

# Guran Turino Anime List

Lorenzo Valtriani

February 9, 2023

Progetto di Programmazione Avanzata  
A.A 22-23

## Contents

<b>1</b>	<b>Introduzione</b>	<b>2</b>
<b>2</b>	<b>L'applicazione</b>	<b>2</b>
2.1	Le interfacce . . . . .	3
2.1.1	L'interfaccia di login . . . . .	3
2.1.2	L'interfaccia di registrazione . . . . .	4
2.1.3	La lista degli anime . . . . .	5
2.1.4	L'interfaccia di inserimento anime . . . . .	6
2.2	Dettagli implementativi . . . . .	6
2.2.1	ScheletroController.java . . . . .	6
2.2.2	Notifica.java . . . . .	7
2.2.3	Interazione col servizio . . . . .	7
<b>3</b>	<b>Il servizio</b>	<b>7</b>
3.1	Interazione col database . . . . .	7
3.2	API . . . . .	8
<b>4</b>	<b>Info e credenziali di prova</b>	<b>8</b>

# 1 Introduzione

Questa applicazione nasce con l'obiettivo di fornire agli utenti la possibilità di avere una lista di anime privata, pensata per contenere quelli che l'utente vorrà vedere in futuro, così da appuntarsi e non dimenticarsene.

