Corso di Laurea in Ingegneria Gestionale

Classe L-8

Politecnico di Torino

Realizzazione di un applicativo per la generazione di piani d'allenamento personalizzati

Relatore

Prof. Giuseppe Bruno Averta

Candidato

Luca Pellegrino

S296680

Sessione di Laurea Luglio 2025

INDICE

1 Proposta di progetto

Titolo della proposta

Realizzazione di un applicativo per la generazione di piani d'allenamento personalizzati

Descrizione del problema proposto

Molte persone nella loro quotidianità praticano attività fisica per diversi motivi, ma non sempre seguono un programma specifico che, attraverso una migliore scelta degli esercizi, consentirebbe loro di ottenere maggiori risultati. Lo scopo di questo software è organizzare piani d'allenamento personalizzati in base alle esigenze del singolo utente, con l'obiettivo di soddisfare i requisiti espressi (e.g. la perdita di peso o lo sviluppo della potenza muscolare) tenendo conto di vari fattori, quali ad esempio i giorni di allenamento alla settimana, il tempo disponibile per ogni allenamento e il livello di difficoltà. Il programma proporrà una pianificazione settimanale ottimizzando le richieste dell'utente.

Descrizione della rilevanza gestionale del problema

L'ottimizzazione del piano di allenamento migliora l'efficienza nella gestione del tempo e delle risorse personali. Un approccio automatizzato consente, infatti, di ridurre il tempo necessario per pianificare gli allenamenti, migliorando la coerenza e la qualità del programma di allenamento. L'applicazione potrebbe essere utilizzata anche da personal trainer o palestre, permettendo loro di offrire un servizio su misura ai clienti senza dover

elaborare manualmente i singoli programmi. Infine, questo software, usato standalone o integrato eventualmente in una applicazione di fitness, potrebbe rappresentare anche un plus e quindi un vantaggio competitivo per le palestre che lo implementano.

Descrizione dei data-set per la valutazione

Il dataset utilizzato dal software sarà creato prendendo come spunto il dataset "Best 50 exercises for your body" presente sulla piattaforma Kaggle.
La tabella elenca i vari tipi di esercizi, con annesse le determinate caratteristiche:

- Name of exercise (STRING)
- Sets (INTEGER)
- Reps (INTEGER)
- Benefit (STRING)
- Burns calories (per 30 min) (INTEGER)
- Target muscle group (STRING)
- Equipment needed (STRING)
- Difficulty level (STRING)

Considerando il fatto che il dataset contiene un numero di record limitato, sarà ampliato creando dei nuovi record, replicando quelli esistenti ma modificando alcuni parametri come il numero di ripetizioni (Reps) e di serie (Sets). Sarà inoltre aggiunta una colonna con un identificativo univoco (INTEGER) per ogni riga al fine di distinguere esercizi con lo stesso nome eseguiti con numero diverso di serie e ripetizioni. Alcuni dati presenti nel database saranno elaborati all'interno del programma per ottenere, mediante opportuni calcoli e proporzioni, la durata effettiva dell'esercizio (calcolata dal numero di serie) e le calorie effettive (calcolate in proporzione alla durata dell'esercizio).

Descrizione preliminare degli algoritmi coinvolti

Il software sarà realizzato in linguaggio Python, utilizzando i pattern MVC e DAO. Nella prima parte saranno utilizzate alcune funzioni di ricerca nel database tramite query SQL per mostrare a schermo le tipologie di esercizio che rispettano i vincoli forniti dall'utente in input. Nella seconda parte si chiederanno all'utente le proprie preferenze ed il proprio obiettivo e, mediante un algoritmo ricorsivo che lavora sulla base di questi filtri, sarà generato un piano di allenamento personalizzato. L'utente potrà inserire il numero di allenamenti settimanali che ha intenzione di fare, il tempo a disposizione per ogni allenamento ed il livello (beginner, intermediate, advanced). L'applicativo, in base alle richieste calcolerà un piano d'allenamento andando a massimizzare l'obiettivo impostato. Se l'obiettivo è la perdita di peso sarà massimizzato il consumo calorico. Se l'obiettivo è invece potenziare la massa muscolare si minimizzerà il numero di ripetizioni degli esercizi nell'ipotesi che ripetizioni brevi siano più efficaci per il raggiungimento dello scopo. L'algoritmo, inoltre, bilancerà gli esercizi tra i vari giorni, in modo da non avere nel piano lo stesso esercizio troppe volte ed evitando che lo stesso esercizio venga ripetuto per due giorni consecutivi, consentendo al muscolo il necessario riposo. Sarà anche possibile per l'utente inserire in una lista di tutti gli esercizi che non si vogliono svolgere e l'algoritmo farà in modo che essi non compaiano poi nel piano d'allenamento.

Descrizione preliminare delle funzionalità previste per l'applicazione software

L'applicazione dispone di più funzionalità:

• Permette di osservare tutti gli esercizi presenti nel database, o osservarne solo alcuni, in base ai filtri scelti dall'utente.

• Permette all'utente di creare una lista in cui mettere gli esercizi non graditi, per fare in modo che non vengano selezionati nel piano (è ovviamente possibile modificare e svuotare questa lista quando necessario);

• Permette all’utente di impostare i propri vincoli (tempo a disposizione, numero di allenamenti settimanali, livello) ed il proprio obiettivo principale. In base a questi dati inseriti in input sarà calcolato il miglior piano d’allenamento possibile.

Al termine l’applicazione stamperà a schermo il programma settimanale pronto per essere utilizzato.

2 Descrizione dettagliata del problema affrontato

Contesto operativo e potenzialità

Numerosi studi confermano che il benessere fisico e l'esercizio costante migliorano non solo la prestazione sportiva, ma anche la quotidianità e, di riflesso, la produttività lavorativa.

In un contesto in cui i ritmi di vita sono sempre più serrati, dedicare tempo a sé stessi diventa un investimento essenziale. È qui che l'applicativo si colloca: non è pensato per chi punta alle gare di bodybuilding, bensì per chi desidera mantenersi in forma e ritagliarsi un momento di pausa rigenerante.

Quando il tempo è limitato, però, l'allenamento deve essere ottimizzato: routine improvvisate non producono risultati e aumentano il rischio di infortuni. Un software capace di generare piani personalizzati, calibrati su livello, disponibilità di tempo e finalità individuali, svolge quindi un duplice ruolo.

Da un lato semplifica la vita dell'utente, che riceve indicazioni puntuali e aggiornate; dall'altro alleggerisce il lavoro di personal trainer e palestre, offrendo un servizio "premium" replicabile a costi ridotti e liberando risorse per un coaching di qualità.

Input e output

Durante l'esecuzione il software elabora due categorie di input, tra loro complementari:

1. Dati statici – il dataset degli esercizi.
   Ogni record comprende nome, numero di serie e ripetizioni, calorie stimate per 30 minuti, livello di difficoltà e altri attributi di base. Costituiscono il vocabolario di movimenti dal quale l'algoritmo attinge per costruire i piani.
2. Dati dinamici – informazioni fornite dall'utente tramite la GUI.
   Comprendono: livello di allenamento, numero di sessioni settimanali, tempo disponibile per ciascuna seduta, obiettivo primario (perdita di peso o aumento forza) ed eventuale blacklist di esercizi da escludere per infortuni o preferenze.

