

GymService: Web Service per la Gestione di Esercizi Fitness

Giulio Scarpellini, Raffaele Restifo, Cardellini Lorenzo

Maggio 2025

1 Introduzione e Obiettivi

L'obiettivo principale del progetto è creare un **Web Service RESTful** in Java per:

- Gestire un database composto da due tabelle "Trainer" ed "Esercizi", suddivisi per gruppi muscolari.
- Consentire operazioni CRUD (Create, Read, Update, Delete), ed anche login e registrazione sugli esercizi.

2 Struttura del Progetto

Il sistema si articola in due componenti principali:

2.1 Back-End

- **Java con JAX-RS (Jersey):** Framework per la realizzazione del web service REST.

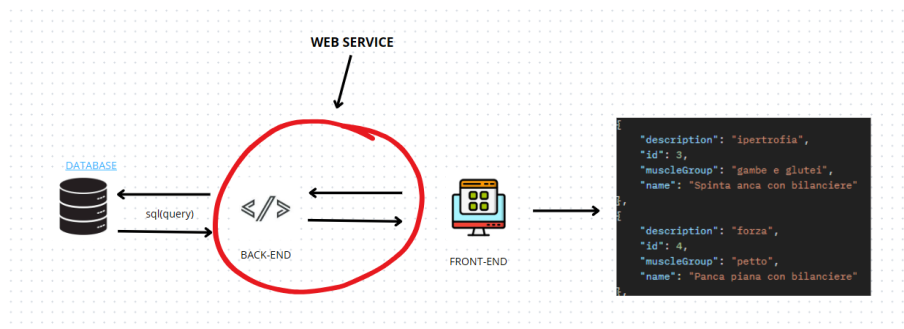


Figura 1: Struttura

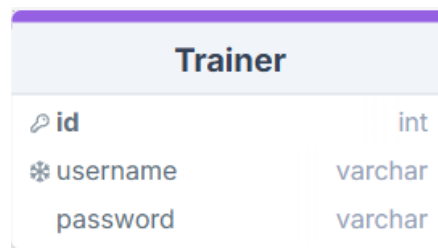
2.2 Database MySQL

- **Tabella esercizi** (vedi Figura 2):
 - * Nome esercizio
 - * Gruppo muscolare (es. “Pettorali”, “Quadricipiti”)
 - * Descrizione testuale
- **Tabella trainer** (vedi Figura 3):
 - * ID
 - * Username
 - * Password



Esercizi	
 id	int
nome	varchar
descrizione	varchar
gruppo_muscolare	varchar

Figura 2:





Trainer	
 id	int
 username	varchar
password	varchar

Figura 3:

2.3 Front-End (Opzionale)

- Non implementato

3 API Endpoints

Principali endpoint REST implementati:

GET	/api/exercises	# Restituisce la lista di tutti gli esercizi
GET	/api/exercises/{id}	# Restituisce i dettagli di un esercizio tramite ID
POST	/api/post	# Crea un nuovo esercizio
PUT	/api/put/{id}	# Aggiorna un esercizio esistente
DELETE	/api/delete/{id}	# Elimina un esercizio tramite ID
SIGN IN	/api/signin	# Crea un nuovo Trainer
LOGIN	/api/login	# Accedi al Trainer

4 Tecnologie Utilizzate

Componente	Tecnologia
Back-End	Java con JAX-RS (Jersey)
Front-End	Non implementato
Database	MySQL e XAMP
Testing API	Postman
Documentazioni	Latex

5 Come avviare il progetto

5.1 Requisiti

- Postman (o qualsiasi altro strumento per testare API REST)
- XAMPP per la gestione del database locale
- Java + Maven
- Eclipse (o un IDE compatibile)

6 Setup del Progetto

6.1 Collegamento al Database

1. Avvia XAMPP.
2. Crea un database chiamato `trainer`.
3. Importa il file `trainer.sql` presente nella repository all'interno del database.

6.2 Avvio di Apache e MySQL

- Dal pannello di controllo di XAMPP, avvia i servizi **Apache** e **MySQL**.

6.3 Esecuzione del Progetto in Eclipse

1. Apri il progetto in Eclipse.
2. Vai su Run As → Maven Build....
3. Inserisci nel campo Goals: `jetty:run`.
4. Clicca su Run.

6.4 Test dell'API

- Una volta avviato Jetty, puoi testare gli endpoint tramite Postman o strumenti simili. Esempi seguenti con Postman.

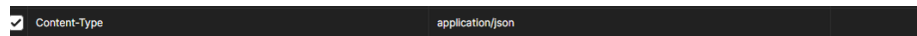


Figura 4: Valido per GET, SIGNIN, LOGIN, PUT, DELETE

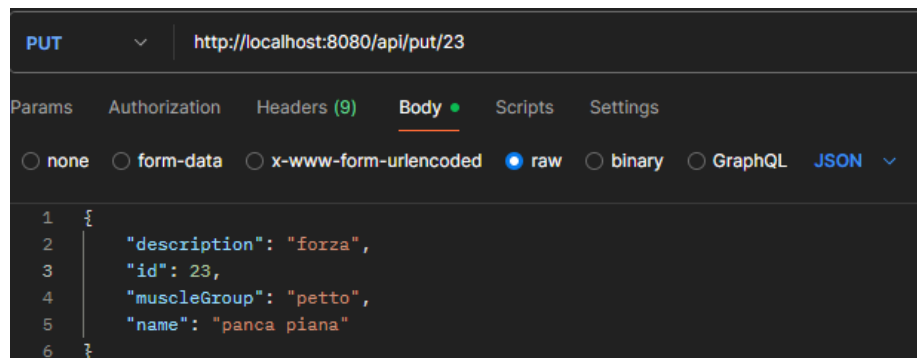


Figura 5: PUT

6.5 Autenticazione

L'endpoint POST /api/post è protetto da autenticazione tramite **token**.

