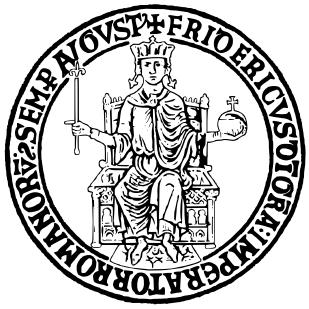
Università degli Studi di Napoli Federico II



Scuola Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica

Exercise Recognition and Automated Repetition Counting in Real Time using AI

Professore
Roberto Pietrantuono

Studenti
Giuseppe Mazzocca M63001610
Angelo A. Nozzolillo M63001602
Pierluigi Montieri M63001584

Anno Accademico 2024/2025

Indice

Introduzione

Contesto e Motivazione

Il settore del fitness sta vivendo una trasformazione guidata dalla tecnologia, con un aumento esponenziale della richiesta di soluzioni flessibili, personalizzate ed economiche. Tuttavia persistono diverse criticità:

- **Frammentazione dell'attenzione durante gli allenamenti**: Gran parte degli utenti amatoriali interrompe gli esercizi per annotare serie, ripetizioni o gestire i tempi di recupero, compromettendo qualità e sicurezza.
- **Mancanza di feedback immediati**: L'assenza di correzioni tecniche in tempo reale (es. postura errata) espone al rischio di infortuni, specialmente per i principianti.
- **Monitoraggio inefficace dei progressi**: La registrazione manuale di esercizi e ripetizioni è soggetta a errori (es. dimenticanze, conteggi approssimativi) e risulta dispersiva, soprattutto in sessioni intense. Ciò genera dati inaffidabili per valutare i progressi e demotiva gli utenti, che non percepiscono miglioramenti oggettivi.
- **Alti costi del coaching professionale**: Servizi di personal trainer qualificati rimangono inaccessibili a fasce di utenti con budget limitati.
- **Limiti nel supporto ai professionisti**: I trainer faticano a integrare strumenti digitali per il monitoraggio remoto, riducendo l'efficacia del coaching ibrido.

Questi gap evidenziano la necessità di un'applicazione che unisca intelligenza artificiale, analisi dati e usabilità per rendere accessibile un fitness di qualità, sia in palestra che a casa.

Obiettivi del Progetto

Il progetto si propone di sviluppare un'applicazione innovativa per il fitness, con l'obiettivo di migliorare l'esperienza di allenamento attraverso diverse funzionalità avanzate.
Un primo obiettivo è ottimizzare l'interazione con l'utente, implementando un sistema basato su intelligenza artificiale per il riconoscimento automatico degli esercizi e il conteggio delle ripetizioni. Questo permetterà di fornire feedback in tempo reale sugli errori di esecuzione, contribuendo a ridurre il rischio di infortuni.
Un altro aspetto chiave è la gestione del tempo: si prevede di introdurre timer automatici per le pause, in modo che l'utente possa concentrarsi sull'allenamento senza dover avviare manualmente i conteggi.
Per aumentare la motivazione e il monitoraggio dei progressi, il progetto prevede l'integrazione di un sistema di archiviazione dei dati storici, con visualizzazioni intuitive come grafici e dashboard.
L'applicazione sarà progettata anche per supportare i professionisti del fitness, offrendo un modulo dedicato alla gestione dei clienti. Questo includerà strumenti per l'invio di schede di allenamento digitali, il monitoraggio dei progressi in tempo reale e la creazione di report personalizzati.
Infine, il progetto punta a rendere accessibili funzionalità avanzate a un costo contenuto, sostituendo alcune attività base di un personal trainer con un sistema intelligente, garantendo così un'esperienza di qualità senza costi elevati per l'utente finale.

Questo progetto nasce dalla volontà di colmare il divario tra innovazione tecnologica e bisogni concreti del mondo del fitness. Combinando automazione, precisione e accessibilità, l'applicazione si posiziona non solo come strumento di tracking, ma come partner intelligente per chiunque voglia trasformare i propri obiettivi sportivi in risultati misurabili.[Pie25]

Capitolo 1

Requirements Analysis

1.1 Analisi delle Necessità degli Stakeholder

In questa sezione vengono descritti i principali stakeholder del progetto, ossia coloro che sono direttamente o indirettamente coinvolti nell'uso del sistema e le loro esigenze.

1. **Utenti finali (atleti e appassionati di fitness)**: Gli utenti finali, che vanno dagli atleti professionisti agli appassionati di fitness, sono i principali destinatari del sistema di monitoraggio. Le loro esigenze principali sono:

    - **Precisione nel conteggio delle ripetizioni**: Gli utenti desiderano un sistema che registri accuratamente ogni ripetizione degli esercizi, fornendo un feedback tempestivo. L'accuratezza è essenziale per garantire un allenamento efficace e per evitare errori nel monitoraggio delle performance.
    - **Gestione automatica del tempo di pausa**: Il sistema deve fornire un timer automatico che si avvii alla fine di ogni serie e che scandisca il tempo di recupero. Durante la pausa, il sistema può fornire informazioni per mantenere la concentrazione e prepararsi alla serie successiva.
    - **Monitoraggio della tecnica di esecuzione**: Oltre al conteggio delle ripetizioni, gli utenti sono interessati a migliorare la qualità dei loro esercizi, ricevendo suggerimenti utili per correggere errori posturali o di movimento. Il sistema deve essere in grado di riconoscere variazioni nella corretta esecuzione degli esercizi e fornire un feedback sugli errori posturali o di movimento. In caso di esecuzione errata, la ripetizione non deve essere conteggiata.
    - **Registrazione storica degli allenamenti**: È fondamentale che il sistema offra la possibilità di registrare e visualizzare i dati relativi agli allenamenti precedenti. Questi dati aiutano gli utenti a tracciare i propri progressi nel tempo, a confrontare le performance attuali con quelle passate e a capire se gli obiettivi prefissati sono stati raggiunti.

- **Facilità d'uso e accessibilità**: L'interfaccia del sistema deve essere semplice, intuitiva e facilmente accessibile, in modo che gli utenti possano concentrarsi completamente sull'esercizio senza necessità di interrompere l'attività per configurare il sistema. Deve essere adatto anche a utenti non esperti in tecnologia.

2. **Allenatori e personal trainer**: I personal trainer sono un altro gruppo di stakeholder importante. Utilizzano il sistema per monitorare i progressi dei propri clienti e per ottimizzare i programmi di allenamento. Le loro esigenze principali includono:

   - **Registrazione storica degli allenamenti**: Per i personal trainer, è essenziale avere accesso alla cronologia degli allenamenti dei propri clienti. Questo permette di monitorare l'evoluzione delle performance nel tempo, di osservare eventuali miglioramenti o aree che necessitano di maggiore attenzione e di personalizzare ulteriormente il programma di allenamento. La visione dei progressi storici aiuta i trainer a ottimizzare le sessioni future, migliorando l'efficacia dei programmi d'allenamento.
   - **Gestione proattiva dei malfunzionamenti del sistema**: Monitorare le statistiche di sistema tramite una dashboard è cruciale per garantire che il sistema funzioni correttamente in ogni momento. Tracciando lo stato del software, le performance e gli eventuali errori in tempo reale, si possono identificare rapidamente problemi tecnici che potrebbero interferire con l'esperienza dell'utente o con la qualità dell'allenamento. Una gestione tempestiva dei malfunzionamenti evita disagi e garantisce un'esperienza fluida per gli utenti.

1.2 Requisiti Funzionali

I requisiti funzionali definiscono le capacità essenziali del sistema affinché possa operare in modo efficace e soddisfare le esigenze degli utenti finali e dei personal trainer, tra questi possiamo elencare:

- **Monitoraggio degli allenamenti**: Il sistema deve permettere di tracciare in tempo reale le sessioni di allenamento degli utenti, registrando parametri come durata, numero di ripetizioni, intensità e tipologia di esercizio svolto.
- **Valutazione della performance**: Deve essere in grado di fornire un'analisi dettagliata delle prestazioni degli utenti.
- **Registrazione storica dei dati**: Il sistema deve mantenere un archivio consultabile degli allenamenti passati.
- **Gestione multiutente e personalizzazione**: Ogni utente deve poter accedere a un profilo personalizzato che tenga conto del proprio livello di

allenamento, degli obiettivi e delle preferenze, mentre i personal trainer devono poter gestire più utenti simultaneamente.

- **Dashboard di analisi**: Un'interfaccia intuitiva deve offrire statistiche aggregate e insight basati sui dati raccolti, aiutando gli utenti a prendere decisioni informate sul proprio allenamento e consentendo ai personal trainer di ottimizzare i piani di esercizio.

1.3 Requisiti Non Funzionali e FASTEPS

I requisiti non funzionali definiscono le caratteristiche qualitative del sistema, garantendo che operi in modo affidabile, sicuro e performante. Di seguito sono elencati i principali requisiti non funzionali dell'applicazione di riconoscimento degli esercizi basata su AI.

- **Prestazioni e Scalabilità**
  - Il sistema deve elaborare i dati in **tempo reale**, fornendo feedback immediato sugli esercizi.
  - Deve supportare almeno **1000 utenti simultanei** senza degradare le prestazioni.
  - Il tempo di risposta per il riconoscimento di un esercizio non deve superare i **50 ms**.
  - Il sistema deve supportare **scalabilità verticale e orizzontale** per gestire un numero crescente di utenti.
- **Sicurezza**
  - Deve essere implementata un'autenticazione **a due fattori (2FA)** per l'accesso degli utenti admin.
  - Gli accessi al sistema devono essere **loggati e monitorati** per individuare tentativi di accesso non autorizzati.
- **Affidabilità e Disponibilità**
  - Deve essere implementato un meccanismo di **backup automatico** con cadenza giornaliera.
  - In caso di errore o crash, l'app deve **recuperare lo stato dell'ultima sessione** senza perdita di dati.
- **Manutenibilità ed Estensibilità**
  - Il codice deve essere **modulare e documentato**, per facilitare future modifiche e aggiornamenti.
  - Deve essere possibile integrare nuovi algoritmi di riconoscimento senza dover modificare l'architettura principale.