In output l'applicativo produce due livelli di risultato:

- Schermata 1 – Ricerca esercizi
  Visualizza l'elenco di esercizi che rispettano i filtri impostati, facilitando la consultazione del dataset.
- Schermata 2 – Piano settimanale
  Genera un programma ottimizzato giorno per giorno. Per ciascuna seduta fornisce:
  – la sequenza di esercizi con dettagli su serie, ripetizioni, durata stimata, calorie e gruppi muscolari;
  – un riepilogo dei carichi globali (totale calorie o punteggio forza) per verificare l'equilibrio complessivo.

In pochi secondi l'utente ottiene quindi una guida pratica e personalizzata, pronta per essere messa in pratica e monitorata nel tempo.

Criticità

La progettazione automatica di una settimana di allenamenti implica esplorare un enorme spazio di possibili combinazioni. Nell'applicazione pratica emergono due criticità principali, strettamente collegate fra loro:

1. Tempo di calcolo proporzionale alla durata della sessione.
   L'algoritmo deve selezionare per ogni giornata una sequenza di esercizi la cui durata complessiva non superi il limite impostato dall'utente. All'aumentare di questo limite cresce in modo esponenziale il numero di combinazioni da valutare. Di conseguenza, il tempo necessario alla ricerca tramite back-tracking può diventare significativamente più lungo, rendendo la generazione delle soluzioni meno immediata.
2. Elevata densità del grafo.
   Poiché non esistono criteri precisi per escludere a priori determinate combinazioni di esercizi, il programma genera un grafo praticamente completo (ogni nodo collegato a quasi tutti gli altri). Ciò comporta un grado di ramificazione molto elevato (branching-factor), che moltiplica rapidamente i percorsi da analizzare, rallentando il processo di ricerca—soprattutto quando il dataset è ampio.
   Per mitigare questa criticità, varianti dello stesso esercizio (stesso nome ma serie e ripetizioni diverse) non vengono collegate tra loro nel grafo. Infatti, sarebbe inutile proporre due varianti dello stesso movimento una dopo l'altra nello stesso piano. Altri meccanismi di potatura (descritti nei paragrafi successivi) riducono ulteriormente la complessità, ma non eliminano del tutto la necessità di mantenere numerosi archi tra i nodi.

3 Descrizione del data-set

Struttura del data-set

L'applicativo genera i piani di allenamento attingendo da un archivio di esercizi predefinito. Come punto di partenza è stato adottato il dataset "Best 50 exercises for your body", disponibile sulla piattaforma Kaggle.

Il dataset iniziale contiene per ciascun esercizio i seguenti attributi:

- Name of Exercise – nome univoco dell'esercizio.
- Sets – numero di serie consigliate.
- Reps – numero di ripetizioni per serie.
- Benefit – principali benefici fisici ottenibili.
- Burns Calories (per 30 min) – stima delle calorie bruciate in 30 minuti di attività.
- Target Muscle Group – gruppi muscolari coinvolti.
- Equipment Needed – attrezzatura necessaria per l'esecuzione.
- Difficulty Level – livello di difficoltà (beginner, intermediate, advanced).

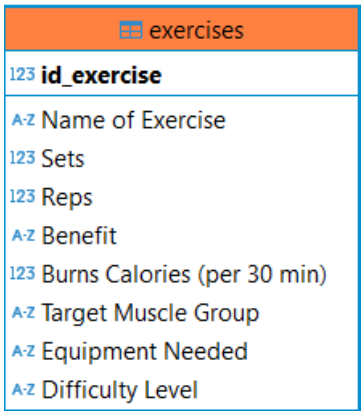

Figura 3.1 Diagramma ER del data-set

A partire dal dataset iniziale, sono stati aggiunti manualmente ulteriori attributi utili per realizzare gli algoritmi di ottimizzazione. Questi nuovi campi, calcolati sulla base dei dati già presenti, sono:

- Id – un identificatore numerico univoco assegnato in ordine crescente, necessario per distinguere chiaramente ogni esercizio, incluse le varianti dello stesso movimento (stesso nome, ma numero differente di serie e ripetizioni).
- tempoTot – il tempo totale (in secondi) necessario per eseguire l'esercizio.
  Per il calcolo si assume un tempo di 10 secondi per ogni ripetizione e 120 secondi di recupero fra le serie. Sebbene in teoria le pause siano pari a (sets - 1) × 120 secondi, è stata considerata anche la pausa finale dopo l'ultima serie, poiché è sempre presente prima di passare a un nuovo esercizio.
- caloriesTot – il numero di calorie consumate durante l'esercizio, rapportato al tempoTot. Nel dataset originale era presente solo il dato relativo a 30 minuti di attività, pertanto questa colonna rende il valore più realistico e personalizzato.
- punteggioForPath – un punteggio calcolato attraverso una formula specifica per favorire esercizi caratterizzati da un basso rapporto tra ripetizioni e serie.
  Questo criterio si basa sull'assunto che esercizi con meno ripetizioni e più serie siano più efficaci nello sviluppo della massa muscolare, permettendo l'utilizzo di carichi superiori. L'algoritmo cercherà poi di minimizzare tale punteggio nel costruire i piani orientati all'aumento della forza.

Di seguito è riportata la parte di codice Python presente nel file esercizio.py che evidenzia il calcolo di questi nuovi campi:

```python
def __post_init__(self):
    #in secondi (2 minuti per le pause tra una serie e l'altra, 10 s per ogni ripetizione)
    self.tempoTot = (120*self.sets) + (10*self.reps*self.sets)

    #formula in cui viene diviso il tempo per 1800(secondi presenti in 30 minuti) e poi moltiplicato per il dato forntoci dal database
    self.caloriesTot = (self.tempoTot/1800)*self.calories30min

    #punteggio per consentire creazione di piano che favorisca esercizi con più sets e meno reps per lo sviluppo di maggior massa muscolare
    self.punteggioForPath = self.reps/self.sets + 0.2 * self.reps
```

Figura 3.2 Creazione campi classe Esercizio

Infine, per ampliare il dataset e aumentarne la varietà, sono state generate nuove righe a partire dagli esercizi originali. Questo processo ha previsto la modifica del numero di serie e ripetizioni, garantendo comunque che gli altri valori associati, ad esempio le calorie, rimanessero realistici e affidabili.

Collegamento programma-dataset

Per accedere al dataset nel proprio ambiente di sviluppo è necessario seguire questi passaggi:

1. Importare il file SQL fit_exercises.sql presente nella cartella database. È possibile usare strumenti grafici come DBeaver o eseguire il file direttamente in MariaDB.
2. Stabilire una connessione al database, utilizzando le proprie credenziali (username e password) inserite nel file connector.cnf.

Una volta configurata correttamente la connessione, il dataset viene interrogato attraverso il livello DAO, che contiene diverse funzioni responsabili di eseguire le query SQL. Ogni record del dataset ottenuto dalle query viene poi istanziato nel programma come un oggetto di tipo Esercizio.

