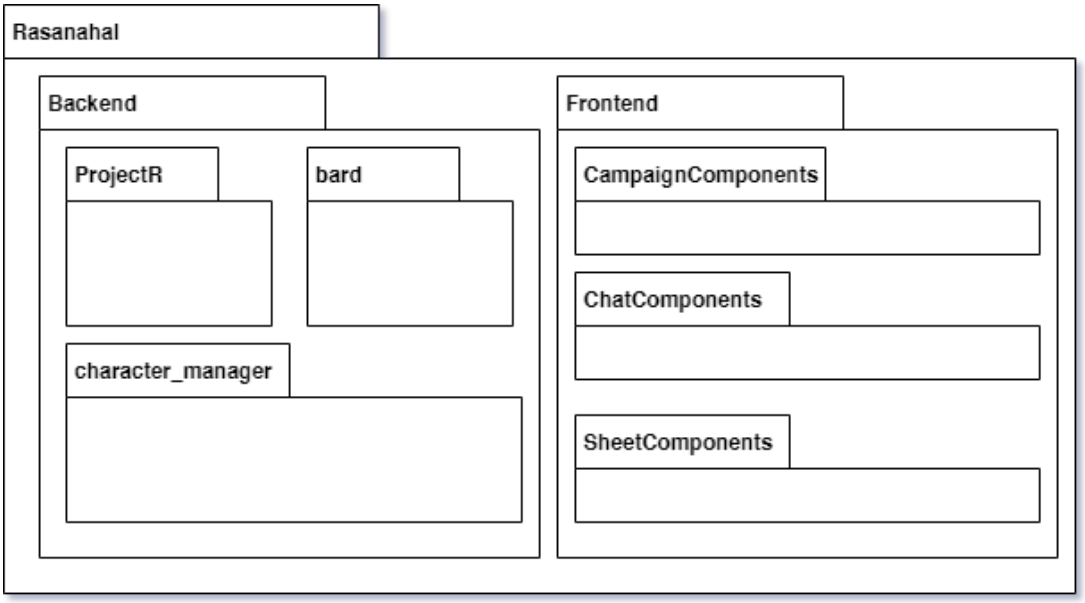
Il progetto è diviso, ad alto livello, in due parti, le quali sono divise ulteriormente per fornire maggiore modularità:

- Backend
    - ProjectR, il progetto django. Consente di autenticarsi presso l'applicazione. E' a sua volta diviso in due parti, al fine di consentire modularità: non è detto che le funzionalità di chat interessino a tutti coloro che potrebbero essere interessati a Rasanhal, in quanto potrebbero solo volere un sistema per tenere traccia dei propri personaggi, e tenendo queste funzioni in moduli separati si permette di disattivare queste funzioni, nel caso non siano necessarie. Le applicazioni di cui è composto il progetto sono:
        - Character_manager
            - Character_manager è l'app dedicata alla creazione dei personaggi. Contiene i metodi per accedere a:
                - Personaggi
                - Utenti
                - Oggetti
                - Specie
                - Abilità
                - Incantesimi
                - Classi
        - Bard
            - Bard è l'app dedicata alla gestione e allo svolgimento delle campagne. Contiene i metodi per accedere a:
                - Campagne
                - Gruppi
                - Messaggi
                - Collegamenti con le tabelle di character-builder
            - Bard, inoltre, contiene la componente che gestisce la comunicazione via websocket.
- Frontend
    - Il frontend è sviluppato in React, ed ogni macro-componente della webapp è contenuta in un modulo javascript separato dagli altri. Nella cartella dei Containers, insieme agli altri moduli, sono presenti 3 cartelle:
        - CampaignComponents: contiene i componenti legati al sistema della creazione di campagne.

- ChatComponents: contiene i componenti utilizzati nel sistema di chat in tempo reale.
- SheetComponents: contiene i componenti utilizzati nel sistema di creazione del personaggio

- Elementi dei vari componenti vengono utilizzati dai diversi moduli del frontend, sfruttando la natura modulare di React. Questo ha consentito di ottimizzare i tempi di sviluppo.