Abbiamo considerato i requisiti non funzionali del sistema secondo il framework **FASTEPS**, che garantisce che il sistema di riconoscimento degli esercizi basato su AI sia equo, sicuro, trasparente e conforme agli standard etici e di privacy.

- **Fairness**
  - Il sistema deve garantire un trattamento equo per tutti gli utenti, evitando discriminazioni nell'analisi degli esercizi.
  - Il modello di AI deve essere addestrato con un dataset bilanciato per minimizzare bias nei risultati e garantire prestazioni uniformi per utenti con diverse caratteristiche fisiche.
- **Accountability**
  - Deve essere possibile tracciare e monitorare tutte le decisioni prese dal sistema AI.
  - Il sistema deve fornire log dettagliati e strumenti di debugging per verificare il corretto funzionamento del modello di AI.
- **Security**
  - Il sistema deve implementare protocolli di sicurezza avanzati per prevenire accessi non autorizzati e proteggere i dati degli utenti.
  - Il sistema deve supportare autenticazione sicura, come OAuth 2.0 e gestione sicura delle sessioni utente.
- **Transparency**
  - Gli utenti devono poter comprendere come vengono elaborate le loro informazioni e come il sistema prende decisioni, in particolare per il riconoscimento degli esercizi.
  - Deve essere disponibile una documentazione chiara sulle tecnologie e i modelli utilizzati nel sistema.
- **Ethics**
  - Il sistema deve rispettare i principi etici nello sviluppo dell'AI, evitando bias e discriminazioni nei risultati del riconoscimento.
  - L'uso del riconoscimento dei movimenti deve essere conforme alle normative su *AI e sorveglianza*, evitando raccolta dati invasiva o non necessaria.
  - Devono essere previsti meccanismi di revisione indipendenti per verificare la conformità etica del sistema.

- **Privacy**
  - Gli utenti devono poter controllare i propri dati, con possibilità di accesso, modifica e cancellazione su richiesta.
  - Devono essere implementate politiche di **data minimization**, raccogliendo solo le informazioni strettamente necessarie al funzionamento del sistema.
- **Safety**
  - Il sistema deve garantire che il riconoscimento degli esercizi non generi errori che possano causare danni fisici agli utenti.
  - Il software deve funzionare in modo affidabile anche in condizioni di variazione della qualità del video (scarsa illuminazione, angolazioni diverse).

L'analisi dei requisiti è fondamentale per definire le funzionalità chiave del sistema e garantire che sia scalabile, affidabile e conforme agli standard di sicurezza e privacy. I requisiti funzionali assicurano che il sistema fornisca un supporto efficace agli utenti e ai personal trainer, mentre i requisiti non funzionali, definiti secondo il framework FASTEP, garantiscono l'integrità, la trasparenza e la sicurezza dell'intelligenza artificiale. Questi elementi costituiscono la base per la fase successiva di progettazione e sviluppo del sistema.

Capitolo 2

Progettazione

2.1 Architettura del Sistema

L'applicazione è stata progettata seguendo un'architettura modulare e scalabile, organizzata in diversi componenti chiave che interagiscono tra loro:

2.1.1 Frontend

- **Interfaccia Web Responsive**: Implementata utilizzando HTML5 e JavaScript per garantire accessibilità da diversi dispositivi
- **Video Stream Processing**: Sistema real-time di elaborazione del flusso video della webcam per il tracciamento dei movimenti
- **Dashboard Interattiva**: Visualizzazione in tempo reale dei dati dell'allenamento e dello storico delle performance

2.1.2 Backend

- **Server Flask**: Core dell'applicazione che gestisce le richieste HTTP e coordina i vari moduli
- **Sistema di Autenticazione**: Implementato utilizzando Keycloak per garantire sicurezza e gestione dei ruoli utente
- **Engine di Computer Vision**:
  - Modulo MediaPipe per il rilevamento delle pose
  - Modello SVM pre-addestrato per il riconoscimento degli esercizi
  - Sistema di conteggio ripetizioni specifico per ogni tipo di esercizio

2.1.3 Moduli Funzionali

Motion Detection System

- Analisi real-time dei keypoint del corpo
- Calcolo degli angoli tra le articolazioni
- Validazione delle posture corrette

Exercise Management

- Conteggio automatico delle ripetizioni
- Gestione dei tempi di riposo
- Sistema di feedback

Data Storage & Analytics

- Tracciamento dello storico allenamenti
- Generazione di report e statistiche
- Persistenza dei dati utente

2.2 Tecnologie Selezionate

2.2.1 Framework e Librerie

Flask

- Framework Python leggero e flessibile
- Facilità di integrazione con altre librerie Python
- Supporto per WebSocket per comunicazioni real-time
- Gestione efficiente delle sessioni utente

MediaPipe

- Framework di Google per Computer Vision
- Modelli pre-addestrati per il rilevamento pose
- Ottimizzato per elaborazione in tempo reale
- Alta precisione nel tracciamento dei keypoint

OpenCV (cv2)

- Elaborazione video stream in tempo reale
- Integrazione efficiente con MediaPipe
- Funzionalità avanzate di image processing
- Ampio supporto per operazioni di computer vision

2.2.2 Strumenti di Machine Learning

Scikit-learn

- Implementazione del modello SVM per classificazione esercizi
- Facilità di addestramento e serializzazione dei modelli
- Strumenti per valutazione delle performance

2.2.3 Sistema di Autenticazione

Keycloak

- Sistema SSO robusto e sicuro
- Gestione avanzata dei ruoli utente
- Supporto per OAuth2 e OpenID Connect
- Facilità di integrazione con applicazioni web

2.2.4 Librerie di Supporto

pyttsx3

- Engine di sintesi vocale per feedback audio
- Supporto multipiattaforma
- Personalizzazione del voice output

NumPy e Pandas

- Elaborazione efficiente dei dati
- Manipolazione delle feature per il modello ML
- Analisi statistica delle performance

2.2.5 Frontend Technologies

HTML5 & JavaScript

- Interfaccia web responsive
- Gestione dinamica dei contenuti

Le tecnologie sono state selezionate considerando:

- Performance nell'elaborazione real-time
- Scalabilità e manutenibilità del codice
- Robustezza e affidabilità
- Supporto della community e documentazione
- Compatibilità cross-platform

2.2.6 Flusso principale

1. **Acquisizione dei dati**: La webcam cattura il video dell'utente durante l'allenamento.
2. **Elaborazione con AI**: Il modello MediaPipe analizza la postura e invia i dati a un modello SVM per il riconoscimento dell'esercizio.
3. **Conteggio delle ripetizioni**: Il backend elabora le informazioni per tracciare le ripetizioni e identificare gli errori di esecuzione.
4. **Feedback in tempo reale**: Il sistema fornisce feedback vocale tramite sintesi vocale (pyttsx3) e mostra informazioni sullo schermo.
5. **Archiviazione dei dati**: Gli allenamenti vengono memorizzati in uno storico per il monitoraggio dei progressi.
6. **Autenticazione e gestione utenti**: Keycloak viene utilizzato per garantire un accesso sicuro e gestire i ruoli utente.

2.3 Casi d'Uso Principali

Attori Principali

I principali attori coinvolti nell'applicazione sono:

- **Utente Finale:** Utilizzatore principale dell'applicazione fitness
- **Amministratore:** Responsabile della gestione e manutenzione del sistema

2.3.1 Casi d'Uso per Utente Finale

Impostazione Target Ripetizioni

Attore Primario	Utente Finale
Obiettivo	Personalizzazione dell'obiettivo di allenamento
Descrizione	Definizione del numero di ripetizioni target per la sessione di esercizio
Flusso Base	1. Accedere all'interfaccia di allenamento 2. Selezionare il pulsante "Target Ripetizioni" 3. Impostare il numero target di ripetizioni 4. Confermare la configurazione
Precondizioni	Utente autenticato, dispositivo funzionante
Postcondizioni	Target ripetizioni impostato per la sessione

Allenamento

Attore Primario	Utente Finale
Obiettivo	Inizio della sessione di esercizio
Descrizione	Attivazione del sistema di riconoscimento automatico degli esercizi e conteggio delle ripetizioni
Flusso Base	1. Posizionarsi davanti alla webcam 2. Avviare la sessione di allenamento 3. Aspettare il count-down 4. Eseguire l'esercizio
Precondizioni	Target ripetizioni impostato, webcam funzionante
Postcondizioni	Ripetizioni conteggiate, progressi registrati

Riepilogo Allenamento

Attore Primario	Utente Finale
Obiettivo	Analisi e visualizzazione dei risultati dell'allenamento
Descrizione	Generazione e presentazione di un report dettagliato delle performance
Flusso Base	1. Cliccare "Riepilogo" 2. Il sistema genera un report automatico 3. Visualizzare dettagli (esercizi svolti, numero ripetizioni, durata sessione) 4. Analizzare le proprie performance
Precondizioni	Sessione di allenamento in corso
Postcondizioni	Report generato e visualizzato

Reset Allenamento

Attore Primario	Utente Finale
Obiettivo	Interruzione e azzeramento della sessione corrente
Descrizione	Possibilità di interrompere l'allenamento e ricominciare da zero
Flusso Base	1. Richiedere il reset dell'allenamento 2. Il sistema azzera tutti i contatori 3. Cancellazione della sessione corrente
Precondizioni	Sessione di allenamento in corso
Postcondizioni	Sessione azzerata, pronta per un nuovo inizio