L'applicazione è creata appositamente per gli anime, così da essere la prima scelta per tutti coloro che li amano.

L'applicazione è stata pensata per raggiungere un numero più alto possibile di utenti, per questo è possibile usarla anche in lingua inglese.

Non è stata implementata la possibilità (per un admin) di aggiungere ulteriori anime al database, esso ne contiene alcuni che sono stati inseriti al momento del popolamento inviando una richiesta GET ad un'API<sup>1</sup>.

Per utilizzare le funzionalità di essa, l'utente dovrà accreditarsi con le sue credenziali.

Parleremo adesso di come l'applicazione è strutturata, discutendo, oltre a tutte le sue funzionalità, anche di alcuni dettagli tecnici ed implementativi.

## 2 L'applicazione

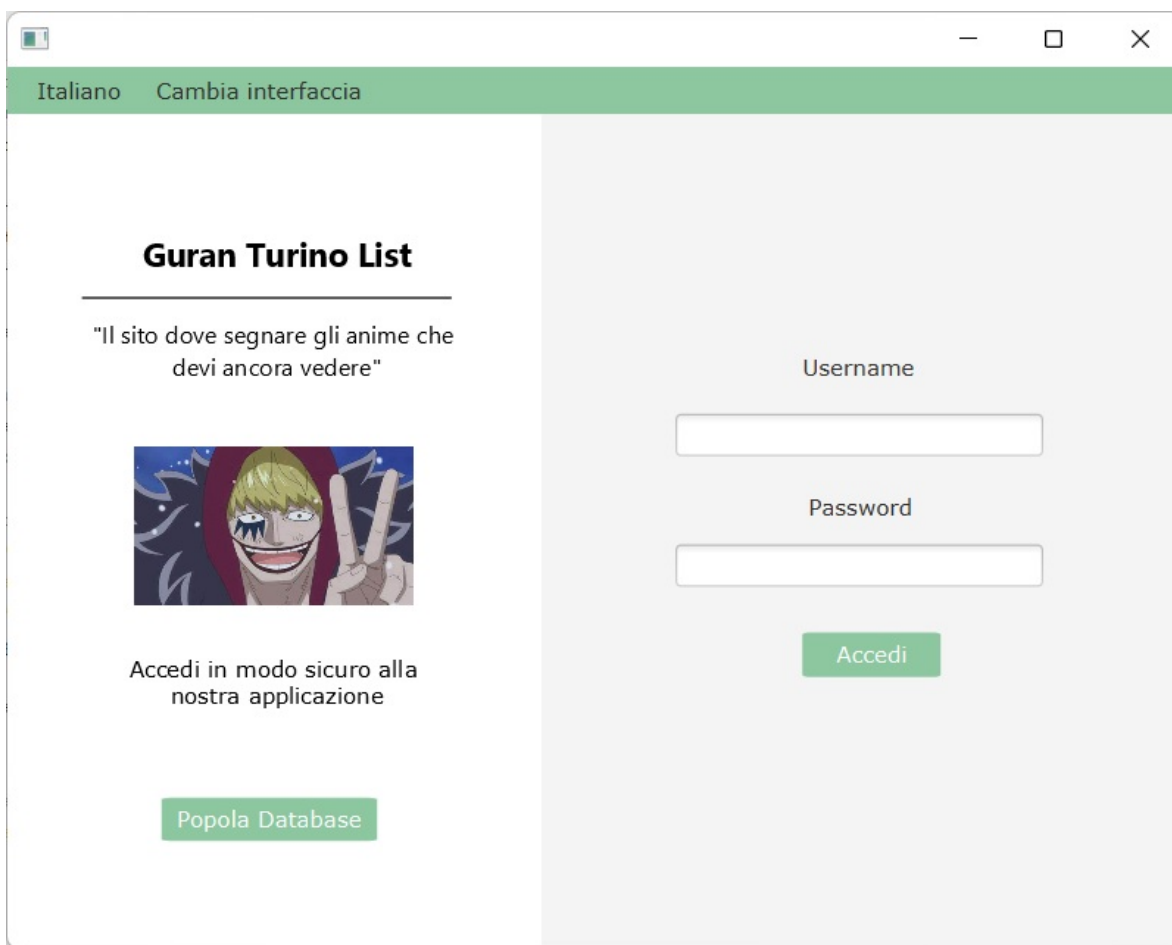
In questa sezione parleremo inizialmente delle varie interfacce che costituiscono l'applicativo e delle funzionalità di quest'ultime, infine discuteremo anche alcune delle principali scelte progettuali prese.