1. Ottieni un token valido tramite la richiesta POST /api/login.

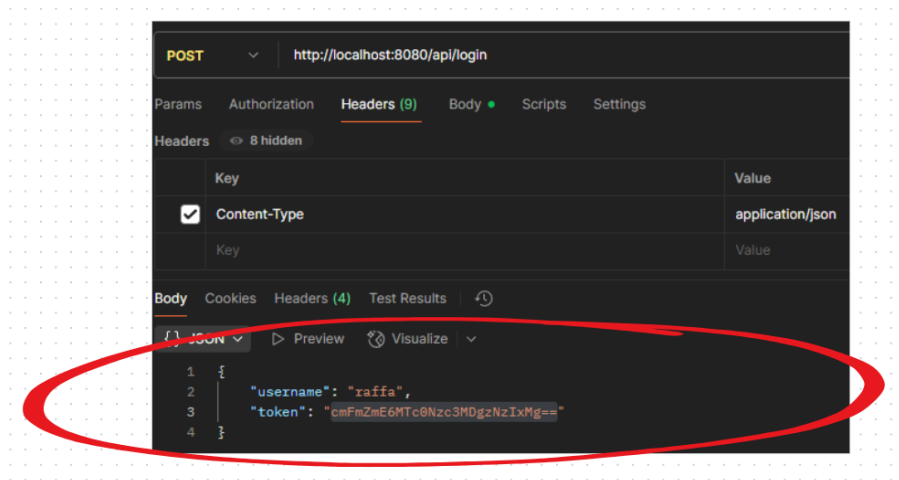


Figura 6: LOGIN

2. Inserisci il token nell'header `Authorization` della richiesta:

`Authorization: Bearer <token>`

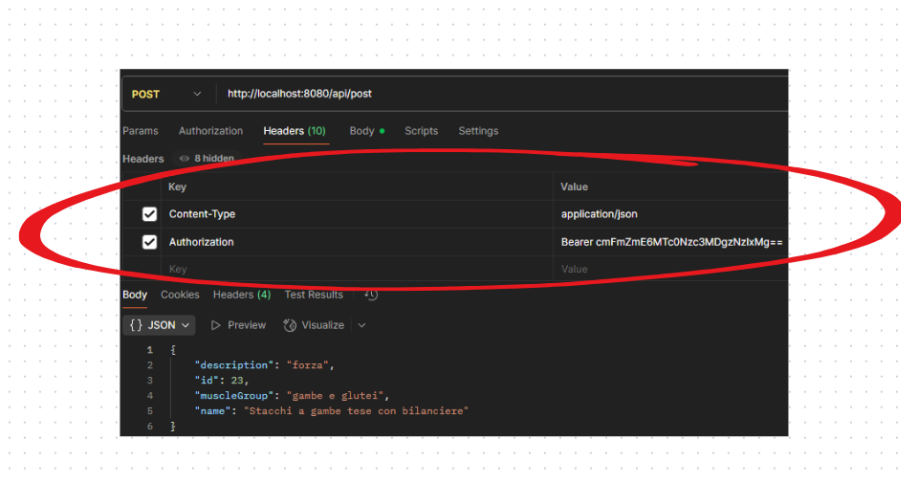


Figura 7: POST

6.6 Endpoints disponibili

Per l'elenco completo, fai riferimento alla sezione **API Endpoints**.

7 Note

- Assicurati che XAMPP sia avviato prima di avviare il server.

8 Scelte architetturali

8.1 Database

Abbiamo scelto di utilizzare un database per memorizzare le informazioni relative agli esercizi e ai trainer (vedi sezione *Schema del Database*). Le operazioni su questo livello avvengono tramite query SQL, eseguite dal back-end utilizzando **JDBC (Java Database Connectivity)**.

8.2 Back-End

È la parte principale del nostro progetto, sviluppata in Java utilizzando **JAX-RS (Jersey)** per la realizzazione di un web service RESTful. Il back-end funge da intermediario tra l'interfaccia utente (front-end) e il database. Per maggiori informazioni si rimanda alla sezione *Struttura del Progetto* → *Back-End*.

8.3 Front-End e Testing

Abbiamo scelto di non sviluppare un'interfaccia grafica (HTML/JS), in quanto non richiesta ai fini della valutazione. Per il testing delle API REST abbiamo utilizzato **Postman**, uno strumento che ci consente di:

- Testare ed inviare richieste HTTP (GET, POST, DELETE, ecc.);
- Visualizzare le risposte del server (body, header, codice di stato, ecc.),
fai riferimento a Setup del Progetto

8.4 Comunicazione tra livelli

La comunicazione tra i vari livelli del sistema è così strutturata:

- Il **Front-End** (in questo caso Postman) invia richieste HTTP (CRUD, login e registrazione) al web service.
- Il **Web Service** (Back-End) elabora la richiesta, interagisce con il database e restituisce i dati richiesti.
- I dati vengono ritornati in formato **JSON**

9 Conclusioni e ipotesi di implementazione

Il progetto **GymService** e' funzionante, abbiamo rispettato l'obiettivo che ci eravamo posti. Le possibili implementazioni future potrebbero essere, realizzare i token con JWT, creare una classe utente con annesso ad esso la propria area privata e la sviluppare un Front-end.