Il diagramma riportato di seguito riassume la divisione in moduli del progetto:



Scelte di sviluppo

Backend

Per la realizzazione degli endpoint dell'API è stato deciso di utilizzare le ModelViewSets del Django Rest Framework, in quanto fortemente consigliato dalla documentazione ufficiale di Django Rest Framework. La scelta si è rivelata azzeccata, in quanto ha permesso di ridurre notevolmente i tempi di sviluppo lato backend.

Per l'autenticazione la scelta è ricaduta sui Json Web Token, dal momento che non richiedono l'uso dei meccanismi di sessione (un API dovrebbe essere stateless), non richiedono l'uso di cookie e il sistema di autenticazione può facilmente venire decentralizzato, nel caso in cui ce ne dovesse essere la necessità. L'uso di JWT, inoltre, consentirebbe facilmente l'integrazione con bot di telegram o discord.

Comunicazione

Il sistema di Chat ha rappresentato una notevole sfida. Al fine di soddisfare il progetto iniziale (diversi canali in grado di venire isolati dal dungeon master), sono stati predisposti due websocket: uno che trasmette comunicazioni sul cambio dei canali, su cui solo il master può inviare messaggi, e uno di comunicazione all'interno del canale, tramite il quale i giocatori e il master possono comunicare.

Frontend

I componenti react non sono class-based all'interno di Rasanhal, ma sono componenti funzionali. Questo approccio è fortemente consigliato dalla documentazione di react stessa, e i motivi sono svariati: il più significativo è che sono più semplici da comprendere, dal momento che è, fondamentalmente, Javascript con alcune particolarità e alcune funzioni in più fornite da React.

Gestione moduli

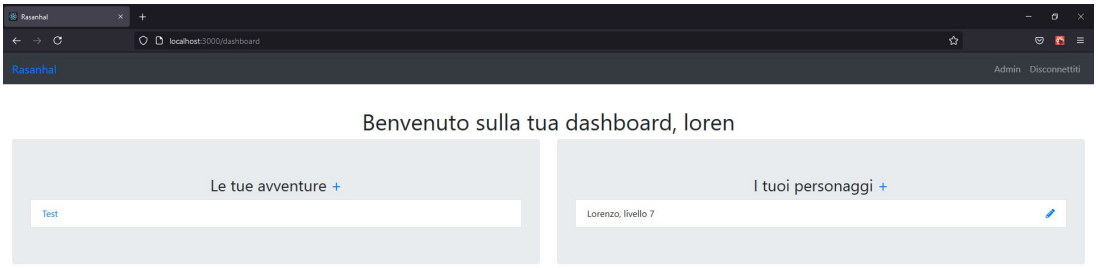
Per la gestione dei moduli Python, al posto di pipenv, è stato utilizzato Poetry. Entrambe le soluzioni sono ottime per la gestione dei moduli, ma Poetry consente di pubblicare i pacchetti su PyPy in modo estremamente semplice. In futuro, Rasanhal potrebbe diventare un modulo python facilmente installabile e configurabile. Poetry è, inoltre, più veloce di Pipenv in molti scenari.