Segnalazione Problema

Attore Primario	Utente Finale
Obiettivo	Reporting di errori o malfunzionamenti
Descrizione	Invia una segnalazione al supporto tecnico riguardo il mal-riconoscimento automatico di uno specifico esercizio
Flusso Base	1. Identificare un problema nell'applicazione 2. Accedere all'interfaccia di segnalazione 3. Selezionare l'esercizio da segnalare 4. Inviare la segnalazione al sistema
Precondizioni	Utente autenticato, problema riscontrato
Postcondizioni	Segnalazione registrata nel sistema di supporto

2.3.2 Casi d'Uso per Amministratore

Visualizzazione Dashboard di Sistema

Attore Primario	Amministratore
Obiettivo	Monitoraggio complessivo dell'applicazione e analisi delle metriche di sistema
Descrizione	Analisi delle metriche di sistema e utilizzo
Flusso Base	1. Admin accede con credenziali di sistema 2. Visualizza dashboard principale 3. Analizza: • Numero di utenti attivi • Utilizzo dell'applicazione • Performance del sistema • Statistiche degli allenamenti
Precondizioni	• Autenticazione con ruolo amministrativo • Connessione al sistema funzionante
Postcondizioni	• -

Riavvio del Server

Attore Primario	Amministratore
Obiettivo	Procedura di riavvio completo dell'infrastruttura applicativa
Descrizione	Gestione dei servizi di sistema
Flusso Base	1. Admin accede all'interfaccia di gestione server 2. Seleziona opzione di riavvio 3. Conferma l'operazione 4. Sistema esegue shutdown e riavvio 5. Verifica dello stato dei servizi post-riavvio
Precondizioni	• Autenticazione con privilegi di amministratore root • Nessuna operazione critica in corso
Postcondizioni	• Tutti i servizi riavviati correttamente • Stato del sistema verificato e funzionante • Log di riavvio generato e archiviato

Backup del Database

Attore Primario	Amministratore
Obiettivo	Creazione di copie dei dati
Descrizione	Preservazione delle informazioni critiche del sistema
Flusso Base	1. Admin accede all'interfaccia di backup 2. Seleziona tipologia di backup: • Completo • Incrementale 3. Avvia procedura di backup 4. Sistema genera file di backup 5. Verifica integrità del backup generato
Precondizioni	• Spazio di archiviazione sufficiente • Connessione al database funzionante • Autenticazione con privilegi di backup
Postcondizioni	• File di backup creato e salvato in posizione sicura • Integrità dei dati verificata • Timestamp e metadati del backup registrati

Visualizzazione System Logs

Attore Primario	Amministratore
Obiettivo	Analisi dettagliata degli eventi di sistema
Descrizione	Monitoraggio e troubleshooting
Flusso Base	1. Admin accede all'interfaccia dei log 2. Seleziona periodo di interesse 3. Applica filtri di ricerca: • Tipo di evento • Livello di severità • Componente sistema 4. Visualizza log dettagliati 5. Possibilità di esportare i log
Precondizioni	• Autenticazione con ruolo amministrativo • Log system correttamente configurato
Postcondizioni	• Log analizzati e classificati • Eventuali anomalie segnalate • Report di analisi dei log generato e archiviabile

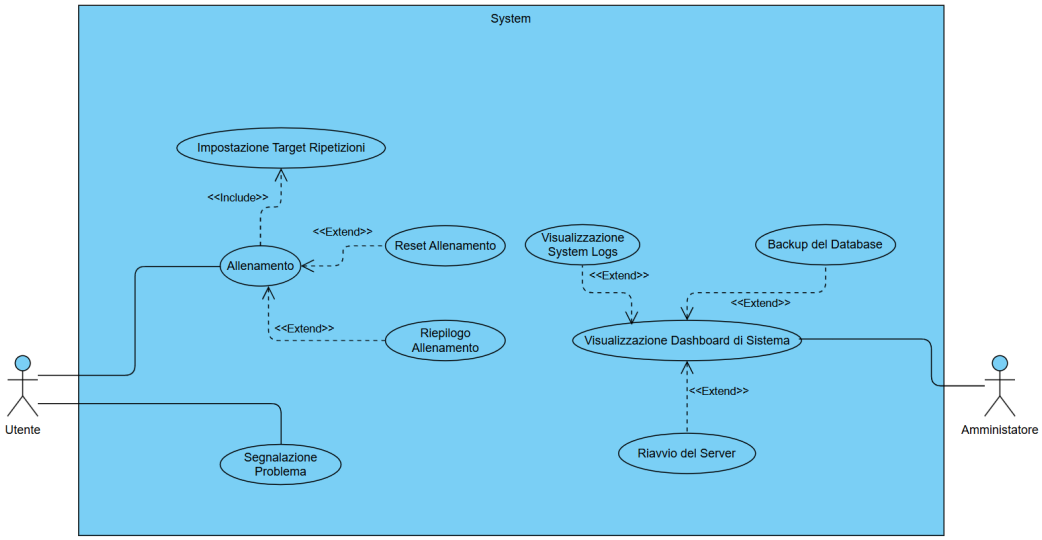


Figura 2.1: Diagramma dei casi d'uso

Capitolo 3

Sviluppo del Modello ML

3.1 MLOps

MLOps (Machine Learning Operations) è un insieme di pratiche che mirano a ottimizzare l'intero ciclo di vita dei modelli di machine learning.
L'obiettivo di MLOps è garantire un approccio efficiente e automatizzato per lo sviluppo, il deployment e la gestione dei modelli, assicurando che siano affidabili, scalabili e facilmente mantenibili, anche in ambienti dinamici.
Attraverso l'integrazione di questi flussi di lavoro, MLOps aiuta a velocizzare l'adozione dei modelli di machine learning, riducendo il rischio di errori e migliorando la qualità complessiva delle soluzioni.[McM23] [MK24]

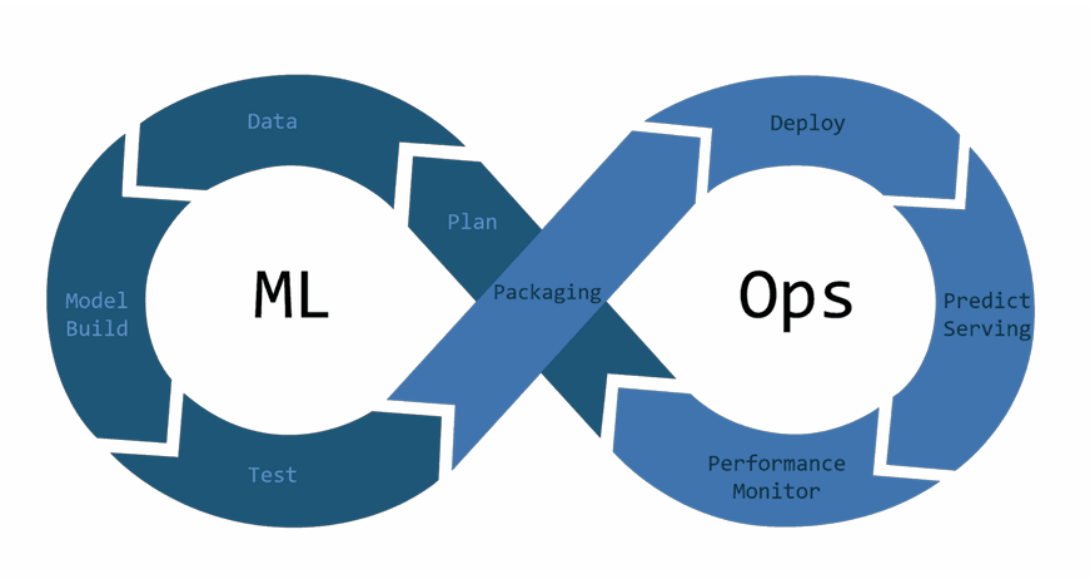


Figura 3.1: Fasi del ciclo di vita MLOps

1. **Plan**
   Definizione degli obiettivi del modello, metriche di successo (accuratezza, recall), requisiti dei dati, risorse disponibili e vincoli legali/etici. Collaborazione tra data scientist, ingegneri e stakeholder per allineare esigenze aziendali e tecnologiche.

2. **Data**
   Raccolta, pulizia (gestione valori mancanti, outlier) e trasformazione dei dati. Feature engineering, creazione di variabili significative attraverso tecniche come normalizzazione o encoding. Implementazione di pipeline automatizzate per garantire riproducibilità e aggiornamenti continui.

3. **Model Build**
   Selezione degli algoritmi (knn, alberi decisionali, svm) e ottimizzazione tramite cross-validation e tuning degli iperparametri. Versionamento del codice e del modello per tracciare modifiche e risultati sperimentali.

4. **Test**
   Valutazione delle prestazioni su dati di test mediante metriche predefinite (es. F1-score). Analisi di robustezza, bias e validazione finale per confermare l'idoneità alla produzione.

5. **Packaging**
   Preparazione del modello per il deployment: serializzazione in formati come ONNX o pickle, containerizzazione e definizione delle dipendenze software per garantire portabilità.

6. **Deploy**
   Distribuzione in ambienti cloud, on-premises o edge tramite API REST, microservizi o batch. Automazione con pipeline e configurazione di scalabilità e monitoraggio in tempo reale.

7. **Predict Serving**
   Esecuzione di inferenze in tempo reale. Ottimizzazione della latenza e logging delle predizioni per tracciabilità e debug.

8. **Performance Monitor**
   Monitoraggio continuo di accuratezza, latenza e drift dei dati. Rilevamento automatico di anomalie e ri-addestramento programmato su nuovi dataset per mantenere prestazioni ottimali.

3.2 Fasi di sviluppo

3.2.1 Plan

L'obiettivo principale è creare un sistema di machine learning capace di identificare con precisione differenti tipologie di movimenti, come squat, military press e push-up, attraverso l'analisi di input video. Abbiamo definito rigorose metriche di successo, stabilendo un target di accuratezza superiore al 90% per ogni categoria di esercizio. La strategia ha seguito le seguenti fasi:

1. Ricerca di un dataset estremamente variegato, che includa riprese di atleti con diverse corporature, in condizioni di illuminazione e ambienti di allenamento differenti, per garantire la robustezza dell'algoritmo.