---

<sup>1</sup><https://api.jikan.moe/v4/anime>

## 2.1 Le interfacce

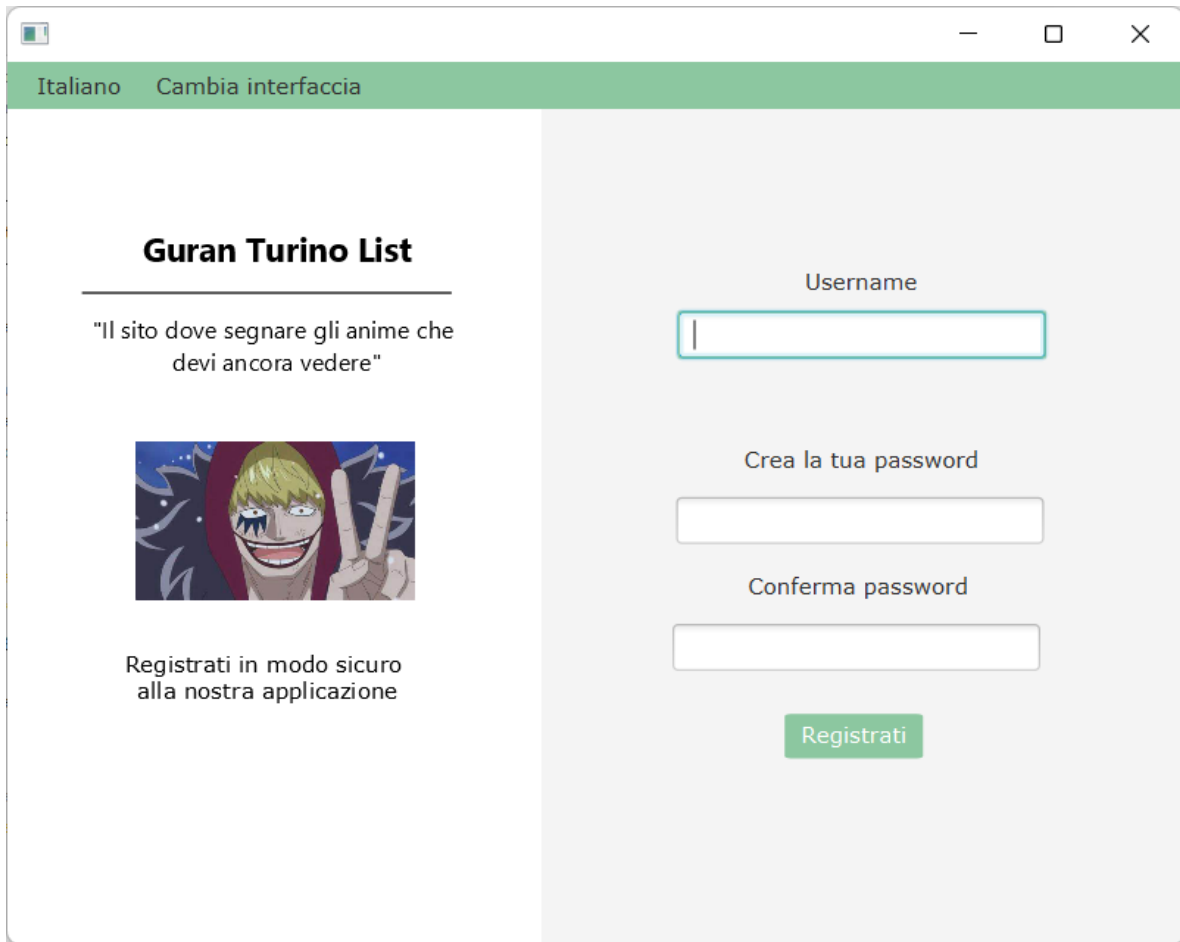
### 2.1.1 L'interfaccia di login



The screenshot shows a web browser window with a green header bar containing the text "Italiano" and "Cambia interfaccia". The main content area is split into two columns. The left column has a white background and contains the title "Guran Turino List" in bold, followed by the tagline "Il sito dove segnare gli anime che devi ancora vedere". Below this is a character image of Gurren from Gundam, and the text "Accedi in modo sicuro alla nostra applicazione". At the bottom of this column is a green button labeled "Popola Database". The right column has a light gray background and contains a login form with labels "Username" and "Password" above their respective input fields, and a green "Accedi" button at the bottom.

Interfaccia iniziale dell'applicazione, qui l'utente può accreditarsi con username e password. Nel caso non fosse ancora registrato, può farlo passando all'interfaccia per la registrazione cliccando su *Cambia interfaccia* e poi su *Registrazione*.

### 2.1.2 L'interfaccia di registrazione




The screenshot shows a web browser window with a registration form. The browser's address bar shows 'Italiano' and 'Cambia interfaccia'. The page has a green header bar. The main content area is split into two columns. The left column has a white background and contains the title 'Guran Torino List', a quote 'Il sito dove segnare gli anime che devi ancora vedere', an anime character image, and a registration prompt. The right column has a light gray background and contains the registration form fields: 'Username', 'Crea la tua password', 'Conferma password', and a 'Registrati' button.

Italiano Cambia interfaccia

## Guran Torino List

"Il sito dove segnare gli anime che devi ancora vedere"