Test svolti

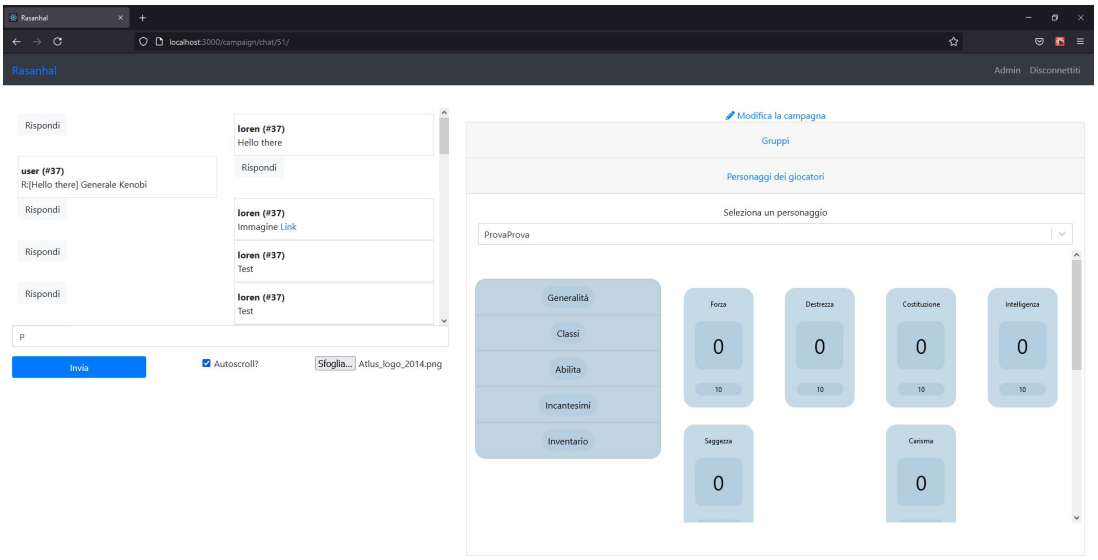
Per effettuare i test sul backend di Rasanhal è stata utilizzata la classe TestCase fornita da Django, unita ad alcuni strumenti legati a Django Rest Framework, come APIRequestFactory e APIClient. I test creati sono i seguenti:

- Character_manager tests
  - UserTest
    - Test_get_data_authorized: verifica che un utente autenticato possa accedere ai propri dati.
    - Test_get_data_unauthorized: verifica che un utente anonimo non possa accedere ai propri dati.
  - CharacterTest
    - Test_create_character_no_auth: verifica che un utente anonimo non possa creare personaggi.
    - Test_create_character_and_retrieve: verifica che un utente autenticato possa creare personaggi, ottenere le loro informazioni e cancellarli
    - Test_create_character_and_manage_as_other: verifica che un personaggio creato da un utente non possa venire modificato/letto/eliminato.
    - Test_character_validity: verifica il corretto funzionamento di un metodo della classe Personaggio, che verifica che alcuni valori rimangano entro soglie accettabili.
- Bard tests:
  - CampaignTest
    - Test_user_create_campaign_and_retrieve: verifica che un utente autenticato possa creare una campagna, ottenere i suoi dati ed eliminarla.
    - Test_user_not_authorized: verifica che un utente non autenticato non possa interagire in alcun modo con le campagne (creazione/lettura/rimozione).
    - Test_user_impostor: verifica che un utente non possa accedere e modificare una campagna a cui non ha accesso.