2. In assenza di un dataset che rispecchi le nostre esigenze abbiamo tentato di generare il dataset utilizzando uno script Python che simulava i movimenti degli esercizi attraverso la creazione di dati sintetici.

3. Dopo aver analizzato i dati generati abbiamo riscontrato che non erano sufficientemente realistici e non rappresentavano accuratamente gli esercizi eseguiti nella realtà. Per questo motivo abbiamo registrato diverse persone mentre eseguivano esercizi specifici ("Push-up", "Military Press" e "Squat") e creato il dataset direttamente da questi video, permettendoci di avere un maggiore controllo sulla qualità e la precisione dei dati raccolti.

3.2.2 Data

Per il nostro applicativo abbiamo scelto di utilizzare **MediaPipe Pose Landmarker** [AI24] per estrarre informazioni dettagliate sulla postura e il movimento del corpo. Questa scelta ha determinato la necessità di costruire un dataset compatibile con il formato dei dati prodotti da questo strumento.

Struttura dei Dati di MediaPipe Pose Landmarker

MediaPipe Pose Landmarker fornisce una rappresentazione del corpo umano basata su **33 landmarks**, ognuno dei quali è descritto da tre coordinate spaziali: **x, y, z**. In dettaglio:

- **x e y** rappresentano le coordinate normalizzate rispetto all'immagine di input (valori tra 0 e 1), dove (0,0) corrisponde all'angolo in alto a sinistra e (1,1) all'angolo in basso a destra.

- **z** indica la profondità relativa rispetto alla posizione del bacino. Più il valore di z è negativo, più il punto si trova vicino alla telecamera.

Ogni frame di un esercizio è quindi rappresentato da una matrice di dimensione $\mathbf{33 \times 3}$, contenente le coordinate di ciascun landmark. Questo formato ci permette di catturare in modo preciso la postura dell'utente in ogni momento dell'esecuzione dell'esercizio.

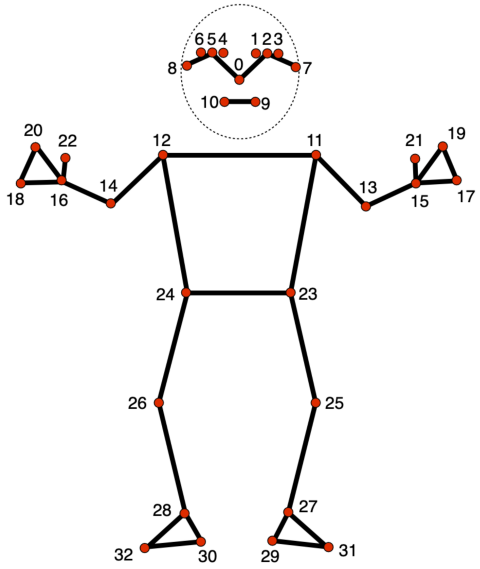


Figura 3.2: Mappa dei Landmarks di MediaPipe Pose

Indice dei Landmarks di MediaPipe Pose

ID	Landmark
0	Nose
1	Left eye (inner)
2	Left eye
3	Left eye (outer)
4	Right eye (inner)
5	Right eye
6	Right eye (outer)
7	Left ear
8	Right ear
9	Mouth (left)
10	Mouth (right)
11	Left shoulder
12	Right shoulder
13	Left elbow
14	Right elbow
15	Left wrist
16	Right wrist
17	Left pinky
18	Right pinky
19	Left index
20	Right index
21	Left thumb
22	Right thumb
23	Left hip
24	Right hip
25	Left knee
26	Right knee
27	Left ankle
28	Right ankle
29	Left heel
30	Right heel
31	Left foot index
32	Right foot index

Creazione del Dataset 1

Generazione di Dati Sintetici per il Riconoscimento di Pose

Descrizione del Sistema: Abbiamo implementato uno script python per la generazione di dati sintetici per simulare pose del corpo umano durante l'esecuzione di esercizi fisici, producendo 500 esempi per ciascuna delle tre categorie: squat, push-up e military press.

Struttura: Il sistema si basa su un modello a 33 punti di riferimento (landmarks), dove ogni punto è rappresentato da coordinate tridimensionali (x, y, z), risultando in una dimensionalità totale di $33_{landmarks} \times 3_{coordinate} = 99_{features}$ per esempio.

Modello Matematico: La generazione delle pose avviene attraverso trasformazioni trigonometriche secondo l'equazione:

$$P_{articolazione} = P_{base} + \begin{bmatrix} d \cdot \cos(\theta + \delta) \\ d \cdot \sin(\theta + \delta) \\ z_{offset} \end{bmatrix} \tag{3.1}$$

dove $P_{articolazione}$ è la posizione finale, P_{base} è il punto di riferimento, d è la distanza dall'articolazione precedente, θ è l'angolo base e δ è un offset casuale.

Parametrizzazione: Gli esercizi sono caratterizzati dai seguenti angoli principali:

- Squat (Classe 0): $\theta_{anca} = 60$, $\theta_{ginocchio} = 90$
- Push-up (Classe 1): $\theta_{spalla} = 45$, $\theta_{gomito} = 90$
- Military Press (Classe 2): $\theta_{spalla} = 10$, $\theta_{gomito} = 45$

Output: Il risultato finale è un file CSV contenente 99 colonne di features più una colonna di label, per un totale di 1500 righe di dati sintetici.

Creazione del Dataset 2

Raccolta dati sul movimento del corpo

Abbiamo creato uno script progettato per la raccolta di dati relativi al movimento del corpo durante l'esecuzione di diversi esercizi fisici, utilizzando la webcam del computer. Lo script ha due obiettivi principali: rilevare i movimenti del corpo in tempo reale e salvare i dati per l'utilizzo nell'addestramento di modelli di machine learning.
Il sistema utilizza due librerie principali: MediaPipe per il rilevamento della posa del corpo e OpenCV per la gestione del video. All'avvio, lo script presenta all'utente tre opzioni di esercizi:

- Squat
- Push-up

- Military Press

Una volta selezionato l'esercizio, il programma avvia la registrazione video attraverso la webcam. Durante la registrazione lo script:

1. Cattura continuamente fotogrammi dalla webcam.
2. Analizza ogni fotogramma per identificare i punti chiave del corpo.
3. Visualizza in tempo reale il video con una sovrapposizione che mostra i punti del corpo rilevati.
4. Raccoglie le coordinate tridimensionali di tutti i 33 punti chiave.

I dati vengono salvati in un file CSV specifico per ciascun esercizio (ad esempio *squat_data.csv* per gli squat). Ogni riga del file CSV contiene:

- 99 valori numerici (33 punti * 3 coordinate per punto).
- Una label numerica che identifica il tipo di esercizio.

La registrazione continua fino a quando l'utente non preme il tasto `q` per terminarla. Abbiamo poi proceduto ad unire tutti i file csv in un unico file che rappresenta il dataset utilizzato nelle fasi successive. Il dataset in questione presenta una distribuzione bilanciata delle classi (circa 1500 entry per ogni esercizio), non vi sono valori mancanti o outlier. Per la natura stessa del dataset non è necessario normalizzare i valori.

3.2.3 Model Build

Abbiamo creato uno script che implementa una pipeline di addestramento e valutazione di modelli di classificazione sul dataset. Il processo prevede le seguenti fasi:

Caricamento e Preparazione dei Dati

- Il dataset viene importato da un file CSV utilizzando `pandas`.
- I dati vengono mescolati casualmente con `shuffle()` per evitare dipendenze sequenziali.
- Le feature indipendenti ($\mathbf{X}$) e la variabile target ($\mathbf{y}$) vengono separate.
- Il dataset è suddiviso in **training set (80%)** e **test set (20%)** con suddivisione **stratificata**, preservando la distribuzione originale delle classi.

Preprocessing delle Feature (Opzionale, Commentato)

- È previsto il ridimensionamento dei dati con `StandardScaler()` per normalizzare le feature.
- È inclusa un'opzione per la riduzione della dimensionalità tramite PCA (`PCA(n_components=0.98)`) per mantenere il 98% della varianza originale.

Ottimizzazione e Addestramento dei Modelli Lo script utilizza `GridSearchCV` con **cross validation** (`cv=5`) per ottimizzare gli iperparametri di più modelli di classificazione:

- **K-Nearest Neighbors (KNN)**: ricerca ottimale di `n_neighbors`, `weights` e `metric` (euclidean, manhattan).
- **Support Vector Machine (SVM)**: ottimizzazione dei parametri `C`, `kernel` (lineare, rbf) e `gamma`.
- **Random Forest Classifier**: tuning di `n_estimators`, `max_depth` e `min_samples_split`.
- **Gradient Boosting Classifier**: ricerca dei migliori valori per `n_estimators`, `max_depth` e `min_samples_split`.

Valutazione delle Prestazioni Per ogni modello selezionato, viene calcolata la **matrice di confusione** e il **classification report** (accuratezza, precisione, richiamo, F1-score).

Persistenza dei Modelli I migliori modelli ottenuti da `GridSearchCV` vengono serializzati in formato `.pkl` tramite `pickle`, consentendone il riutilizzo senza necessità di riaddestramento.

Esecuzione Abbiamo eseguito questo script utilizzando il dataset creato precedentemente e i risultati ottenuti sono stati molto soddisfacenti. Dopo aver applicato il processo di ottimizzazione degli iperparametri tramite GridSearchCV e valutato le performance dei modelli, abbiamo ottenuto buone prestazioni in termini di accuratezza, precisione, recall e F1-score. I modelli selezionati hanno mostrato una solida capacità di generalizzazione sui dati di test, con una matrice di confusione che confermava la validità delle previsioni. In particolare i modelli **Random Forest** e **Support Vector Machine** hanno evidenziato una buona capacità di adattarsi alle caratteristiche del dataset, risultando tra i migliori nella valutazione finale.