4 Descrizione ad alto livello delle strutture dati e degli algoritmi utilizzati

Pattern MVC e DAO

Per strutturare l'applicazione in maniera modulare e garantire una chiara separazione delle responsabilità, sono stati adottati due pattern di progettazione: MVC (Model-View-Controller) e DAO (Data Access Object).

Il pattern MVC suddivide il sistema in tre componenti principali:

- Model – si occupa della logica applicativa e della realizzazione degli algoritmi di ottimizzazione, gestendo i dati e le regole di dominio.
- View – responsabile della presentazione grafica dei dati e della raccolta degli input forniti dall'utente tramite l'interfaccia basata su Flet.
- Controller – agisce da intermediario, ricevendo gli input dalla View, elaborandoli e delegando al Model il compito di gestire le richieste.

Il pattern DAO (Data Access Object) separa il Model dalla gestione diretta del database. Il DAO fornisce infatti un'interfaccia astratta e unificata, permettendo al Model di accedere ai dati senza preoccuparsi dei dettagli implementativi legati al database utilizzato (come il tipo di DBMS, la sintassi SQL specifica o le connessioni di rete). Ciò rende il sistema più robusto, facilmente testabile e mantenibile.

Gestione grafica e di ricerca nel DAO

Nel DAO sono definite diverse funzioni, utilizzate principalmente nella prima schermata dell'applicazione per recuperare gli esercizi dal dataset sulla base dei filtri scelti dall'utente.

Tra queste funzioni, la più rilevante è getExercisesByLivelloEMuscolo. Questa funzione ha lo scopo di recuperare dal database tutti gli esercizi che rispettano due vincoli contemporaneamente:

1. il livello di difficoltà indicato dall'utente (es. beginner, intermediate, advanced);
2. il gruppo muscolare target scelto dall'utente, che deve comparire nella lista dei muscoli coinvolti dall'esercizio.

```
query = """
SELECT e.*
FROM exercises AS e
JOIN (
  SELECT
    `Name of Exercise`,
    MIN(id_exercise) AS min_id
  FROM exercises
  WHERE `Difficulty Level` = %s
    AND FIND_IN_SET(%s, REPLACE(`Target Muscle Group`, ', ', ','))
  GROUP BY `Name of Exercise`
) AS sub
  ON e.`Name of Exercise` = sub.`Name of Exercise`
 AND e.id_exercise           = sub.min_id
```

Figura 4.1 Query presente nel DAO

La query SQL utilizzata nella funzione segue questi passaggi fondamentali:

- Normalizzazione del campo Target Muscle Group,
  tramite il comando REPLACE, viene eliminato lo spazio dopo la virgola che separa i muscoli. Questo consente di gestire correttamente l'elenco comma-separated.
- Ricerca del gruppo muscolare,
  mediante FIND_IN_SET, la query verifica che il muscolo specificato dall'utente sia contenuto all'interno della lista normalizzata. Ciò è necessario perché il campo può includere più muscoli, e la semplice uguaglianza sarebbe insufficiente.
- Filtro sul livello di difficoltà,
  viene applicato un controllo diretto sul campo Difficulty Level per selezionare esclusivamente gli esercizi compatibili con il livello indicato.
- Eliminazione dei duplicati,
  usando le istruzioni GROUP BY e MIN(id_exercise), la query raggruppa gli esercizi per nome, scegliendo per ciascun gruppo l'esercizio con il minimo identificatore univoco. In questo modo, non compaiono varianti duplicate dello stesso esercizio nella lista finale.
- Join finale,
  attraverso un join tra la tabella originale e la tabella ottenuta dalla sotto-query, vengono recuperati tutti i dettagli degli esercizi che soddisfano contemporaneamente entrambi i vincoli.

Le altre query presenti nel DAO derivano direttamente da questa logica di base. Essendo casi meno complessi (filtri applicati su un solo attributo anziché due), non sono state trattate separatamente.

Una volta recuperati gli esercizi tramite le query del DAO, il programma visualizza una lista degli esercizi filtrati. Tale lista viene utilizzata per popolare il menù a tendina della sezione destra dell'interfaccia, dalla quale l'utente può scegliere gli esercizi da inserire nella lista dei "non desiderati". A sua volta, la lista dei non desiderati sarà mostrata in un secondo menù a tendina, sempre modificabile secondo le preferenze dell'utente.

Creazione del grafo

La creazione del grafo rappresenta una fase centrale e indispensabile per il funzionamento dell'algoritmo ricorsivo. Per gestire tale struttura è stata utilizzata la libreria NetworkX, che facilita la modellazione e la gestione dei dati.

Il procedimento per creare il grafo avviene nei seguenti passaggi:

1. Vengono recuperati dal DAO tutti gli esercizi che corrispondono al livello di difficoltà scelto dall'utente.
2. Tra questi esercizi, vengono esclusi quelli inseriti dall'utente nella lista dei "non desiderati".
3. Gli esercizi che soddisfano queste condizioni vengono quindi aggiunti al grafo sotto forma di nodi.

Successivamente vengono aggiunti gli archi tra i vari nodi. Un arco collega due nodi solo se il nome degli esercizi è differente: collegare esercizi con lo stesso nome (anche se con numero diverso di serie e ripetizioni) sarebbe infatti inutile, dato che due varianti dello stesso esercizio non potrebbero essere proposte consecutivamente nello stesso piano.

Si ottiene quindi un grafo quasi completo, ossia una struttura in cui ciascun nodo è connesso a quasi tutti gli altri nodi, con l'eccezione appena descritta.

```python
def creaGrafo(self, level, listaNonDesi):

    self.grafo.clear()
    exercises = DAO.getExercisesLevel(level)

    listaDesiCompl = []
    for es in exercises:
        desi = True
        nome = es.name
        for esercizio in listaNonDesi:
            if nome == esercizio.name:
                desi = False
        if desi:
            listaDesiCompl.append(es)

    self.grafo.add_nodes_from(listaDesiCompl)


    edges = [
        (u, v) for u, v in itertools.combinations(listaDesiCompl, r: 2)
        if u.name != v.name
    ]

    self.grafo.add_edges_from(edges)
```

Figura 4.2 Funzione per creazione del grafo

Algoritmo ricorsivo per la massimizzazione delle calorie

L'algoritmo ricorsivo rappresenta il cuore del programma, essendo il più importante e complesso tra quelli implementati. Esso si basa principalmente su quattro funzioni principali.

Nella funzione getPathCalories vengono inizializzate due strutture dati fondamentali:

- pathCalories: una lista composta da tante sotto-liste quanti sono i giorni di allenamento, destinate a contenere i piani giornalieri.
- bestCals: una lista numerica, inizializzata con zeri, che memorizzerà la miglior soluzione (in termini di calorie consumate) trovata per ciascuna giornata.