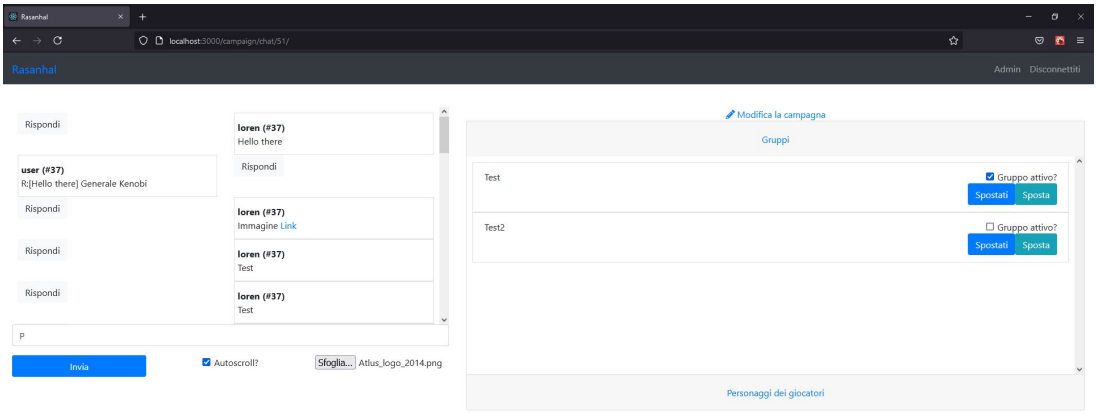
Risultati



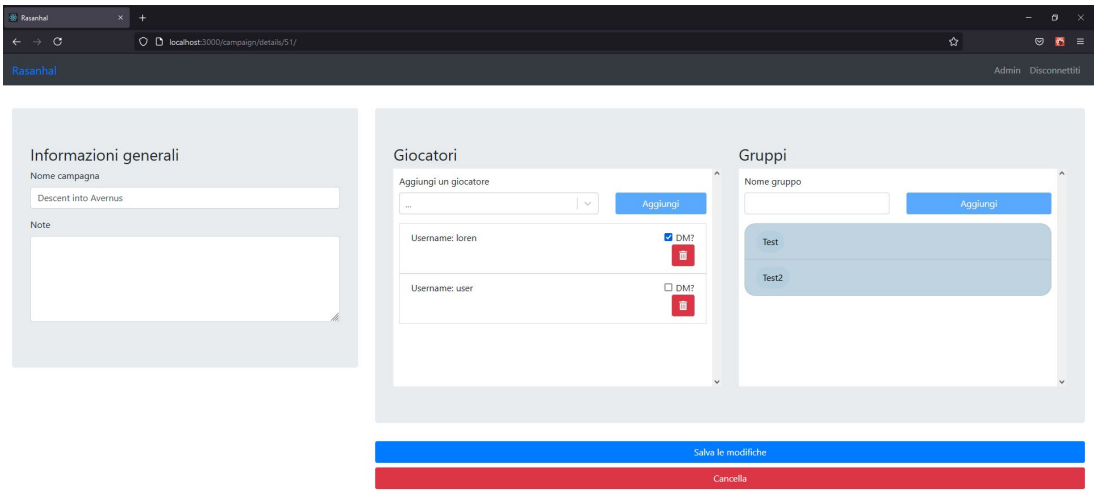
La Dashboard di Rasanhal.



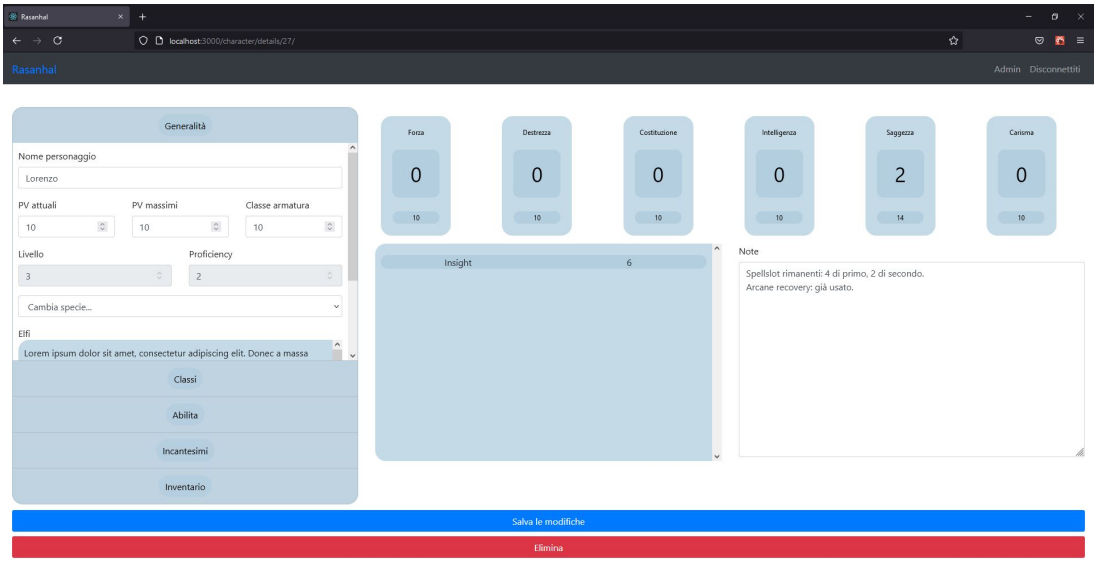
La vista del master sulla chat, con aperto il pannello per la consultazione delle schede dei personaggi dei giocatori.