Registrati in modo sicuro alla nostra applicazione

Username

Crea la tua password

Conferma password

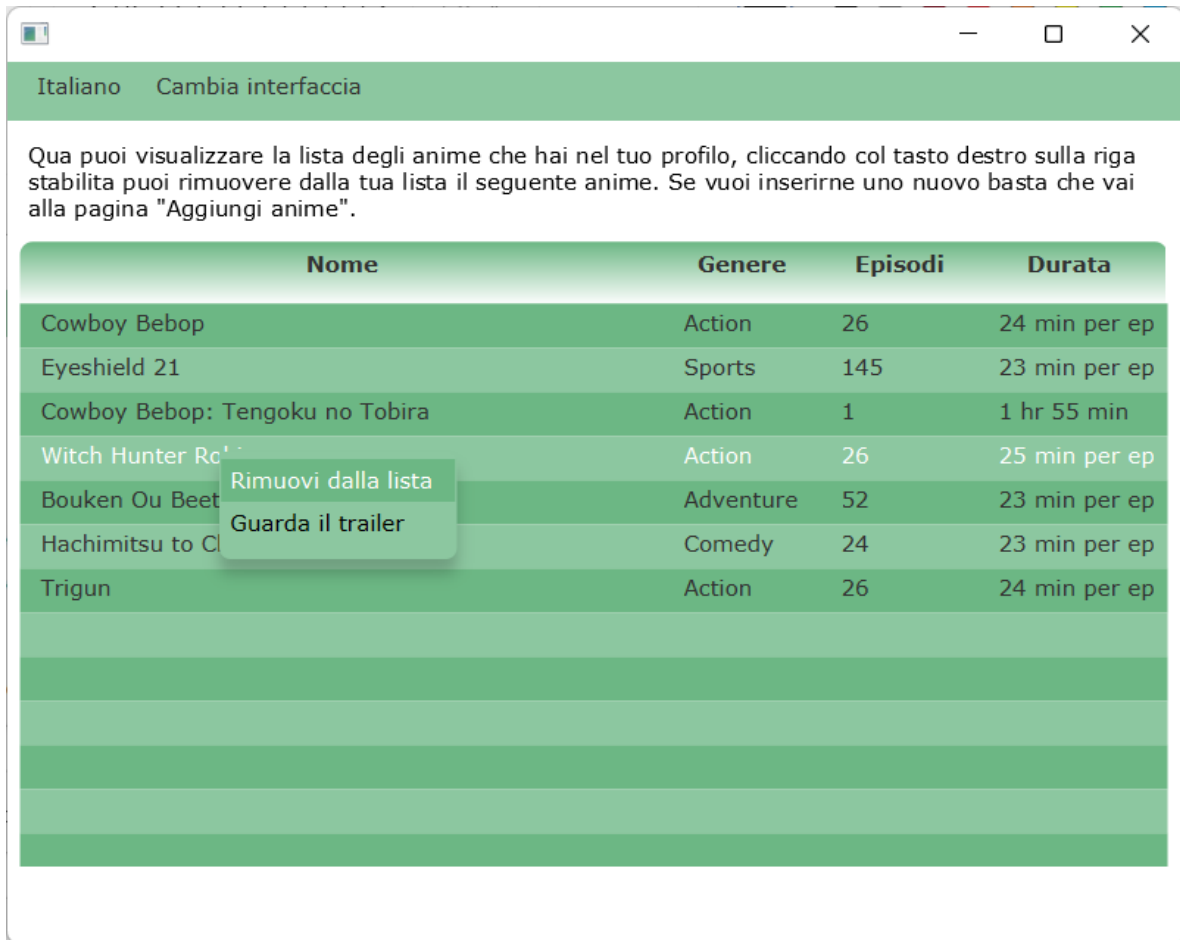
Registrati

In questa interfaccia l'utente può registrarsi.

L'username deve essere unico all'interno del database, quindi l'applicazione notificherà all'utente quando egli inserirà un username già in uso.

La password per la sicurezza dell'utente verrà hashata secondo l'algoritmo "bcrypt" prima di inserirla nel database.

### 2.1.3 La lista degli anime



Interfaccia principe dell'applicazione.

Può essere raggiunta andando su *Cambia interfaccia* e poi su *Lista anime*. Qua l'utente visualizza gli anime che ha personalmente inserito all'interno.