3.2.4 Test

Abbiamo optato per testare il modello in maniera visiva, piuttosto che generare un dataset di test, per due principali ragioni. Innanzitutto testare in tempo reale permette di osservare direttamente come il modello reagisce a nuovi dati, senza dipendere da un set di test statico. Questo approccio ci consente di monitorare l'accuratezza delle predizioni in un contesto dinamico, come quello del riconoscimento di esercizi fisici, dove variabili come l'angolazione, la velocità del movimento o l'illuminazione possono cambiare continuamente.
Inoltre testare visivamente con la videocamera simula l'ambiente di utilizzo reale. Questo è particolarmente utile per valutare la robustezza del sistema poiché permette di raccogliere feedback immediato e osservare direttamente come il modello si comporta in condizioni variabili. Testando direttamente l'interfaccia possiamo garantire che il modello si integri correttamente in un sistema interattivo, come nel caso di un'applicazione per il monitoraggio in tempo reale degli esercizi, dove il riconoscimento e il feedback devono essere continui e istantanei.

Abbiamo prodotto uno script che fornisce una piattaforma per il test in tempo reale del modello di riconoscimento degli esercizi, utilizzando una webcam per acquisire sequenze video e inferire il tipo di esercizio eseguito tramite un modello di classificazione.
Possiamo dividere in punti il flusso di esecuzione:

1. **Caricamento del Modello Addestrato**: Il modello addestrato viene caricato dal file `.pkl` tramite la libreria `joblib`.

2. **Rilevamento dei Punti di Riferimento tramite MediaPipe**: L'elaborazione del flusso video viene gestita dalla libreria *MediaPipe* che rileva la posa del corpo umano.

3. **Preprocessing e Preparazione delle Feature**: I punti di riferimento estratti vengono riorganizzati in un array unidimensionale di feature (x, y, z per ciascun punto di riferimento) e successivamente ridimensionati in un formato compatibile con l'input del modello addestrato. Ogni fotogramma diventa un vettore di caratteristiche che rappresenta la configurazione spaziale del corpo durante l'esercizio.

4. **Predizione dell'Esercizio**: I vettori di feature vengono passati al modello tramite il metodo `predict()`. Il modello restituisce una classe numerica, che viene mappata a una delle etichette predefinite, ovvero *squat*, *push-up*, o *military press*. La previsione viene visualizzata nell'interfaccia utente.

5. **Interfaccia Grafica con Tkinter**: Per il display dei risultati, lo script utilizza la libreria *Tkinter*, che gestisce l'interfaccia grafica. Un *canvas* visualizza il flusso video in tempo reale, mentre l'etichetta dell'esercizio riconosciuto viene aggiornata dinamicamente.

6. **Aggiornamento del Video in Tempo Reale**: Il ciclo di acquisizione video viene gestito tramite la funzione `update_video()`, che legge il flusso video dalla webcam, elabora i fotogrammi e aggiorna l'interfaccia grafica ogni 33 millisecondi. Questo approccio permette un'interazione fluida e immediata con il sistema.

Questo script è stato utilizzato con tutti i modelli precedentemente addestrati per testarne le prestazioni in tempo reale e determinare quale di essi producesse i migliori risultati. Dopo aver eseguito diversi test visivi si è scelto il modello SVM come quello più performante, grazie alla sua capacità di riconoscere correttamente gli esercizi.

3.2.5 Packaging e Deployment

Preparazione del Modello per il Packaging E' fondamentale assicurarsi che il modello sia pronto per essere distribuito. Durante questa fase si sarebbe dovuto:

- **Salvare il modello addestrato**: Salvere il modello in un formato che potesse essere facilmente caricato e utilizzato successivamente. Una delle tecniche comuni per questo è l'uso di `pickle`. Ad esempio utilizzando `pickle.dump(model, 'path/to/model.pkl')` per salvare il modello.

- **Verificare le dipendenze**: Il modello non è solo un file che contiene i parametri appresi, ma può anche dipendere da librerie esterne specifiche. Bisogna dunque creare un file `requirements.txt` che elenchi tutte le librerie e le versioni necessarie per il funzionamento del modello (ad esempio, `scikit-learn`, `numpy`, `flask`, ecc.).

- **Gestire la versione del modello**: Prendere in considerazione la gestione delle versioni del modello, per permettere aggiornamenti e miglioramenti in futuro senza compromettere le versioni precedenti. Per esempio, usando un sistema di controllo versione per i modelli (come `MLflow`) o semplicemente numerare i file dei modelli.

Creazione di un'API per l'Accesso al Modello Una volta che il modello è stato salvato e preparato, è possibile creare un'interfaccia per permettere alle applicazioni esterne di interagire con esso. Una delle soluzioni più comuni è quella di implementare una **API REST**.

- **Flask**:Scegliendo un framework come `Flask` per costruire un'API che potesse ricevere richieste HTTP, passare i dati al modello e restituire le previsioni.

Container per il Modello (Docker) Dopo aver creato l'API, Avremmo dovuto "impacchettare" tutto in un container per garantire che il modello potesse essere eseguito in modo consistente in qualsiasi ambiente. La creazione di un **container Docker** avrebbe avuto i seguenti vantaggi:

- **Isolamento e Portabilità**: Avremmo creato un file `Dockerfile` per specificare le dipendenze, l'ambiente di esecuzione e il comando per eseguire l'app. Questo garantirebbe il funzionamento del modello in modo coerente su qualsiasi macchina senza problemi di configurazione.
- **Configurazione dell'Ambiente di Esecuzione**: All'interno del container Docker, Avremmo incluso tutte le librerie necessarie per l'esecuzione del modello e dell'API (come `flask`, `scikit-learn`, `numpy` e `pandas`).
- **Scalabilità**: Se il modello fosse stato destinato a gestire molte richieste contemporaneamente, avremmo configurato il container Docker in modo che fosse scalabile orizzontalmente, utilizzando, ad esempio, un bilanciatore di carico per gestire più istanze del container.

Deployment su Server/Cloud Una volta pronti e testati il modello e l'API si può procedere con il deployment. Le opzioni per il deployment includono:

- **Server on-premise**: Nel caso di un'infrastruttura locale, avremmo dovuto configurare un server web (ad esempio, `Nginx` o `Apache`) per gestire le richieste e indirizzarle all'API Flask.
- **Cloud (AWS, Azure, GCP)**: Nel caso di un'infrastruttura su cloud, avremmo scelto una piattaforma come `AWS EC2`, `Google Cloud Platform` o `Azure` per ospitare il container Docker e l'API.
- **Continuous Integration / Continuous Deployment (CI/CD)**: Avremmo configurato un processo di CI/CD per automatizzare il testing e il deployment del modello in produzione, ad esempio utilizzando strumenti come `GitHub Actions`.

3.2.6 Predict Serving e Performance Monitor

Predict Serving La fase di Predict Serving riguarda l'esecuzione del modello in produzione, ovvero come il modello fornisce le previsioni per i nuovi dati che vengono inviati.

- **Gestione delle richieste in tempo reale**: bisogna configurare l'API per ricevere richieste in tempo reale e ricevere la previsione corrispondente ("squat", "push up" o "military press").
- **Segnalazione degli errori nelle previsioni**: Un elemento importante è permettere agli utenti di segnalare errori nelle previsioni. Se un utente

ritiene che il modello abbia classificato erroneamente un esercizio potrebbe segnalare l'errore attraverso l'API. In questo modo i dati relativi all'errore potrebbero essere raccolti per migliorare il modello nel futuro.

Performance Monitoring Il monitoraggio delle performance è fondamentale per garantire che il modello continui a funzionare correttamente in produzione, che mantenga alte prestazioni e che eventuali anomalie vengano identificate e corrette tempestivamente. Durante questa fase, avremmo dovuto implementare i seguenti processi:

- **Monitoraggio delle performance del modello**: sarebbe necessario monitorare regolarmente la qualità delle previsioni del modello, tracciando metriche come l'accuratezza, la precisione, la recall e l'F1-score su un insieme di dati di test in produzione.
- **Monitoraggio della latenza dell'API**: È fondamentale monitorare anche il tempo di risposta dell'API per assicurarsi che non ci siano ritardi significativi, soprattutto in scenari di produzione ad alta richiesta.
- **Verifica dell'accuratezza nel tempo**: avremmo configurato una strategia per monitorare l'accuratezza del modello nel tempo, per identificare eventuali degradazioni delle performance, come il cosiddetto "model drift". Questo può essere fatto confrontando le previsioni con i dati reali nel tempo e verificando se il modello si adatta a nuovi pattern.
- **Gestione delle anomalie**: Se il monitoraggio del modello o dell'API rileva performance insoddisfacenti (ad esempio un abbassamento improvviso della precisione o della latenza), il sistema dovrebbe inviare notifiche agli amministratori del sistema o agli sviluppatori per risolvere tempestivamente il problema.

Capitolo 4

Integrazione con il Sistema

Questo capitolo descrive l'integrazione del modello di ML con i vari componenti del sistema, includendo la gestione dell'autenticazione e autorizzazione tramite Keycloak, l'elaborazione del flusso video con OpenCV e MediaPipe, e l'uso di algoritmi per il conteggio delle ripetizioni. L'obiettivo è fornire un ambiente interattivo, sicuro ed efficiente, garantendo una gestione fluida dell'accesso utente e un monitoraggio accurato delle attività fisiche.

4.1 Autenticazione e Autorizzazione con Keycloak

Nell'applicazione abbiamo implementato un meccanismo di autenticazione e autorizzazione con RBAC (Role Based Access Criteria) basato su Keycloak , un sistema di gestione dell'identità e dell'accesso (IAM) open-source che aderisce al protocollo OAuth 2.0. [Key25]

4.1.1 Fasi del Processo di Autenticazione

Il processo di autenticazione si articola nelle seguenti fasi:

1. **Registrazione del Client OAuth:** L'applicazione si registra presso Keycloak come "client", fornendo identificativi univoci (client_id e client_secret) e definendo gli endpoint per l'autorizzazione e la gestione dei token.

