Documentazione progetto Programmazione Avanzata 2023-2024
MyAnimeTracker
Edoardo Cremente – Mat. 564817

1) Caso d'uso

MyAnimeTracker è un'applicazione desktop che permette di tenere traccia di una lista di anime guardati nell'ultimo periodo, permettendo di associare a ognuno di essi un voto personale nella forma di punteggio numerico, e l'aggiunta di note personalizzate. Gli anime tra cui si può scegliere sono quelli attualmente presenti nella top 100 del sito MyAnimeList.
All'avvio dell'applicazione ci si trova davanti a una schermata di Login, da cui si può scegliere se inserire i propri dati[1] o registrare un nuovo account, cliccando sul tasto apposito. In questo caso si viene rimandati a una pagina di Registrazione in cui inserire i propri dati. Gli username degli utenti devono essere univoci, per cui non viene accettato uno Username se è già in uso da qualcun altro (un messaggio d'errore ci avviserà se ciò accade). Una volta completata la registrazione possiamo tornare alla schermata di Login ed effettuarlo.
La pagina che si apre è una pagina contenente la top 100 attuale degli anime al momento. Vengono mostrati su una tabella sulla destra dello schermo con varie informazioni sull'anime, tra cui titolo, numero di episodi, score... Se si clicca su un anime della tabella, alla sua sinistra appare una copertina rappresentativa dell'anime. A sinistra di quest'ultima, compare un'immagine rappresentativa del fatto che l'anime selezionato sia stato inserito oppure no all'interno della propria lista personale[2]. L'immagine che appare è casuale (ci sono 6 immagini che simboleggiano "sì" e 6 immagini che simboleggiano "no").
Se vogliamo aggiungere un anime alla lista personale degli anime guardati basterà, dopo averlo selezionato dalla tabella, premere il tasto "Aggiungi alla mia lista". In qualsiasi momento è anche possibile effettuare il Logout e tornare alla schermata iniziale di Login.
Una volta soddisfatti delle nostre scelte, premendo il tasto "Vai alla mia lista", è possibile spostarsi in una pagina contenente la propria lista personale. Quest'ultima è simile alla pagina della top 100, con la differenza che la tabella a schermo contiene solamente gli anime che abbiamo selezionato dalla tabella precedente, e possiamo vedere ora una maniera per assegnare a ognuno di essi un punteggio e delle note addizionali, nei rispettivi riquadri sulla destra della pagina. Una volta cliccato su un anime nella tabella, possiamo cominciare a modificare i due riquadri (i valori devono essere appropriati. In caso contrario, un messaggio di errore ci avviserà del fatto che non lo sono. Successivamente discuterò di alcune scelte progettuali che per scopo didattico sono state semplificate), per poi premere su "Salva modifiche" una volta soddisfatti. Dopo averlo fatto, possiamo cliccare altrove, e assicurarci che una volta cliccato sull'anime appena modificato, i dati che abbiamo inserito ricompaiano nel rispettivi riquadri. Se si decide di rimuovere totalmente un anime dalla lista degli anime personali, basta premere con il tasto destro su uno degli anime della tabella e cliccare su "Rimuovi". I dati salvati sull'anime verranno eliminati e, nel caso lo si volesse aggiungere nuovamente, si dovrà seguire di nuovo la normale procedura dalla pagina della top 100. Anche da questa pagina è possibile effettuare il logout in ogni momento.
In qualsiasi istante durante la navigazione, è possibile cambiare il linguaggio dell'applicazione a scelta fra 3 distinte lingue (Italiano, Inglese e Cinese semplificato).

[1] Nel database è già stato inserito un utente di prova per poter effettuare il login senza effettuare la registrazione. Nome utente: test. Password: test.
[2] Per l'utente "test", è già stato inserito il primo anime in classifica nella sua lista personale, così da poterne testare le funzionalità senza doverne manualmente aggiungere uno.