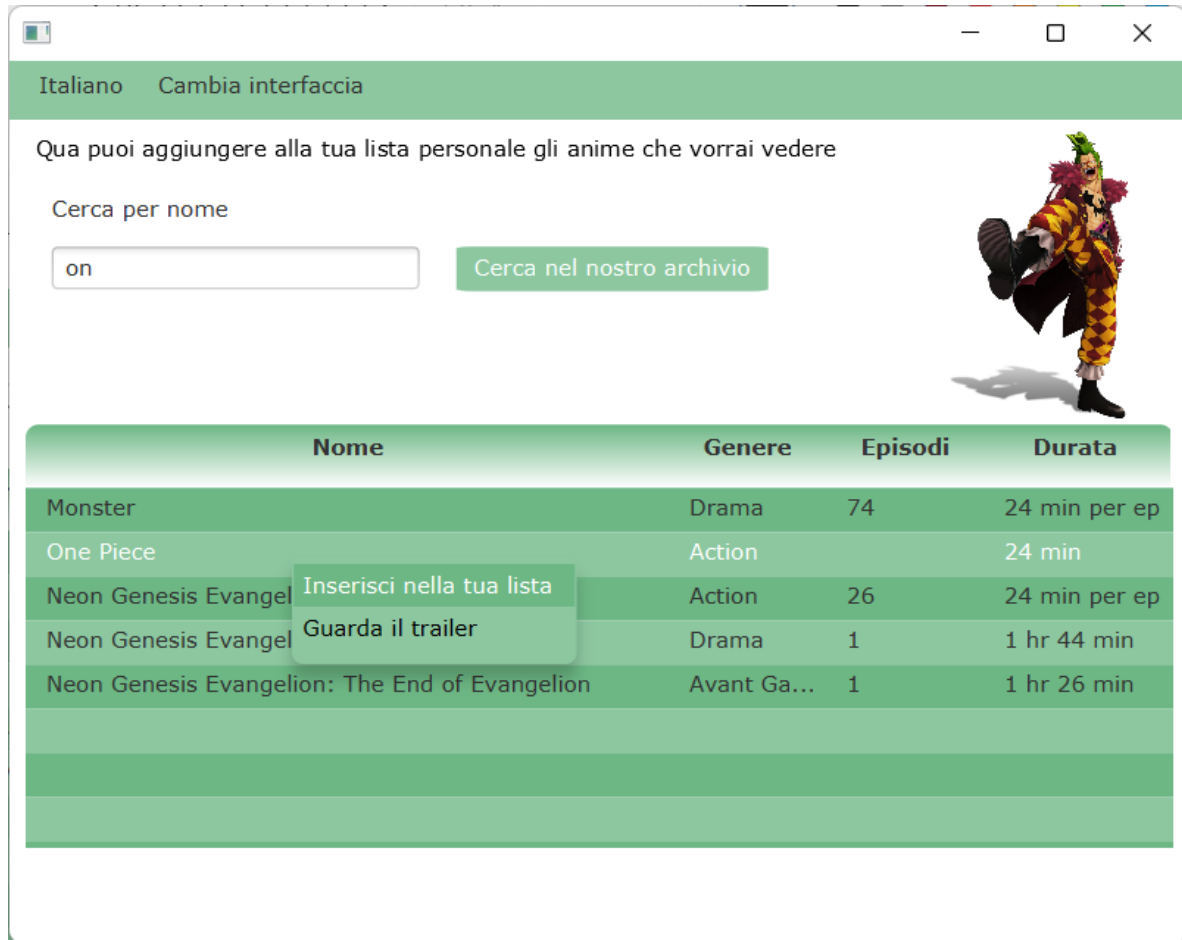
Cliccando *tasto destro* sopra un anime della lista è possibile per l'utente effettuare due operazioni:

- **Rimuovi dalla lista:** rimuove dalla lista dell'utente l'anime selezionato.
- **Guarda il trailer:** l'applicazione apre il browser dell'utente mostrando il trailer dell'anime selezionato (se possibile)<sup>2</sup>.

---

<sup>2</sup>Il trailer potrebbe non essere disponibile sul nostro database, in quel caso l'applicazione lo notificherà all'utente.

### 2.1.4 L'interfaccia di inserimento anime



Può essere raggiunta andando su *Cambia interfaccia* e poi su *Aggiungi anime*.

L'utente può cercare l'anime che desidera aggiungendolo alla propria lista scrivendo una sottostringa del suo nome nella barra di ricerca.

La ricerca, se è andata a buon fine, mostrerà una tabella contenente gli anime che la rispettano.

Cliccando *tasto destro* sopra uno degli anime è possibile per l'utente effettuare due operazioni:

- **Inserisci nella tua lista:** inserisce nella lista dell'utente l'anime selezionato.
- **Guarda il trailer:** l'applicazione apre il browser dell'utente mostrando il trailer dell'anime selezionato (se possibile)<sup>3</sup>.

## 2.2 Dettagli implementativi

Adesso analizziamo i principali dettagli implementativi che compongono il codice dell'applicazione. Per tutto il resto del codice consigliamo la lettura del Javadoc<sup>4</sup>

### 2.2.1 ScheletroController.java

Questa classe è stata creata per essere estesa da ogni controller dell'applicazione, poichè implementa alcune sezioni e funzionalità comuni ad ogni interfaccia.