2. **Login Utente:** L'utente quando accede a risorse protette viene reindirizzato alla pagina di login di Keycloak, dove può autenticarsi tramite le proprie credenziali. Keycloak supporta diverse modalità di autenticazione, inclusi username/password e autenticazione a due fattori.

3. **Autorizzazione e Gestione dei Token:** In seguito all'autenticazione Keycloak rilascia un token JWT (JSON Web Token) contenente informazioni sull'utente, inclusi i suoi ruoli. Questo token viene trasmesso all'applicazione che lo archivia in modo sicuro nella sessione utente. Il token è utilizzato per autenticare le successive richieste dell'utente.

4. **Controllo dei Ruoli e Autorizzazione:** L'applicazione verifica i ruoli dell'utente estratti dal token JWT per determinare le risorse a cui può accedere. L'accesso alle funzionalità è quindi regolato in base ai ruoli, ad esempio, solo gli utenti con il ruolo "adminRole" possono accedere alla dashboard di amministrazione, successivamente ad un login tramite 2FA, con l'ausilio di un OTP.

5. **Logout Utente:** Quando l'utente effettua il logout l'applicazione comunica con Keycloak per invalidare il token JWT, impedendone l'uso futuro. La sessione utente nell'applicazione viene terminata.

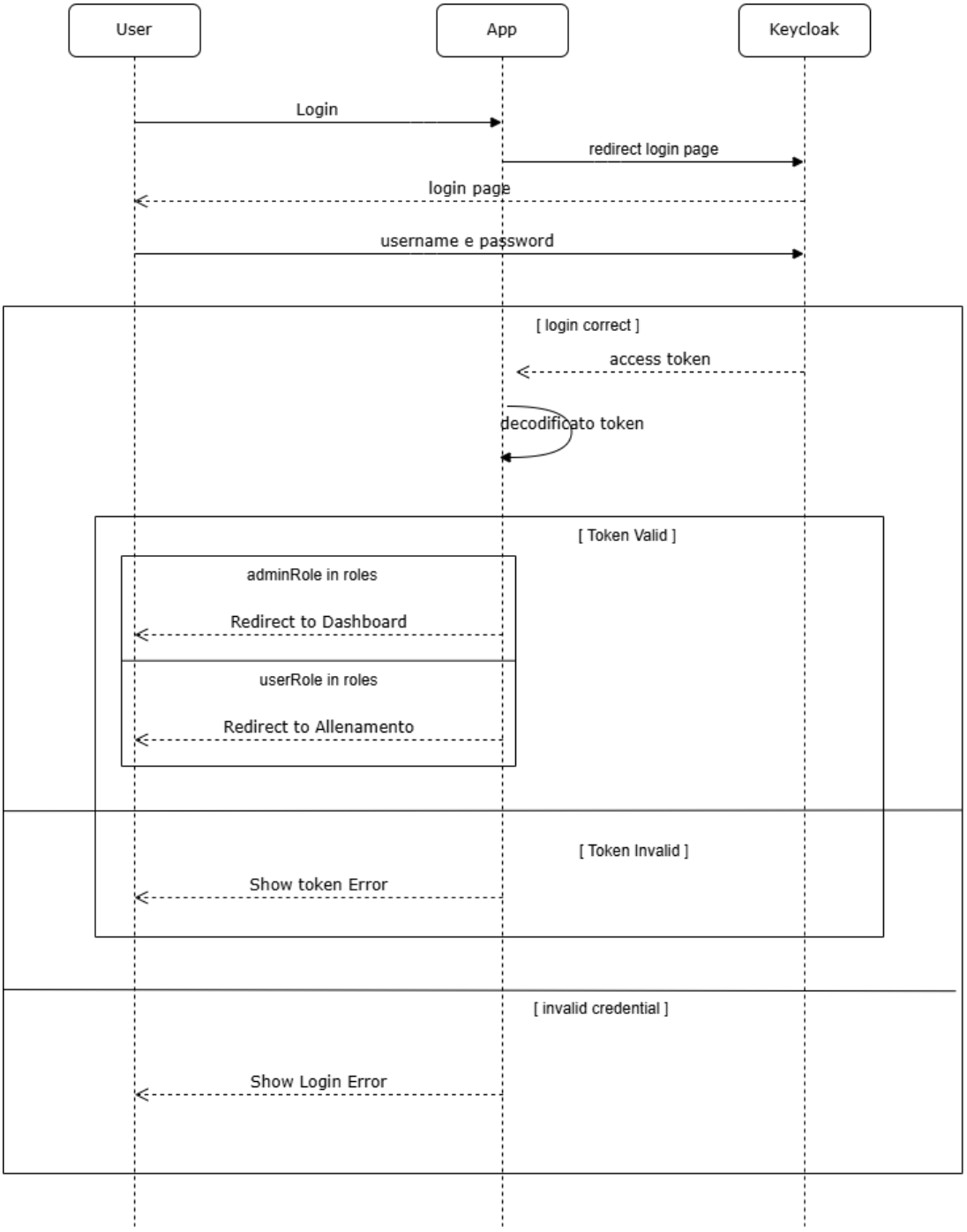


Figura 4.1: Diagramma di flusso dell'autenticazione

4.2 Gestione della Webcam e Flusso Video

L'applicazione implementa un sistema sofisticato per l'acquisizione, l'elaborazione e la trasmissione del flusso video proveniente dalla webcam dell'utente, offrendo un'esperienza interattiva e coinvolgente. Di seguito vengono descritte le fasi principali di questo processo.

Acquisizione del Flusso Video L'acquisizione del flusso video avviene tramite la libreria OpenCV (cv2). OpenCV fornisce gli strumenti necessari per interagire con le periferiche video, come webcam e videocamere. La webcam viene identificata attraverso un indice numerico, tipicamente 0 per la webcam predefinita del sistema.

Elaborazione dei Frame Ciascun fotogramma (frame) del flusso video viene elaborato individualmente. Inizialmente il formato colore viene convertito da BGR (utilizzato da OpenCV) a RGB. Questa conversione è essenziale poiché il modello di riconoscimento delle pose MediaPipe, impiegato successivamente, richiede immagini in formato RGB.

Successivamente viene utilizzata la libreria MediaPipe che ha il compito di identificare i punti chiave del corpo umano (landmarks) presenti in ogni frame.

Disegno delle Pose Una volta identificati i landmarks una funzione dedicata provvede a sovrapporli al frame originale. Ciò consente la visualizzazione in tempo reale del tracciamento dei movimenti dell'utente. La sovrapposizione dei landmarks non è solo un elemento grafico, ma anche uno strumento cruciale per fornire un feedback visivo all'utente durante l'allenamento.

Trasmissione del Flusso Video L'ultima fase prevede la codifica dei frame elaborati in formato JPEG. Questo formato di compressione, ampiamente supportato dai browser web, permette di ridurre le dimensioni dei frame senza compromettere eccessivamente la qualità dell'immagine. I frame codificati vengono quindi trasmessi al browser dell'utente attraverso Flask, un framework web in Python. Flask gestisce la creazione di un flusso video (video stream) fluido. Tramite un formato che consente l'invio continuo dei frame, garantendo un'esperienza video fluida nel browser.

4.3 Riconoscimento dell'esercizio e conteggio delle ripetizioni

Il sistema di riconoscimento degli esercizi implementato si basa sull'integrazione tra l'analisi della postura, resa possibile da MediaPipe, e il modello di *machine learning* pre-addestrato. Questa combinazione permette di identificare l'esercizio svolto dall'utente e di contarne le ripetizioni in modo automatico.

4.3.1 Estrazione delle caratteristiche della postura

Questa fase iniziale si concentra sull'elaborazione dei dati relativi alla postura dell'utente catturati attraverso la webcam.

1. **Rilevamento dei punti chiave:** MediaPipe analizza in tempo reale i frame video acquisiti dalla webcam individuando i punti chiave.

2. **Normalizzazione e vettorizzazione:** Le coordinate spaziali dei punti chiave rilevati vengono elaborate attraverso un processo di normalizzazione. Successivamente le coordinate normalizzate vengono organizzate in un vettore numerico.

4.3.2 Classificazione dell'esercizio

Una volta ottenuto il vettore di caratteristiche posturali si passa alla fase di identificazione dell'esercizio.

1. **Modello di Machine Learning:** Il vettore di caratteristiche viene fornito come input al modello di Support Vector Machine (SVM).

2. **Riconoscimento:** Il modello SVM analizza il vettore di input e classifica la postura corrente predicendo a quale esercizio corrisponde. Il risultato di questa classificazione è un'etichetta che identifica inequivocabilmente l'esercizio in corso di esecuzione.

4.3.3 Conteggio delle ripetizioni

Per ciascun esercizio riconosciuto viene attivata una logica di conteggio dedicata, progettata per monitorare i movimenti specifici che ne caratterizzano l'esecuzione.

Logica specifica per esercizio: La logica di conteggio è adattata alle peculiarità di ogni esercizio. Sfrutta le informazioni relative ai punti chiave per seguire in dettaglio i movimenti delle articolazioni coinvolte.

Angoli e soglie: Un elemento chiave di questa logica è il calcolo degli angoli tra le articolazioni. Il sistema confronta questi angoli con delle soglie predefinite. Il superamento di una soglia indica il completamento di una fase significativa del movimento, come la discesa nello squat o la flessione del gomito nel push-up.

Esempio Squat: Prendiamo ad esempio lo squat. Il sistema monitora costantemente l'angolo formato da anca, ginocchio e caviglia. Quando questo angolo diminuisce (fase di discesa) e successivamente aumenta (fase di risalita), la logica di conteggio registra il completamento di una ripetizione.