2) Scelte progettuali

Per costruire il progetto mi sono basato sull'uso di un database. Al primo avvio del servizio (ServizioProgetto), se il database non è presente, viene creato con il nome corrispondente al mio numero di matricola. Il modo in cui viene creato è la seguente riga di codice nel file application.properties:

spring.datasource.url=jdbc:mysql://${MYSQL_HOST:127.0.0.1}:3306/564817?createDatabaseIfNotExist=true

Una volta creato il database al primo avvio, questo viene popolato attraverso una richiesta get verso l'API che mi fornisce i dati dei primi 100 anime, per poi inserire anche l'utente test nella tabella degli utenti, e il primo anime in classifica nella lista degli anime guardati dall'utente test. Se il database esiste già ed è già stato popolato, tutta questa fase d'inizializzazione viene saltata. Per sapere se è stata eseguita o no, bisogna porre attenzione a una riga di log che appare alla fine dell'apertura del servizio. Nel caso il database non ci fosse, si potrà leggere "Il database inizialmente vuoto è stato inizializzato". In caso contrario (database già esistente) si avrà modo di leggere "Il database è già inizializzato, perciò non è stato generato".

Esclusa questa prima fase di inizializzazione, tutti gli accessi al database e la creazione delle tabelle viene gestita/effettuata attraverso i metodi visti a lezione, con classi Java bean e l'utilizzo di interfacce che estendono CrudRepository.
Per lo scambio di dati tra l'applicazione e il servizio, ho optato per una codifica JSON. La scelta è stata fatta principalmente perché il formato JSON mi è sembrato più semplice da implementare, e soprattutto ultimamente nel panorama informatico sembra avere più consenso generale rispetto all'XML. Ho comunque utilizzato file XML (come visto a lezione) per gestire la traduzione dell'applicazione in più lingue, ma non per lo scambio di dati.

Per quanto riguarda i rimanenti requisiti obbligatori del progetto, abbiamo:

- La lista di elementi e menù a scomparsa per la cancellazione degli elementi

La lista è presente sia nella pagina della top 100 che nella pagina degli anime personali, mentre il menù a tendina per la rimozione di un anime è presente solo nella pagina della lista personale, dalla quale si può decidere di rimuovere a piacimento gli elementi.

- La presenza di un Unit test

È stato aggiunto uno Unit test che esegue una richiesta mock verso il database, e in particolare si assicura che nella tabella della top 100 degli anime, ogni anime abbia almeno un id. Questo è importante per due motivi: in primis per assicurarsi che gli eventuali inserimenti nella tabella Anime non vengano manipolati in qualche modo da qualche richiesta inaspettata che potrebbe modificarne il formato, e poi anche perché per come è progettata la pagina, l'id dell'anime corrisponde di fatto alla sua posizione del ranking per come vengono estratti i dati dall'API. Un record della tabella senza il campo id porterebbe sicuramente a problemi nella gestione delle pagine e tutto ciò che ne consegue.

Alcune ultime scelte implementative che reputo importanti da tenere a mente sono:

- L'inserimento dei dati degli utenti nel database non avviene in modo criptato, per semplicità
- Anche se è possibile cambiare la lingua in cinese, nel campo delle note degli anime nella pagina della propria lista non è possibile inserire caratteri che non siano alfanumerici e facenti parte della punteggiatura più comune. Per effettuare i controlli ho utilizzato una semplice espressione regolare, che non permette inserimento di simboli e/o lettere accentate. Questo è stato fatto per semplificare lo scambio di dati con il Database, dal momento che non ho trovato un metodo comodo ed efficiente per effettuare scambio di dati con caratteri orientali.
- Per le richieste http, è stata creata una classe Utility che si occupa di inviare le richieste, divise in 4 categorie:
    - Richiesta get che legge un json in risposta
    - Richiesta post che manda una stringa e legge un json in risposta
    - Richiesta post che manda un json e legge una stringa in risposta
    - Richiesta post che manda un json e legge un json in risposta

- La classe FinishedCell è stata utilizzata esclusivamente per implementare la funzionalità di convertire una cella di una tabella in una imageview, per poter inserire piccole immagini di simboli all'interno delle celle.
- Per semplicità e compattezza, lato servizio, è stato usato un solo controller per gestire tutti i tipi di richieste dalle varie pagine.
- Nelle resources dell'applicazione, oltre ai file fxml e alle cartelle contenenti le immagini e i file per le lingue, è presente un file css che ho pensato di aggiungere per rendere l'applicazione più gradevole alla vista e sperimentare un po' con il css applicato a javafx, argomento non trattato a lezione.