Dopo aver inizializzato queste liste e verificato preliminarmente alcuni vincoli, inizia il processo ricorsivo vero e proprio. È importante applicare i controlli sui vincoli anche nella scelta del primo esercizio (nodo iniziale), perché:

- l'obiettivo è ottenere una programmazione valida su più giorni consecutivi;
- dal secondo giorno in poi, alcuni esercizi potrebbero essere esclusi se già presenti nella giornata precedente, rendendoli inadatti come punto di partenza.

```
def getPathCalories(self,numDay, timeWork):
    self.pathCalories = [[] for _ in range(numDay)]
    self.bestCals = [0 for _ in range(numDay)]

    for i in range(1, numDay+1):
        for n in self.get_nodes():

            check = False
            for ogg in self.pathCalories:
                for eser in ogg:
                    if eser.id == n.id:
                        check = True
                        break

            if not check and i >= 2:
                for esercizio in self.pathCalories[i - 2]:
                    if esercizio.name == n.name:
                        check = True
                        break

            if not check:
                if n.tempoTot <= timeWork:
                    partial = []
                    partial.append(n)
                    time_used = n.tempoTot
                    self.ricorsione(partial, i, timeWork, time_used)
```

Figura 4.3 Funzione per avviare ricorsione

Nella funzione ricorsione, ogni volta che si raggiunge una soluzione terminale (cioè non è possibile aggiungere ulteriori esercizi nel tempo disponibile), si valuta il risultato ottenuto. In particolare, tramite la funzione calcolaCals, si confronta la spesa calorica totale con il migliore risultato precedentemente trovato per la giornata. Se la nuova soluzione è migliore, essa sostituisce quella memorizzata nella lista bestCals.
Se la soluzione non è terminale (c'è ancora tempo per ulteriori esercizi), la funzione ricorsiva procede aggiungendo altri esercizi, sempre verificando che il limite di tempo totale non venga superato.

```
def ricorsione(self, partial, day, timeWork, time_used):

    n_last = partial[-1]

    neighbors = self.getAdmissibleNeighbs(n_last, partial, day, timeWork, time_used)

    if len(neighbors) == 0:
        cals_accDay = self.calcolaCals(partial)
        if cals_accDay > self.bestCals[day-1]:
            self.bestCals[day-1] = cals_accDay + 0.0
            self.pathCalories[day-1] = partial[:]
        return

    for n in neighbors:
        partial.append(n)
        new_time = time_used + n.tempoTot
        self.ricorsione(partial, day, timeWork, new_time)
        partial.pop()
```

Figura 4.4 Funzione ricorsiva

```
def calcolaCals(self, myList):
    weight = 0
    for es in myList:
        weight += es.caloriesTot
    return weight
```

Figura 4.5 Funzione per calcolare Calorie in un giorno

Infine, per ridurre la complessità computazionale e generare un piano equilibrato e realistico, sono stati applicati alcuni vincoli all'interno della funzione getAdmissibleNeighbs, invocata all'inizio di ogni chiamata ricorsiva. I vincoli inseriti sono i seguenti:

- Un esercizio (identificato da uno specifico id) non può essere ripetuto più volte nella stessa settimana.
- Lo stesso tipo di esercizio (identificato dal campo name) non può essere inserito in giorni consecutivi, né due volte nello stesso giorno, anche se caratterizzato da numeri diversi di serie e ripetizioni.
- La durata totale degli esercizi selezionati per ciascun giorno non può superare il tempo massimo specificato dall'utente.

```python
def getAdmissibleNeighbs(self, n_last, partial, day, timeWork, time_used):
    all_neigh = list(self.grafo.neighbors(n_last))
    result = []

    for es in all_neigh:

        skip = False
        for ogg in self.pathCalories:
            for eser in ogg:
                if eser.id == es.id:
                    skip = True
                    break

        if not skip and day >= 2:
            for esercizio in self.pathCalories[day-2]:
                if esercizio.name == es.name:
                    skip = True
                    break

        if not skip:
            for elem in partial:
                if elem.name == es.name:
                    skip = True
                    break

        if not skip and time_used + es.tempoTot > timeWork:
            skip = True
        # se non ho trovato alcuna corrispondenza lo aggiungo
        if not skip:
            result.append(es)
    return result
```

Figura 4.6 Funzione per impostare vincoli

Algoritmo ricorsivo per l’aumento di forza e massa muscolare

L’algoritmo che genera il piano di allenamento orientato all'aumento della forza muscolare è molto simile a quello descritto in precedenza, ma con alcune modifiche mirate per adattarlo al nuovo obiettivo.

Le differenze principali rispetto all’algoritmo precedente sono:

- L'obiettivo è ora quello di minimizzare la media dei punteggi assegnati agli esercizi selezionati per ogni giorno, anziché massimizzare il totale delle calorie consumate.
- Per questo motivo, la lista bestReps, che tiene traccia della migliore soluzione trovata per ogni giornata, non viene più inizializzata con zeri ma con valori infiniti (inf). L'utilizzo di valori infiniti permette infatti di assicurare che qualunque soluzione iniziale sarà necessariamente inferiore e quindi considerata migliore rispetto al punto di partenza.
- Quando una soluzione finale (ossia una combinazione di esercizi per cui non è possibile aggiungerne altri) viene trovata, il programma la confronta con il risultato memorizzato in precedenza. In questo caso, la soluzione memorizzata verrà aggiornata soltanto se la nuova soluzione presenta una media inferiore del punteggio.
- Infine, cambia leggermente la funzione utilizzata per calcolare il valore ottimale da confrontare: non si considera più la somma dei valori, bensì la loro media aritmetica.