4.3.4 Gestione del riposo

Il sistema prevede anche una fase di riposo, essenziale per un allenamento efficace.

1. **Obiettivo raggiunto:** Una volta che l'utente raggiunge il numero di ripetizioni desiderato per un determinato esercizio (un valore che può essere impostato in base alle proprie preferenze), il sistema lo segnala chiaramente.
2. **Timer di riposo:** Si avvia automaticamente un timer di riposo. Durante questo periodo non vengono conteggiate eventuali ripetizioni.
3. **Ripresa dell'allenamento:** Allo scadere del timer di riposo il sistema si ripristina ed è pronto per monitorare l'esecuzione di una nuova serie di ripetizioni consentendo all'utente di proseguire l'allenamento.

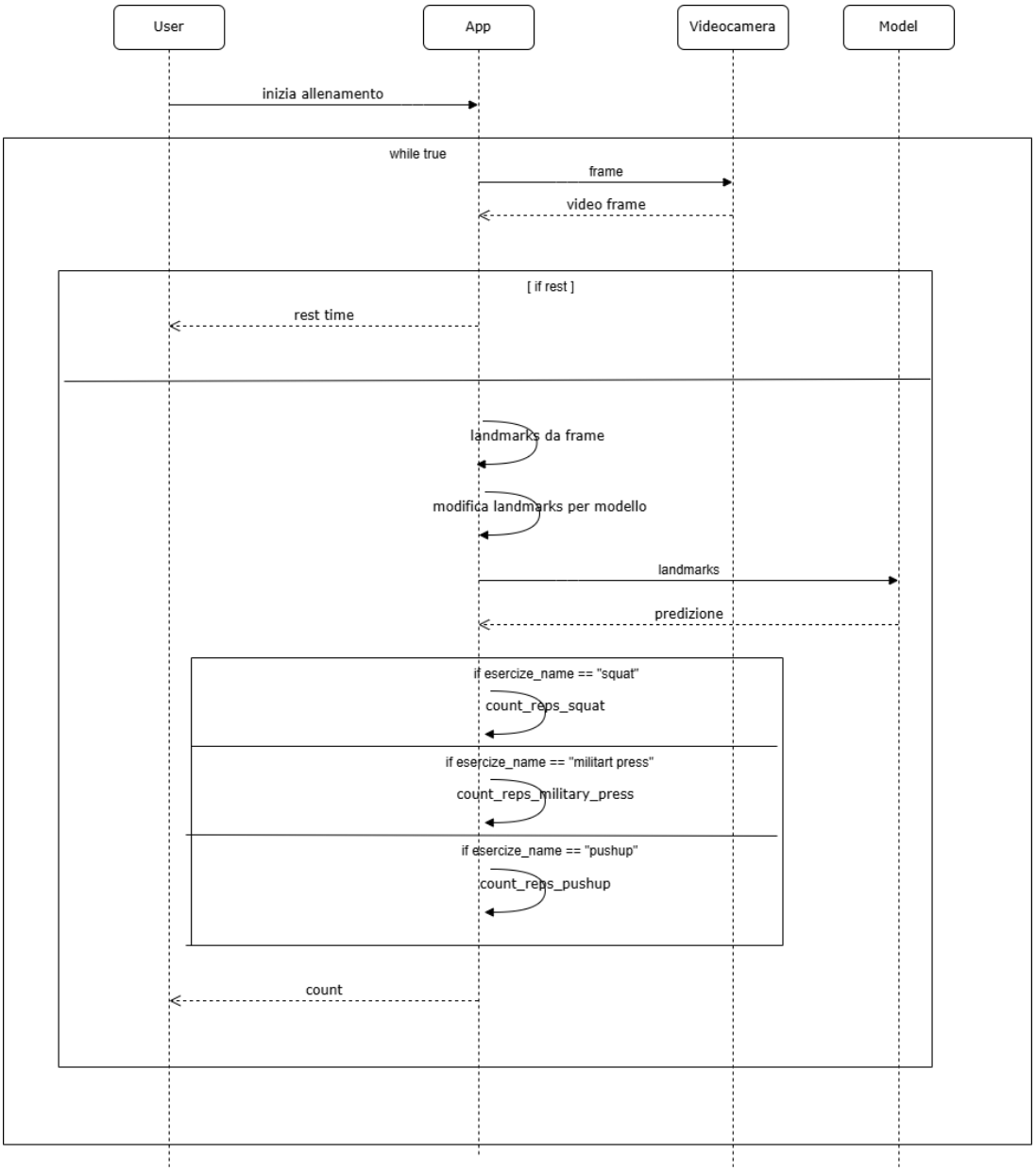


Figura 4.2: Diagramma di flusso dell'allenamento

4.4 Sintesi vocale e feedback audio

Il sistema sfrutta la libreria `pyttsx3` per arricchire l'esperienza utente con istruzioni vocali chiare e feedback audio tempestivi durante l'allenamento. Questa integrazione permette di creare un'interazione più coinvolgente e intuitiva guidando l'utente attraverso le diverse fasi dell'esercizio.

Feedback per ogni ripetizione Al completamento di ogni ripetizione il sistema fornisce un feedback immediato comunicando vocalmente il numero di ripetizioni correnti.

Comunicazioni chiare durante le fasi di riposo Le fasi di riposo, cruciali per il recupero muscolare e la preparazione al set successivo, sono accompagnate da messaggi vocali specifici. Al termine di un set il sistema avvisa l'utente dell'inizio del periodo di riposo fornendo indicazioni chiare sulla durata. Quando il riposo è terminato un messaggio vocale comunica che è possibile riprendere l'allenamento.

Esecuzione in thread La sintesi vocale viene eseguita in un thread separato per garantire che la generazione del feedback vocale non influisca negativamente sulle prestazioni complessive del sistema. Questa implementazione permette di mantenere fluidità nell'elaborazione delle immagini dalla webcam e nell'analisi della postura in tempo reale senza interruzioni o rallentamenti causati dalla generazione vocale.

4.5 Logging e gestione degli errori

Gli errori e gli eventi significativi vengono registrati in un file di log dedicato, `exercise_errors.log`, tramite il modulo python *logging*. Questo approccio centralizzato offre numerosi vantaggi:

- **Diagnosi rapida dei problemi:** Il file di log fornisce una cronologia dettagliata degli eventi, facilitando l'individuazione e la correzione dei problemi.
- **Monitoraggio proattivo dell'applicazione:** Il logging continuo consente di monitorare l'applicazione e identificare potenziali problemi.
- **Analisi delle tendenze e miglioramento continuo:** I log possono essere analizzati per identificare pattern di errore e migliorare la stabilità dell'applicazione nel tempo.

4.5.1 Gestione degli errori su più livelli

L'applicazione implementa una gestione degli errori su due livelli:

- **Lato server:** Gli errori che si verificano sul server vengono intercettati e registrati nel file di log. Vengono restituite risposte di errore appropriate al client, garantendo che l'utente riceva un feedback chiaro in caso di problemi.
- **Lato client:** L'applicazione include un endpoint dedicato (`/log_error`) che consente al client di segnalare errori al server. Questo meccanismo è particolarmente utile per tracciare errori che si verificano nell'interazione dell'utente con l'applicazione.

4.6 Avvio della demo

Abbiamo automatizzato l'avvio del server Keycloak e l'applicazione Python, assicurandoci che entrambi vengano avviati correttamente prima di aprire il browser alla pagina desiderata. Dopo aver avviato il server Keycloak in una nuova finestra del terminale, si attende 20 secondi per garantire che il servizio sia completamente operativo. Successivamente si avvia l'applicazione Python lasciando ulteriore tempo per la sua inizializzazione prima di procedere. Infine si apre automaticamente il browser web all'indirizzo *http://localhost:5000*, permettendo all'utente di accedere rapidamente all'applicazione senza dover eseguire manualmente ogni passaggio.

Capitolo 5

Monitoraggio continuo & Retraining

Il monitoraggio continuo è fondamentale per garantire l'affidabilità e le prestazioni ottimali di sistemi basati su intelligenza artificiale (AI). Attraverso il logging centralizzato e un processo iterativo di segnalazione e correzione degli errori è possibile identificare anomalie, migliorare il modello e adeguarlo alle variazioni dei dati e alle esigenze degli utenti. Questo approccio si integra perfettamente nel ciclo di vita MLOps, supportando il testing, la validazione e il retraining automatico del modello.

5.1 Logging e Monitoraggio Operativo

Il logging rappresenta lo strumento principale per raccogliere e analizzare i dati operativi del sistema. I vantaggi principali di un logging centralizzato sono:

- **Monitoraggio in tempo reale:** Controllo continuo dello stato del sistema e delle performance.
- **Raccolta di metriche:** Registrazione di indicatori di performance e dati utili per la diagnosi.
- **Identificazione di anomalie:** Rilevamento tempestivo di comportamenti inaspettati e deviazioni rispetto ai dati di addestramento (drift detection).
- **Gestione degli errori:** Categorizzazione e analisi degli errori per intervenire.

5.1.1 Integrazione con MLOps

All'interno del framework MLOps il logging supporta:

1. **Validazione e testing continuo:** Ogni predizione segnalata viene registrata e confrontata con i valori attesi.

2. **Drift detection:** Analisi dei log per individuare eventuali deviazioni nei dati di input rispetto a quelli di training.

3. **Automazione degli aggiornamenti:** Attivazione di pipeline di retraining quando il modello mostra un degradamento nelle prestazioni.

5.2 Obiettivo del Monitoraggio Continuo

L'obiettivo principale è migliorare l'accuratezza del sistema attraverso l'identificazione e la correzione delle predizioni errate. Ciò si ottiene implementando un meccanismo di feedback che coinvolga direttamente gli utenti e permetta di:

- Segnalare errori di predizione e fornire la risposta corretta.

