Documento descrittivo del progetto

1. Introduzione

Il progetto include un esempio di architettura software containerizzata. Il progetto combina un **backend HTTP REST** ed un **database relazionale** per la memorizzazione dei dati.

---

2. Componenti principali

2.1 Backend (FastAPI)

- Implementa API REST per:
  - Creazione di sessioni di esercizio (`POST /sessions`).
  - Recupero di sessioni filtrate (`GET /sessions`).
  - Calcolo di statistiche utente (`GET /users/{user_id}/stats`).

2.2 Database (PostgreSQL / TimescaleDB)

- Database opensource di tipo SQL

2.3 pgAdmin

- Interfaccia web per gestire il database.

2.4 Nginx

- Funziona come reverse proxy per il backend e come static server per frontend.
- Consente gestione del traffico, HTTPS, cache, load balancing ecc…

2.5 Tester Python

- Script per generare sessioni simulate.
- Invio automatico delle sessioni al backend.
- Recupero statistiche e salvataggio in CSV.

---

3. Architettura

Il progetto utilizza una **architettura a microservizi** containerizzata:

```
[Frontend React/HTML] -> Nginx -> [Backend FastAPI] -> PostgreSQL/TimescaleDB
                                          |
                                          -> pgAdmin (per gestione DB)
```

- Tutti i servizi comunicano tramite **rete Docker interna**.
- Il backend espone un path `/api/` accessibile tramite Nginx.

- I dati sono persistenti e memorizzati nel database Postgress

---

4. Flusso dati

1. L'utente (o script di test) invia una sessione tramite POST.
2. Il backend valida e salva la sessione nel DB.
3. Gli utenti possono recuperara i dati API tramite GET.
4. I dati possono essere esportati in CSV tramite script di test.

---

5. Discussioni ed assunzioni

Il sistema proposto si basa su un insieme di microservizi.

Ho scelto questa architettura perchè di facile gestione ed estensione, l'utilizzo di Docker permette l'utilizzo del sistemi multipiattaforma.

Il sistema viene deployato tramite un file di configurazione docker-compose facilmente integrabile anche in sistemi di deploy automatizzati

Il backend è realizzato con FASTAPI framework asincrono python leggero e moderno che consente un'integrazione e realizzazione rapida di microservizi web.

Il database utilizzato è timescaleDB, versione di basata su postgres ottimizzata per serie temporali

NGNIX funziona da reverse proxy ed è facilmente modificabile per consentire l'utilizzo di certificati (HTTPS)

I limiti principali di questa soluzione potrebbero essere i seguenti:

- Prestazioni non elevate comparate ad altri framework in caso di traffico elevato (dovuto a FAST API)
- Possibile instabilità su sistemi windows (dovuta a Docker)
