1. Funzionalità base del programma

Il sistema permette a due tipologie di utenti (trainer e clienti) di interagire con una piattaforma per la gestione di schede di allenamento.

1.1 Area Trainer

- **Gestione clienti**: i trainer possono creare, visualizzare, modificare ed eliminare account cliente. Ogni cliente è associato all'ID del trainer che lo ha creato.
- **Gestione esercizi**: CRUD completo (Create, Read, Update, Delete) di esercizi con nome, descrizione, obiettivo, difficoltà, media (URL YouTube o immagine), flag per ripetizioni/serie/tempo/recupero/macchinario/peso, gruppi muscolari primari e secondari.
- **Builder di workout**: interfaccia con tasto per aggiungere esercizi per comporre una scheda personalizzata scegliendo esercizi esistenti, impostando ripetizioni, serie, peso, durata e recupero. Salvataggio in `workout_plan_exercises` con campi `exec_time_s`, `rest_s`, `suggested_kg` e ordine (`ord_idx`).
- **Storico schede**: archiviazione logica di schede datate tramite flag `archived`, con possibilità di visualizzare lo storico.
- **Statistiche**: contatori di numero di clienti, schede create, esercizi totali visibili in dashboard.

1.2 Area Cliente

- **Dashboard**: elenco schede attive, accesso rapido.
- **Dettaglio scheda**: vista read-only di ogni scheda con esercizi dettagliati.
- **Esecuzione workout**: schermata server-side per ogni esercizio con media, descrizione, timer esecuzione e recupero, avanzamento sequenziale con pulsante "Avanti", reset e gestione fasi esecuzione/recupero.
- **Storico**: accesso alle schede archiviate.

2. Scelte implementative adottate

2.1 Struttura del database

- **Utenti**: tabella `users` con `role` ('trainer' o 'client'), relazione self-referencing (`trainer_id` per i clienti).
- **Esercizi**: tabella `exercise` con FK verso `exercise_objectives`, `exercise_difficulty`, campi booleani per flag e `media_url`. Relazione molti-a-molti con `muscle_groups` tramite `exercise_muscle_groups` (PK composita e `is_primary`).
- **Workout plans**: tabella `workout_plans` con `client_id`, `trainer_id`, `created_at`, `archived`.
- **Esercizio in scheda**: tabella `workout_plan_exercises` (precedentemente `workout_exercises`) con campi `exec_time_s`, `rest_s`, `suggested_kg`, FK `plan_id` e `exercise_id`, FK `machine` e `ord_idx` per ordinamento.

  Motivazione: separare metadati (scheda) e righe (esercizi) rende semplice storicizzare e modificare le schede, aggiungere o rimuovere esercizi senza duplicare dati.

2.2 Strategie di accesso e serializzazione

- Uso di metodi del manager per incapsulare query SQL raw, garantendo foreign key e transazioni semplici.
- Serializzazione in Python (trasformazione di `None` e `Undefined`) prima di passare a Jinja, per evitare problemi di JSON invalidi.
- Sempre `open_connection()` prima delle operazioni e `close_connection()` in uscita, minimizzando il tempo di connessione aperta.

2.3 Caricamento delle risorse

- **Server-side rendering**: pagine HTML dinamiche generate da Flask senza SPA, per semplicità e SEO.
- **JavaScript**: solo per arricchire builder di workout e timer, senza chiamate API aggiuntive.

3. Framework grafico e tecniche responsive

- **Bootstrap 5**: layout grid, tabelle responsive, form controlli, componenti button/card.
- **Custom CSS**: nelle cartelle `static/css`, file specifici per trainer, client e about.
- **Media queries**: applicate in `base.html` e nei file CSS custom per adattare padding, font-size e visibilità di elementi.
- **Flexbox / Utility classes**: `.d-flex`, `.justify-content-between`, `.table-responsive` per adattare tabelle a dimensioni variabili.

Perché: Bootstrap garantisce un supporto mobile-first out-of-the-box e una curva di apprendimento veloce.

4. Tecnologie e istruzioni non viste a lezione

- **Progressive Web App (PWA)**: `manifest.json`, `service-worker.js` per caching offline e installazione.
- **Flask-JWT-Extended**: generazione e gestione di token JWT, protezione API e pagine.
- **Bcrypt**: hashing sicuro delle password.
- **SQLite PRAGMA foreign_keys=ON**: per attivare vincoli FK in SQLite.

5. Migliorie possibili

- **Drag-and-drop**:usare il drag and drop per inserire gli esercizi dentro le schede
- **Uploader media**: anziché solo URL, permettere upload di immagini/video.
- **Grafici di progresso**: usare grafici per statistiche visive sui log degli esercizi.
- **FrontEnd piú curato**

6. Progettazione e feedback

6.1 Metodologia adottata

- **Project Charter**: definizione di obiettivi, stakeholder, rischi, budget.
- **WBS / Gantt / PERT**: scomposizione in task e scheduling.

6.2 Punti di forza

- **Modularità**: separazione tra dati, logica e presentazione.
- **Database consistente**: uso di FK e normalizzazione per evitare ridondanze.
- **PWA readiness**: compatibilità offline e mobile-first.

6.3 Criticità incontrate

- **Registrazione di Utente da parte del PT** problemi con l'autenticazione ed errori 401
- **Chiavi Esterne** problemi con le tabella per esempio quando si eliminava un esercizio
- **Modifiche di schede ed esercizi** difficile fare risultare le pagine piú comprensibili
- Esecuzione della scheda da parte del cliente