- Utilizzare i dati raccolti per il riaddestramento e il continuo miglioramento del modello.

1. Segnalazione da parte dell'utente Attraverso un'interfaccia intuitiva l'utente può segnalare errori di predizione. Durante l'interazione con il modello l'utente riceve una risposta e, in caso di errore (ad esempio, se l'esercizio non è riconosciuto), può indicare la risposta corretta.

2. Raccolta e Analisi delle Segnalazioni Le segnalazioni vengono centralizzate in un database sicuro. Un team analizza i feedback per:

- Identificare pattern ricorrenti.

- Valutare l'entità degli errori, distinguendo tra casi isolati e problemi sistematici.

- Verificare la coerenza e la fondatezza dei feedback

3. Aggiornamento e Riaddestramento del Modello Le segnalazioni validate alimentano il ciclo di miglioramento:

- I dati raccolti vengono utilizzati per il riaddestramento del modello, permettendogli di correggere errori e migliorare la precisione.

- Il modello aggiornato è sottoposto a test.

4. Rilascio e Monitoraggio Continuo Il modello aggiornato viene rilasciato in produzione e monitorato costantemente. Questo ciclo di feedback continuo consente:

- L'identificazione tempestiva di eventuali errori residui.
- Un costante aggiornamento e miglioramento del modello in risposta ai nuovi dati e alle esigenze degli utenti.

5.3 Benefici dell'Approccio

L'adozione di un processo integrato di logging, monitoraggio e feedback presenta numerosi vantaggi:

- **Miglioramento continuo:** Il modello si adatta dinamicamente alle variazioni dei dati e alle esigenze degli utenti.
- **Affidabilità:** Un processo strutturato di validazione e monitoraggio garantisce interventi mirati e efficaci.
- **Coinvolgimento degli utenti:** Il feedback diretto aumenta la fiducia e consente di affrontare problematiche specifiche in contesti reali.

Capitolo 6

Conclusioni

Riassumiamo quindi le principali valutazioni sul progetto, evidenzia le criticità riscontrate e delinea le possibili direzioni di sviluppo futuro, ponendo particolare attenzione agli aspetti legati alla precisione del riconoscimento, all'espansione delle funzionalità e all'integrazione con altre tecnologie e servizi.

6.1 Valutazioni e Considerazioni sui Risultati Ottenuti

Il sistema sviluppato è in grado di riconoscere correttamente gli esercizi eseguiti dagli utenti in condizioni ideali, garantendo un feedback immediato durante l'allenamento. Tuttavia se da un lato il riconoscimento dei movimenti risulta soddisfacente, dall'altro il conteggio delle ripetizioni presenta ancora margini di miglioramento. In particolare la precisione complessiva è fortemente influenzata da fattori esterni e dalla variabilità individuale, come verrà dettagliato nella sezione seguente.

6.2 Limiti e Sviluppi Futuri

Nonostante i risultati ottenuti siano promettenti, il progetto presenta alcune limitazioni che evidenziano la necessità di ulteriori sviluppi. Le principali aree di intervento sono:

6.2.1 Precisione del Riconoscimento e Variabilità Individuale

Limiti e Sfide

Il riconoscimento degli esercizi tramite rilevamento della postura è sensibile a diversi fattori:

1. **Qualità delle Immagini:** Immagini sfocate o di bassa risoluzione possono compromettere l'accuratezza del rilevamento.

2. **Condizioni di Illuminazione:** Ambienti con scarsa o non uniforme illuminazione rendono difficile l'estrazione corretta delle features.

3. **Esecuzione del Movimento:** Movimenti parziali o eseguiti in maniera non standard possono portare a errori di classificazione e mancate rilevazioni.

La variabilità individuale (diverse conformazioni fisiche, stili di movimento e livelli di esperienza) comporta che un modello addestrato su un set di dati limitato potrebbe non generalizzare perfettamente per tutti gli utenti, portando a false rilevazioni o errori nel conteggio delle ripetizioni.

Strategie e Soluzioni Proposte

Per migliorare la precisione e l'adattabilità del sistema si propongono le seguenti azioni:

1. **Ottimizzazione della Qualità Visiva:** Ottimizzazione delle impostazioni di acquisizione e applicazione di algoritmi di pre-processing (riduzione del rumore, miglioramento del contrasto) per garantire immagini di migliore qualità.

2. **Dataset Diversificato:** Ampliare il training set con dati raccolti in condizioni variegate e da utenti con differenti caratteristiche fisiche e livelli di esperienza, per migliorare la capacità di generalizzazione del modello.

3. **Calibrazione Personalizzata:** Sviluppare un sistema di calibrazione iniziale che adatti il riconoscimento alle peculiarità motorie e fisiche di ciascun utente, migliorando la precisione del feedback.

6.2.2 Espansione del Catalogo di Esercizi Riconosciuti

L'implementazione attuale consente solo un numero limitato di esercizi, non permettendone così l'uso in un programma di allenamento completo. Per ampliare l'applicazione si rendono necessarie le seguenti azioni:

1. **Raccolta e Ampliamento del Dataset:** Acquisire dati per una gamma più ampia di esercizi, includendo varianti (es. squat standard e squat bulgaro) e differenti livelli di difficoltà, considerando angolazioni diverse, condizioni di illuminazione e stili esecutivi.

2. **Validazione Approfondita:** Condurre test rigorosi per verificare l'accuratezza e l'affidabilità del riconoscimento prima di integrare nuovi esercizi nella versione finale dell'app.

6.2.3 Feedback sulla Correttezza dell'Esercizio

Un importante sviluppo futuro consiste nell'estendere le funzionalità dell'app non solo al riconoscimento e conteggio delle ripetizioni ma anche alla valutazione della correttezza dell'esecuzione degli esercizi. Le possibili strade includono:

1. **Analisi dei Landmark:** Utilizzo dei punti chiave (landmark) per calcolare angoli e distanze tra le parti del corpo, confrontando i valori con range di riferimento per determinare eventuali deviazioni dallo standard.

2. **Confronto con Esempi di Esecuzione Corretta:** Implementare un sistema di confronto che metta a confronto il movimento dell'utente con esempi predefiniti, fornendo suggerimenti e correzioni.

3. **Integrazione di Dati da Sensori:** L'impiego di sensori inerziali (accelerometri, giroscopi) da dispositivi wearable può fornire ulteriori dati per una valutazione più accurata del movimento.

6.2.4 Integrazione con Dispositivi Wearable e App di Salute

Una direzione di sviluppo consiste nell'integrare il sistema con dispositivi wearable (smartwatch, fitness tracker) e altre app dedicate al benessere, al fine di arricchire i dati a disposizione e migliorare l'esperienza utente. I vantaggi di tale integrazione includono:

1. **Migliore Precisione del Riconoscimento:** Combinando dati visivi e sensoriali il sistema potrà analizzare i movimenti in maniera più accurata.

2. **Riconoscimento di Esercizi Complessi:** Sensori aggiuntivi faciliteranno l'identificazione di movimenti articolati e l'analisi della dinamica corporea.

3. **Monitoraggio Globale della Salute:** L'integrazione con app di salute permetterà di centralizzare dati relativi all'attività fisica, alimentazione e sonno, offrendo una visione complessiva del benessere dell'utente.

6.2.5 Gamification e Personalizzazione dell'Esperienza Utente

L'introduzione di elementi di gamification rappresenta un'ulteriore opportunità per rendere l'app più coinvolgente e motivante. Le possibili implementazioni includono:

1. **Sfide Personalizzate:** Proporre obiettivi e sfide in linea con il livello e le preferenze dell'utente.

2. **Ricompense Virtuali:** Assegnare badge, punti o premi al completamento di obiettivi e sfide, incentivando l'utente a progredire.

3. **Classifiche e Condivisione Sociale:** Creare una community in cui gli utenti possano confrontarsi e condividere i propri risultati, stimolando una competizione sana e una maggiore fidelizzazione.

Questa strategia mira non solo a incrementare il coinvolgimento, ma anche a personalizzare l'esperienza di allenamento, rendendo l'app un supporto costante e motivante nel percorso verso uno stile di vita sano.

6.3 Considerazioni Finali

Il progetto ha dimostrato la fattibilità di un sistema di riconoscimento degli esercizi basato su intelligenza artificiale, offrendo già oggi un valido supporto per l'allenamento. Tuttavia, per trasformarlo in uno strumento completo e versatile, saranno necessari ulteriori miglioramenti in termini di precisione, espansione del catalogo degli esercizi, feedback sulla corretta esecuzione e integrazione con altri dispositivi e piattaforme. L'evoluzione futura del sistema, incentrata su tecniche avanzate di machine learning, una raccolta dati sempre più ampia e diversificata, e l'integrazione con dispositivi wearable e app di salute, potrà offrire agli utenti un'esperienza personalizzata, sicura e motivante, in grado di rispondere efficacemente alle esigenze di un pubblico sempre più attento al benessere fisico.

Bibliografia

[AI24] Google AI. *MediaPipe Pose Landmarker*. Accessed: 2024-02-01. 2024. URL: https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker.

[Key25] Keycloak Community. *Keycloak Documentation*. Accessed: 2025-02-03. 2025. URL: https://www.keycloak.org/documentation.

[McM23] Andrew P. McMahon. *Machine Learning Engineering with Python – Second Edition: Manage the production life cycle of machine learning models using MLOps with practical examples*. Packt Publishing, 2023. ISBN: 978-1837631964.

[MK24] David R. Martinez e Bruke M. Kifle. *Artificial Intelligence: A Systems Approach from Architecture Principles to Deployment*. The MIT Press, 2024. ISBN: 9780262048989.

[Pie25] Roberto Pietrantuono. *Slides from lessons and exercises*. Available on the Microsoft Teams platform and the University teaching website. Accessed: 2025-02-03. 2025. URL: http://www.docenti.unina.it/roberto.pietrantuono.