Un esempio è proprio la barra in alto, che contiene il cambio della lingua e dell'interfaccia.

Questa scelta è stata fatta per realizzare l'applicazione in maniera che sia il più possibile modulare.

<sup>3</sup>Il trailer potrebbe non essere disponibile sul nostro database, in quel caso l'applicazione lo notificherà all'utente.

<sup>4</sup>prima devono essere generati da netbeans

### 2.2.2 Notifica.java

La seguente classe modella le notifiche che l'utente visualizza la maggior parte delle volte, dopo una certa operazione, e che comunica lui l'esito di quest'ultima.

Le notifiche sono identificate da un id numerico, il quale determina il tipo di notifica e il suo testo (nel caso venissero mostrate) oppure semplicemente l'evento che seguirà:

- **Notifiche di successo:** da 2000 a 2499
- **Notifiche di warning:** da 2501 a 2999
- **Notifiche di errore lato client:** da 4000 a 4999
- **Notifiche di errore lato server:** da 5000 a 5999

### 2.2.3 Interazione col servizio

L'applicazione interagisce col servizio inviando richieste HTTP. Per l'invio di queste richieste è stata utilizzata la classe `URLConnection`<sup>5</sup>

Alcune richieste possono richiedere l'invio di dati nel body della stessa, in questo caso i dati vengono serializzati in json.

## 3 Il servizio

Esso espone alcune API (che elencheremo successivamente) che sono interrogate al momento in cui l'applicazione fa una richiesta al servizio.

Oltre alle API, è di una certa importanza parlare anche di come abbiamo gestito in Java l'interazione col database.

Per maggiori informazioni riguardanti il codice consigliamo la lettura del Javadoc<sup>6</sup>.

### 3.1 Interazione col database

Il database è formato da tre tabelle:

- **Utente:** contiene i dati degli utenti che hanno effettuato la registrazione.
- **Anime:** contiene i dati di tutti gli anime di cui il servizio dispone, i quali possono essere aggiunti all'interno della lista di ogni utente. I campi trailer<sup>7</sup> e episodi<sup>8</sup> possono essere null.
- **ListaAnime:** considerando che quella tra utente e anime è una relazione *Molti a Molti*, questa è la loro tabella di join.

La modellazione e l'interazione con il database è stata sviluppata usando Spring JPA.

Prima di tutto abbiamo modellato i dati contenuti nel database creando due JavaBean:

- **Utente.java** : rappresenta i dati contenuti nella tabella Utente
- **Anime.java** : rappresenta i dati contenuti nella tabella Anime

Per le query più semplici abbiamo usato alcuni metodi già implementati in Spring JPA come *findByUsername*, mentre per quelle più complesse, per semplicità, abbiamo impostato l'attributo *nativeQuery* a true.

---

<sup>5</sup>`java.net.HttpURLConnection`

<sup>6</sup>prima devono essere generati da netbeans

<sup>7</sup>potrebbe non avere un trailer

<sup>8</sup>potrebbe non essere ancora concluso

## 3.2 API

Le API ritornano all'applicazione sempre un Json, questa spesso rappresenta la notifica che l'applicazione deve mostrare.

Per questo è stata creata la classe *CodiceRisposta.java*, che ha un solo attributo, l'identificativo della notifica. L'oggetto di questa classe viene serializzato ed inviato all'applicazione sotto forma di Json.

Di seguito la lista delle API fornite dal servizio.

Per maggiori informazioni leggere il relativo Javadoc<sup>9</sup>.

**GET** /anime

**GET** /anime/cerca

**DELETE** /anime

**PUT** /anime

**GET** /utente/login

**PUT** /utente/registrazione

## 4 Info e credenziali di prova

Per prima cosa eseguire il client, creare il database cliccando sul bottone *Popola database*, poi eseguire il server.

Per testare l'applicazione è possibile effettuare il login con le seguenti credenziali, che appartengono ad un utente che ha già qualche anime nella propria lista.

- **username** : bobo
- **password** : Onepiece1\*

Per quanto riguarda la UnitTest ne è stata inserita una nel progetto del servizio: *UtenteControllerTest.java*.

---

<sup>9</sup>prima devono essere generati da netbeans