5 Diagramma delle classi

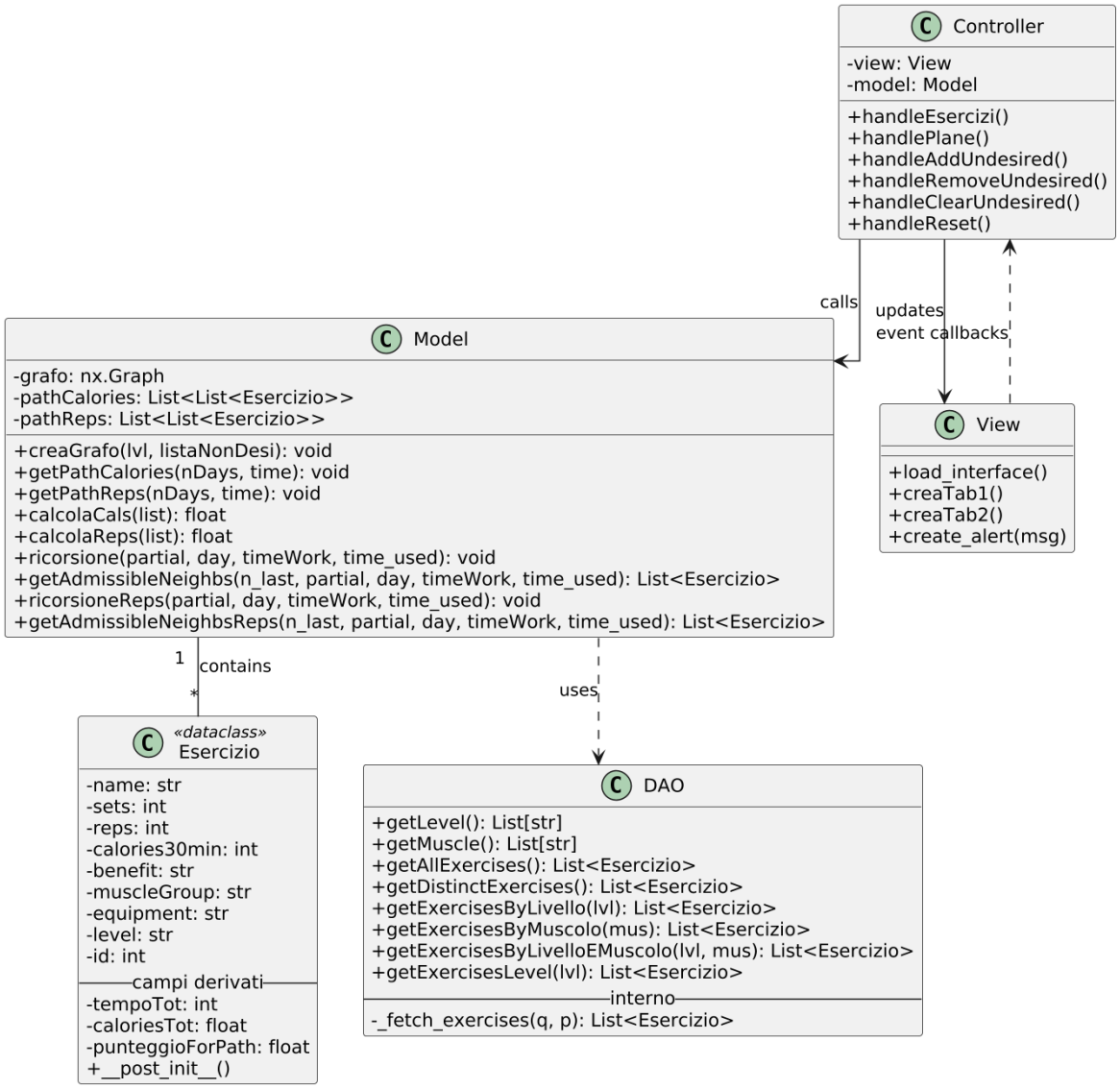

Figura 5.1 Diagramma UML delle classi

Il diagramma delle classi allegato è stato realizzato con PlantUML, un editor che permette di generare diagrammi direttamente dal codice sorgente delle classi. L'obiettivo di questo diagramma non è mostrare in modo esaustivo tutte le componenti dell'applicazione, bensì evidenziare chiaramente le parti più rilevanti per comprendere il funzionamento dell'algoritmo.

Nello specifico, le classi rappresentate sono:

- Esercizio: definisce i nodi del grafo, arricchiti da attributi derivati che vengono calcolati tramite il metodo speciale __post_init__().

- Model: gestisce la struttura del grafo e implementa i metodi necessari a crearlo, oltre agli algoritmi ricorsivi e alle funzioni collegate che realizzano la logica applicativa.
- DAO: contiene le funzioni che permettono l'accesso ai dati. Viene utilizzato direttamente dal Model per recuperare gli esercizi necessari.
- Controller e View: inserite per completezza architetturale. Il Controller gestisce l'interazione tra la UI e il Model, mentre la View contiene la logica grafica realizzata con Flet, definendo l'interfaccia utente.

Per mantenere leggibile e chiaro il programma, sono state volutamente escluse alcune funzioni di servizio, come quelle che popolano i menù a tendina (Dropdown) o che leggono e aggiornano i dati nella classe Controller.
Il diagramma punta infatti a evidenziare principalmente la catena di funzionamento centrale dell'applicazione, ovvero:

1. Recupero dei dati dal database tramite il DAO;
2. Elaborazione degli stessi dati tramite la logica contenuta nel Model;
3. Presentazione finale dei risultati tramite Controller e View.

6 Videate esemplificative dell’applicativo

È stato registrato un video dimostrativo che spiega il funzionamento dell’applicativo, lo si può trovare al seguente indirizzo:

https://youtu.be/5k4aoPnFl9Y

Qui sotto vengono invece riportate alcune videate dell’applicazione.

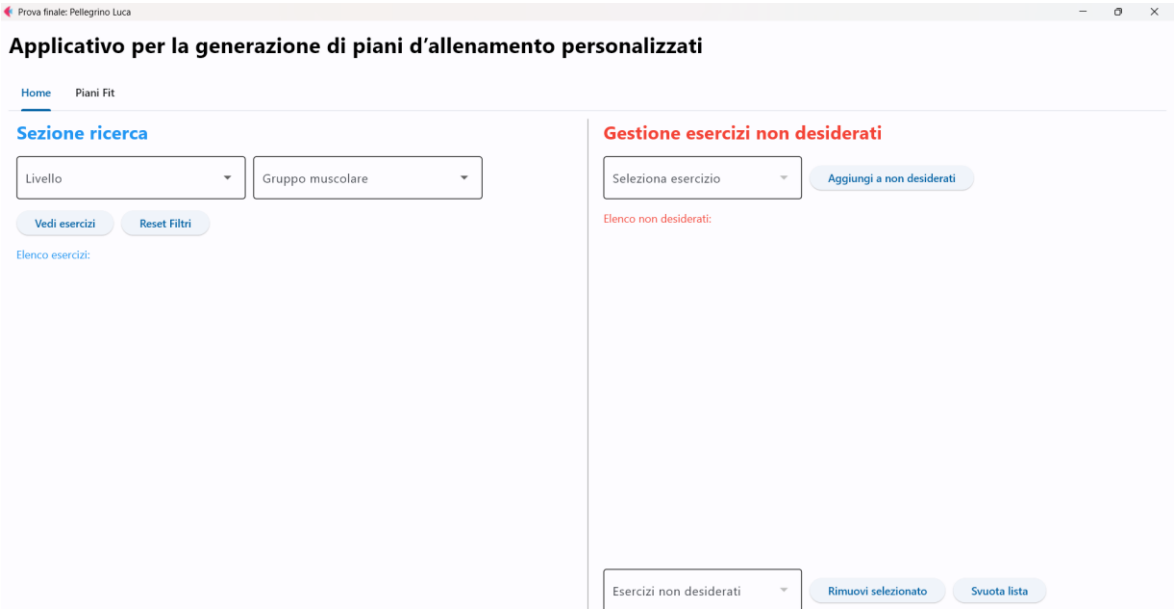

Figura 6.1 Prima schermata all'avvio

La figura 6.1 mostra la schermata iniziale dell'applicazione subito dopo l'avvio. Nella sezione di sinistra ("Sezione ricerca"), l’utente può cercare esercizi utilizzando i due menù a tendina per filtrare in base al livello di difficoltà e al gruppo muscolare target. Nella sezione di destra ("Gestione esercizi non desiderati"), invece, l'utente può gestire la lista degli esercizi che preferisce escludere dal piano. Tali esercizi possono essere aggiunti dopo essere stati selezionati dal menù superiore, e successivamente rimossi tramite il menù a tendina inferiore.