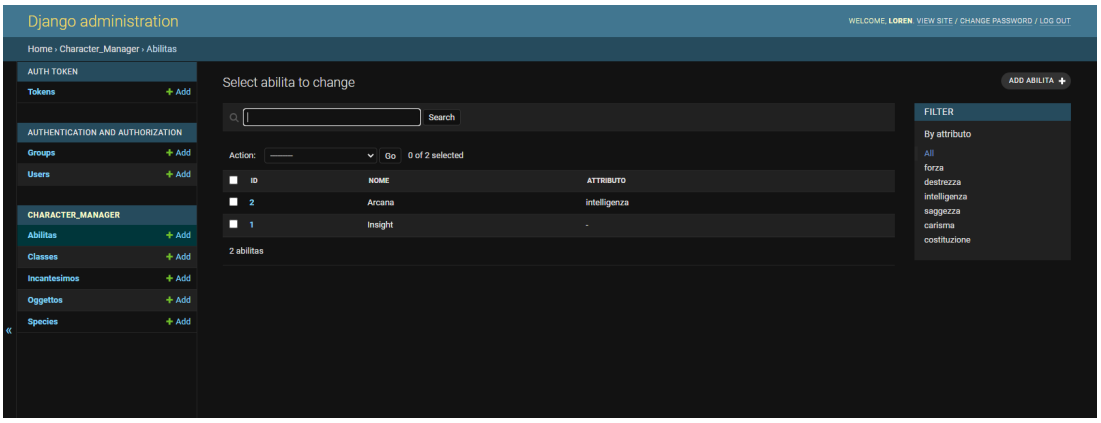
La vista del master sulla chat, con aperto il pannello per la gestione dei gruppi.



Il pannello per la gestione della campagna.



Il wizard per la creazione dei personaggi.



Il pannello amministrativo di Django, con funzioni di ricerca e filtri.

Istruzioni per l'installazione

1. Installare Poetry. Per maggiori informazioni, https://python-poetry.org/docs/
2. Installare nodejs, versione 14.17.0 LTS (https://nodejs.org/en/). Assicurarsi di aggiungere nodejs e npm al PATH.
3. Nella root del progetto, eseguire poetry install da terminale.
    a. Per attivare l'ambiente virtuale di poetry, eseguire poetry shell
    b. Per avviare il backend, python manage.py runserver
4. Spostarsi nella cartella frontend, e eseguire npm install serve
    a. Per avviare il frondend, npx serve -s build
5. Collegarsi a localhost:5000 per accedere al frontend.
    a. Nella schermata di login verranno chieste delle credenziali:
        i. Indirizzo server: http://localhost:8000 (indirizzo del backend)
        ii. Per utente amministratore, username: loren password: admin
        iii. Per utente standard, username: user password: unimore2021

Problemi riscontrati

I problemi riscontrati durante lo svolgimento del progetto sono principalmente legati alla mia scelta di voler sviluppare una webapp con React. E' stata la prima volta che ho scritto un backend con Django Rest Framework, ed è stata anche la mia prima esperienza con React. Ho dovuto quindi imparare rapidamente ad usare questi strumenti, e ho lavorato non-stop al progetto a partire dalla data della sua approvazione.

Sono soddisfatto del risultato finale, sia perché l'applicazione funziona come dovrebbe, sia perché è stata una sfida stimolante che mi ha portato a migliorare la mia conoscenza di Django e React.