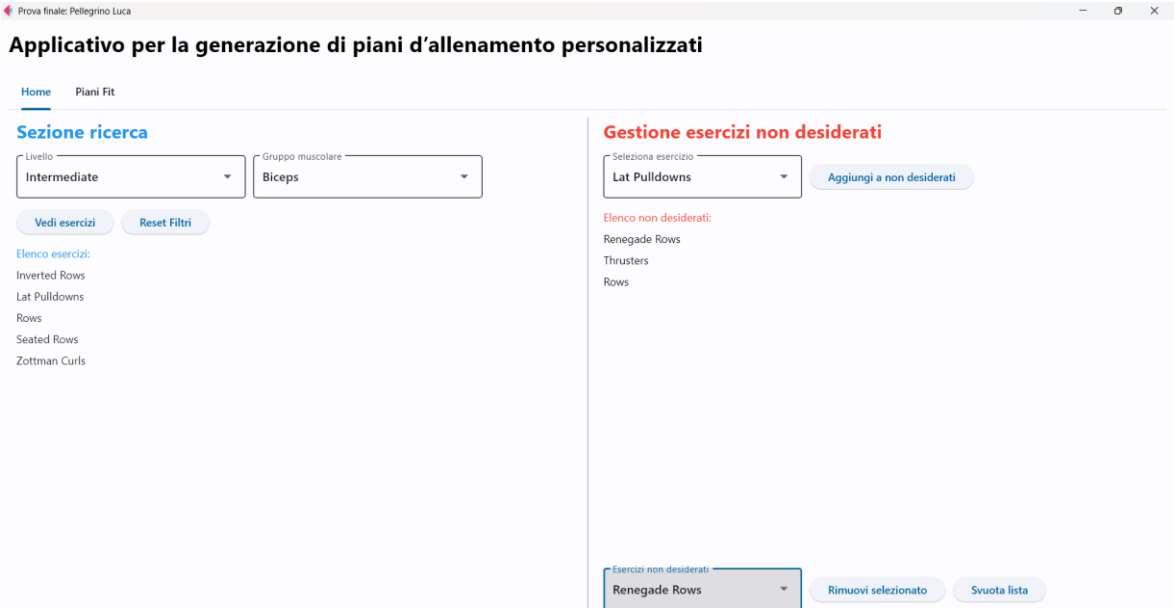

Figura 6.2 Prima schermata dopo dialogo con l'utente

La figura 6.2 rappresenta un esempio della prima schermata dopo che l'utente ha effettuato alcune selezioni e compilato parzialmente i dati.
Il menù a tendina in alto a destra viene popolato con gli esercizi risultanti dalla ricerca effettuata nella parte sinistra. Quello inferiore, invece, mostra gli esercizi già inseriti nella lista dei non desiderati.

I due pulsanti presenti nella sezione permettono di:

- Rimuovere il singolo esercizio selezionato tramite il pulsante "Rimuovi selezionato".
- Svuotare completamente la lista degli esercizi non desiderati tramite il pulsante "Svuota lista".

Il programma gestisce automaticamente diverse eccezioni, come ad esempio il tentativo di aggiungere un esercizio che risulta già presente nella lista dei non desiderati. Inoltre, sebbene nel dataset originale alcuni esercizi si ripetano (differenziandosi solo per numero di serie e ripetizioni), il programma mostra un'unica voce per esercizio nella ricerca iniziale; nel caso in cui uno di essi venga inserito nella lista dei non desiderati, tutte le varianti dell'esercizio con lo stesso nome vengono automaticamente escluse.

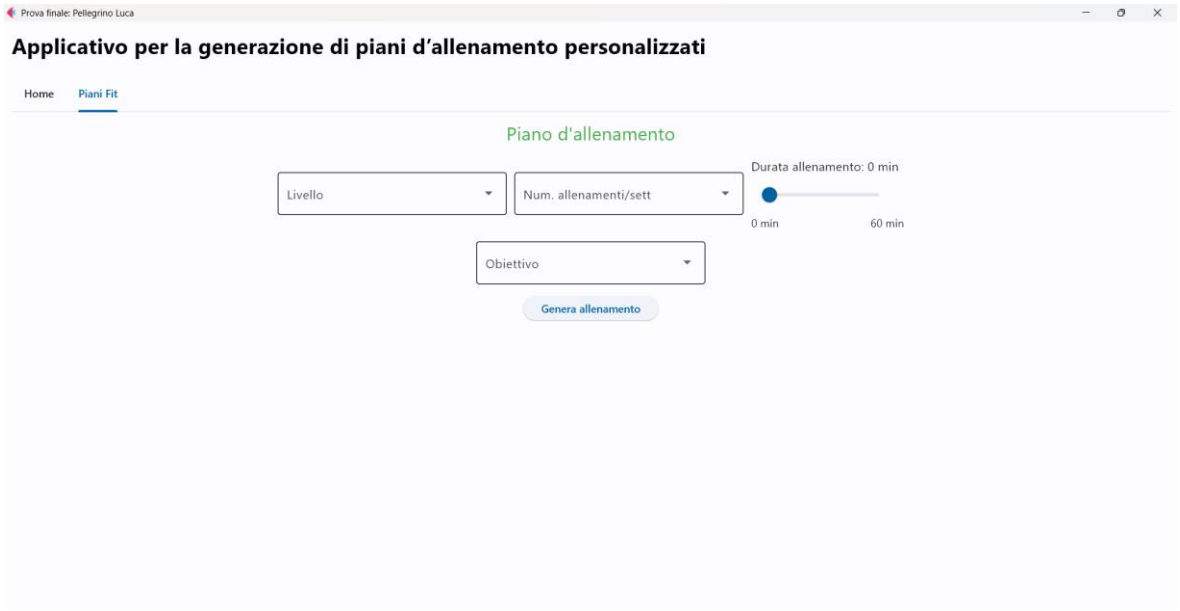

Figura 6.3 Seconda schermata all'avvio

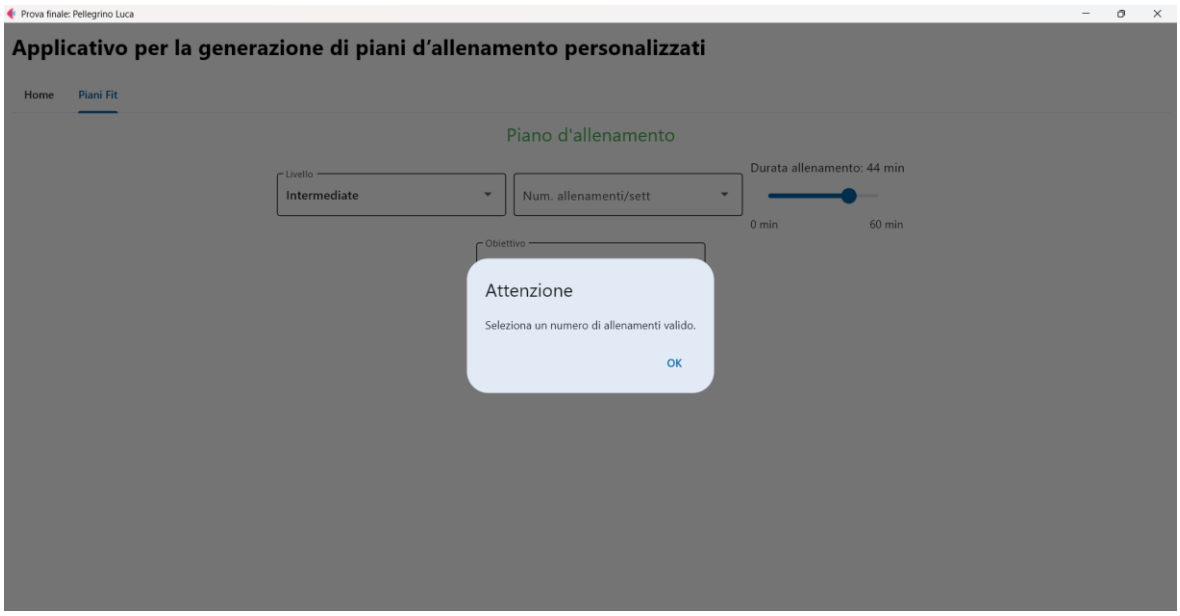

Figura 6.4 Esempio di generazione di avviso

Le figure successive (6.3 e 6.4) illustrano la seconda schermata dell'applicazione, ovvero quella dedicata alla generazione del piano di allenamento.

La figura 6.3 mostra un esempio di seconda schermata prima che l'utente compili tutti i campi necessari. Se l'utente prova a premere il pulsante "Genera allenamento" senza aver

inserito tutti i dati richiesti (come durata, livello o obiettivo), il programma visualizza un avviso informativo, richiedendo all'utente di completare tutti i campi obbligatori (figura 6.4).

Indipendentemente da questa verifica, alla selezione dell'obiettivo del piano (focus), viene sempre visualizzato un avviso che spiega chiaramente all'utente quali parametri il programma ottimizzerà (massimizzerà o minimizzerà) durante la generazione del piano.

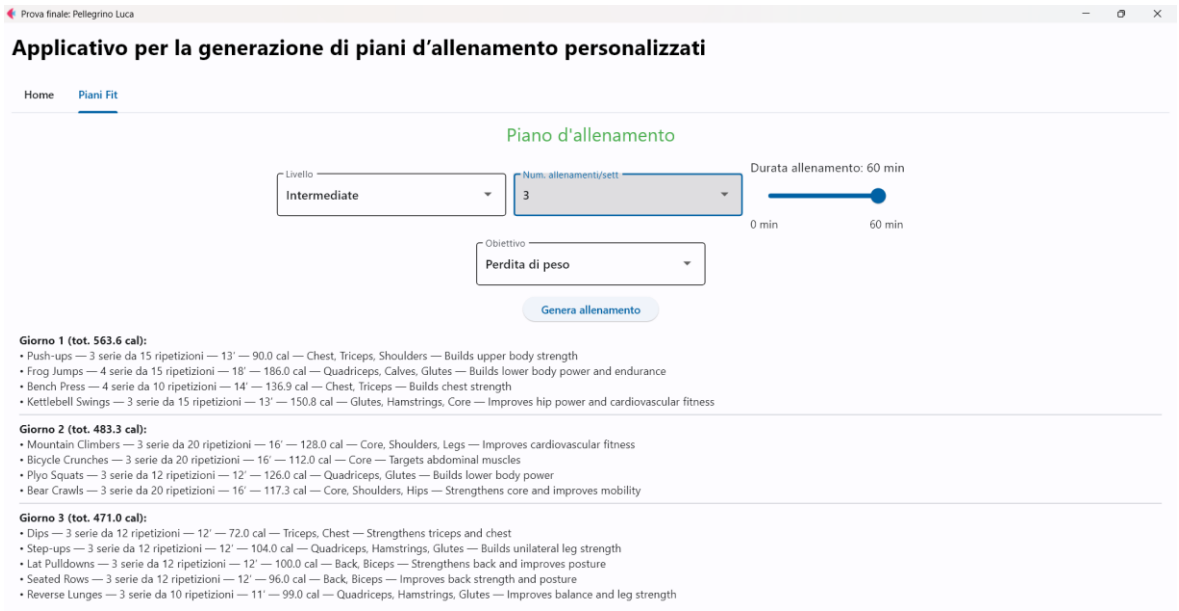

Figura 6.5 Esempio piano con focus "perdita di peso"

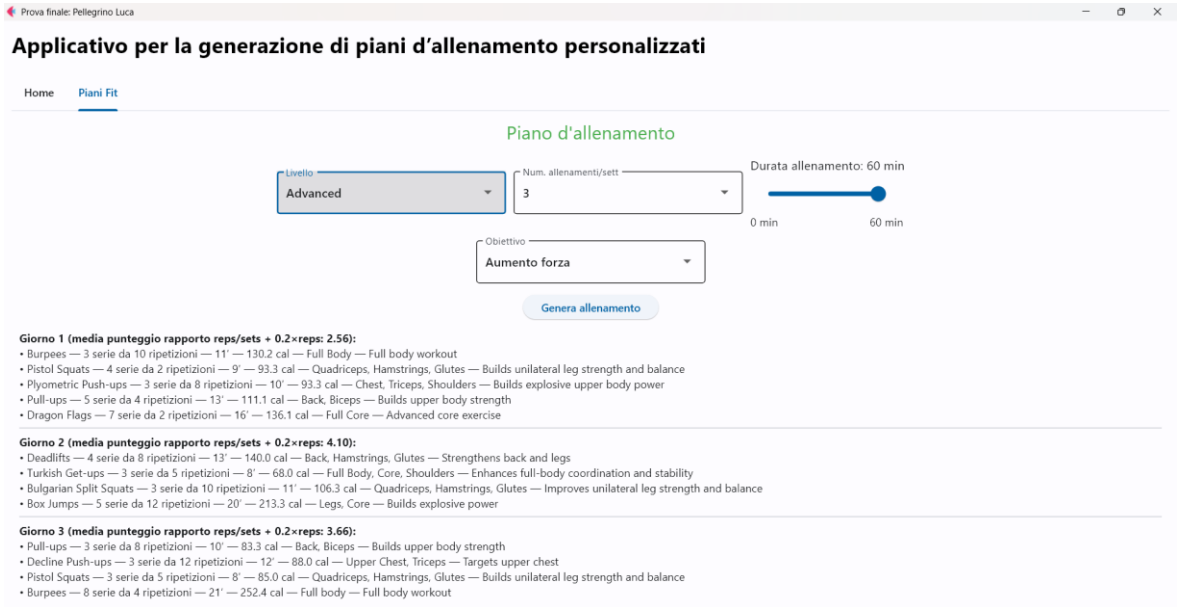

Figura 6.6 Esempio piano con focus "aumento forza"

Le ultime due figure (6.5 e 6.6) mostrano esempi concreti di piani d'allenamento generati automaticamente dall'applicativo.

- Nel primo caso (figura 6.5, focus "perdita di peso"), accanto a ciascun giorno di allenamento è indicato chiaramente il totale delle calorie consumate dagli esercizi selezionati per quella giornata.
- Nel secondo caso (figura 6.6, focus "aumento forza"), viene mostrata invece la media dei punteggi calcolati per gli esercizi di ogni singola giornata, punteggi che l'algoritmo utilizza per ottimizzare il piano nella direzione di privilegiare esercizi con meno ripetizioni e maggior carico.

Entrambi gli esempi mostrano la capacità dell'applicazione di generare piani equilibrati e coerenti con gli obiettivi selezionati.

7 Risultati e valutazioni

Risultati ottenuti

Per analizzare i risultati ottenuti possiamo esaminare i due esempi di piani generati nelle figure 6.5 e 6.6. Si osserva come entrambi siano ben bilanciati: nessun esercizio viene ripetuto in giorni consecutivi (rispettando così il vincolo imposto), e quando uno stesso tipo di esercizio compare più volte nella settimana si tratta sempre di varianti diverse, caratterizzate da numeri differenti di serie e ripetizioni.

Analizzando più nello specifico i risultati relativi alle calorie consumate (primo piano) e al numero di ripetizioni (secondo piano), si nota come l'algoritmo funzioni correttamente. Nel primo caso, infatti, il consumo calorico giornaliero è sempre elevato, mentre nel secondo caso prevalgono esercizi con un basso numero di ripetizioni e un punteggio complessivo contenuto, coerentemente con l'obiettivo di aumentare la forza.

Tempi d'esecuzione

La sfida più grande degli algoritmi ricorsivi è rappresentata dalla gestione dei tempi di esecuzione. Per affrontare questo aspetto, il programma cerca di minimizzare la complessità computazionale, delegando, quando possibile, buona parte della logica alle query SQL, più efficienti rispetto alle elaborazioni in memoria. Inoltre, sono stati inseriti numerosi vincoli nel processo ricorsivo, per bloccare tempestivamente l'esplorazione di rami poco promettenti e alleggerire il carico computazionale.

Per valutare concretamente i tempi di esecuzione, sono stati testati i casi più impegnativi del programma: numero massimo di giorni (5), durata massima della sessione (60 minuti), per ciascuno dei tre livelli di difficoltà (beginner, intermediate, advanced) e per entrambi gli obiettivi disponibili (perdita di peso, aumento forza).

Da questi test emerge che la durata dell'allenamento rappresenta il fattore determinante per la velocità di calcolo. Con sessioni fino a 45 minuti, i risultati sono quasi istantanei, mentre con sessioni più lunghe si possono verificare tempi di attesa fino a qualche secondo, comunque del tutto accettabili.

Nello specifico, per sessioni da 5 giorni di 60 minuti ciascuno, i tempi di elaborazione ottenuti sono stati:

- Livello Beginner: 6 secondi (focus calorie) e 10 secondi (focus forza)

- Livello Intermediate: 9 secondi (focus calorie) e 6 secondi (focus forza)
- Livello Advanced: 4 secondi (focus calorie) e 5 secondi (focus forza)

Come si può notare, tali tempi risultano contenuti e ragionevoli, anche se potrebbero aumentare ulteriormente al crescere della durata delle sessioni. Le differenze nei tempi tra i vari casi dipendono principalmente dalla struttura del grafo e dalla presenza dei vincoli, che in alcune situazioni rendono più o meno rapida la ricerca della soluzione ottimale.

Punti di forza

Tra gli aspetti positivi principali dell'applicativo possiamo individuare:

- Pertinenza dei piani: i piani generati rispettano coerentemente l'obiettivo impostato, proponendo esercizi adeguati e ben combinati.
- Interfaccia grafica: grazie a Flet è stata realizzata un'interfaccia intuitiva e semplice da utilizzare, che limita la possibilità di errori e gestisce in modo efficace le eccezioni.
- Uso di query SQL: tale scelta ha permesso di alleggerire notevolmente la complessità computazionale dell'applicativo, riducendo i tempi necessari per recuperare e gestire i dati.
- Flessibilità e personalizzazione: l'applicativo consente un alto grado di flessibilità nell'inserimento dei vincoli e nella definizione delle preferenze dell'utente.

.

Punti di debolezza

Gli aspetti che presentano margini di miglioramento includono:

- Limitata disponibilità di dataset estesi e dettagliati: trovare su Kaggle dataset di dimensioni maggiori e con informazioni più dettagliate non è risultato semplice. Un dataset più ampio e completo consentirebbe di introdurre ulteriori vincoli (come la diversificazione degli esercizi per gruppi muscolari) e offrire così un servizio ancora più personalizzato ed efficace.
- Tempi di esecuzione variabili: a causa della struttura del grafo generato per l'algoritmo ricorsivo, i tempi di elaborazione possono risultare sensibilmente differenti in base ai parametri inseriti dall'utente.

Conclusioni

Nel complesso, l’applicativo ha dimostrato di essere uno strumento efficace e funzionale per generare piani di allenamento personalizzati settimanali, rispondendo pienamente agli obiettivi stabiliti in fase di progettazione. Non sono emerse problematiche di difficile gestione e, grazie alla struttura modulare e alla chiarezza progettuale, potranno essere facilmente implementate ulteriori funzionalità in futuro. Inoltre, l'applicazione potrà beneficiare della ricerca di nuovi dataset, che consentiranno uno sviluppo e un aggiornamento costante delle sue potenzialità.

8 Licenza

Tu sei libero di:

1. Condividere — riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare questo materiale con qualsiasi mezzo e formato
2. Modificare — remixare, trasformare il materiale e basarti su di esso per le tue opere
3. Il licenziante non può revocare questi diritti fintanto che tu rispetti i termini della licenza.

Alle seguenti condizioni:

1. Attribuzione — Devi riconoscere una menzione di paternità adeguata , fornire un link alla licenza e indicare se sono state effettuate delle modifiche . Puoi fare ciò in qualsiasi maniera ragionevole possibile, ma non con modalità tali da suggerire che il licenziante avalli te o il tuo utilizzo del materiale.
2. NonCommerciale — Non puoi utilizzare il materiale per scopi commerciali .
3. StessaLicenza — Se remixi, trasformi il materiale o ti basi su di esso, devi distribuire i tuoi contributi con la stessa licenza del materiale originario.
4. Divieto di restrizioni aggiuntive — Non puoi applicare termini legali o misure tecnologiche che impongano ad altri soggetti dei vincoli giuridici su quanto la licenza consente loro di fare.

Note:

Non sei tenuto a rispettare i termini della licenza per quelle componenti del materiale che siano in pubblico dominio o nei casi in cui il tuo utilizzo sia consentito da una eccezione o limitazione prevista dalla legge.

Non sono fornite garanzie. La licenza può non conferirti tutte le autorizzazioni necessarie per l'utilizzo che ti prefiggi. Ad esempio, diritti di terzi come i diritti all'immagine, alla riservatezza e i diritti morali potrebbero restringere gli usi che ti prefiggi sul materiale.