WorkoutNow

Progetto presentato da:

Elena Volpe 1000001186

Carmelo Angelo Federico Ragusa 1000056679

Descrizione dell'applicazione

L'applicazione da noi presentata ha lo scopo di fornire ai clienti una piattaforma sulla quale potersi iscrivere per poter creare, in base alle proprie preferenze, una propria scheda da palestra personalizzata. L'utente può infatti visualizzare nella Home una serie di esercizi con relativa descrizione e muscoli allenati e tramite interfaccia grafica, aggiungerli alla propria scheda. Il sistema nel suo Account, oltre a mostrargli le sue generalità, gli proporrà in base ai muscoli che l'utente ha scelto in fase di iscrizione, esercizi adatti alle sue esigenze, esercizi in base a gruppi muscolari che esso sta trascurando ed inoltre gli mostrerà un grafico a torta rappresentante le percentuali con cui sta allenando i suoi muscoli. Se l'accesso viene effettuato con l'account admin, esso avrà la possibilità di inserire o eliminare a suo piacimento gli esercizi dalla lista degli esercizi proposti.

Architettura dell'applicazione

L'applicazione è progettata come un Client-Server le cui componenti comunicano quindi tra loro tramite richieste Http. In C# è stato implementato il lato client e python è il suo server, python inoltre è anche client di go, il quale si interfaccia col database mysql.

Suddivisione del lavoro

Il modulo C# è stato curato da Volpe, il modulo Go è stato curato da Ragusa, per quanto riguarda Python i moduli utils e get_muscle_stats sono stati curati da Ragusa, mentre i restanti user_manager.py, exercise_manager.py e server.py sono stati curati da entrambi.

Modulo C#

In C# è stata implementata l'interfaccia grafica tramite Windows Forms. Con essa possono agire sia classici utenti, i quali possono registrarsi, accedere e creare la propria scheda da palestra, sia l'admin, il quale invece ha la possibilità di eliminare o aggiungere nuovi esercizi al database.

In questo modulo viene in particolare utilizzata la libreria System.Net per riuscire ad effettuare richieste http al server python, in particolare noi usiamo la sua classe WebClient. Utilizziamo inoltre la libreria System.Newtonsoft.Json che ci permette di serializzare e deserializzare gli oggetti che rispettivamente inviamo e riceviamo in una richiesta http POST.

Di seguito presentiamo tutti i file C#, il loro scopo e il relativo funzionamento.

- *Form Login*
  Tramite tale form il cliente, inserendo e-mail e password, può accedere al sistema, sono presenti stringhe di errore nel caso in cui esso sbagli ad inserire qualche parametro.
- *Form Registrazione*

Tramite tale form il cliente si può registrare al sistema, sono anche qui presenti stringhe di errore. In tale form è presente in particolare la funzione IsValidEmail, la quale fa uso di un pattern Regex (utilizzabile grazie alla libreria System.Text.RegularExpressions) per verificare che il formato dell'e-mail inserita sia valido.

- *Form Account*
  Qui il cliente può visionare le proprie informazioni, un grafico a torta rappresentante delle statistiche sui muscoli allenati, una lista di esercizi consigliati sulla base dei muscoli d'interesse inseriti in fase di registrazione ed esercizi consigliati in base a muscoli che l'utente sta trascurando.
  Nel caso in cui invece l'accesso venga effettuato dall'admin, in tale form esso avrà la possibilità di inserire o eliminare degli esercizi dalla lista.
- *Form Home*
  Qui il cliente potrà visionare: i primi tre esercizi più quotati dai clienti, gli esercizi aggiunti più di recente (inseriti entro due giorni da oggi), e la lista totale degli esercizi, tramite bottoni esso potrà interagire per poter aggiungere (o eliminare) tali esercizi nella propria scheda.
- *Form Scheda*
  Qui il cliente potrà visionare la sua scheda che ha precedentemente creato, e nel caso necessitasse anche eliminare qualche esercizio.
- *Form Impostazioni*
  Qui il cliente potrà modificare le proprie impostazioni, in particolare il nome, età, muscoli di interesse e la password, specifichiamo che, se l'utente vorrà cambiare la password, esso dovrà inoltre inserire quella vecchia. Inoltre, verrà notificato un errore nel caso in cui la password nuova sia uguale a quella vecchia.
- *Caricamenti*
  In questo file sono presenti tutte le varie funzioni di caricamento degli esercizi chiamati dai form Account, Home e Scheda.
- *Utils*
  Qui sono presenti le classi utili per deserializzare le risposte json ottenute dalle richieste http POST fatte al server python.
- *Utente*
  Tale singletone ha lo scopo di memorizzare i dati dell'utente, viene infatti usato dai vari form per visualizzare i dati dell'utente come nome, età, muscoli preferiti ecc…
  Esso viene inizializzato nel momento in cui viene effettuato il login e invece viene svuotato nel momento in cui viene eseguito il logout dall'applicazione. Tale classe viene inoltre aggiornata nel caso in cui l'utente interagisse col form impostazioni.

Modulo Python

In python è stata implementata innanzitutto la generazione delle statistiche riguardo i muscoli allenati dall'utente; esso inoltre effettua la validazione dei dati inviati dai form C# prima di inviarli al server Go, come ad esempio nel caso della registrazione, del login o della modifica delle impostazioni utente. Esso inoltre ha il compito di separare le responsabilità utente e admin, di fatti per quanto riguarda le azioni effettuabili dall'admin esso prima di tutto si occupa di chiamare la funzione "authenticate_admin" che nel caso non vada a buon fine ritornerà un errore. Esso si occupa di elaborare gli esercizi da consigliare al cliente in base alle sue informazioni. Il server Python è stato

realizzato grazie all'uso della libreria esterna ***Flask***. Altre librerie esterne utilizzate nel modulo Python sono ***pandas*** e ***matplotlib***, coinvolte nella generazione del grafico citato nel *Form Account*, del Modulo C#.

Il Modulo Python è organizzato in cinque diverse sezioni: *server.py*, *user_manager.py*, *exercise_manager.py*, *utils.py* e *generate_muscle_stats.py*:

- *utils*
  In *utils.py* sono definite le funzioni di utilità per il codice python. In questo file sono definite due funzioni di validazione, per l'e-mail e la password (*is_valid_email* e *is_valid_password*), che validano gli input in base ad un'espressione regolare specifica e una funzione per inviare le richieste http, con un payload passato in input e serializzato come *json*, all'endpoint del server Go, anch'esso passato come parametro (indirizzo e porta sono hard-coded all'interno della funzione).
- *generate_muscle_stats*
  Il codice presente all'interno del file ha la funzione di generare le percentuali dei vari muscoli presenti all'interno della scheda di esercizi (ottenuta in *get_muscoli_allenati_con_ripetuti* del modulo *exercise_manager.py*) e di elaborare un'immagine di un grafico a torta di esse. Ciò è stato implementato utilizzando le *Series* definite nella libreria esterna *pandas* (elaborazione delle percentuali) e l'elaborazione dei grafici della libreria esterna *matplotlib*.
- *server*
  Tale file svolge appunto il ruolo del server (indirizzo: *localhost*, porta: 5000), nel codice troviamo infatti le varie route su cui arrivano le richieste inviate dal client C# (in server.py sono presenti i vari commenti che spiegano il ruolo di ogni route). Esso, quindi, ricevute le richieste, si occuperà di richiamare le rispettive funzioni da user_manager ed exercise_manager o, nel caso di get_grafico, di generate_muscle_stats.
- *user_manager*
  In user_manager.py sono state inserite le funzioni utili alla gestione dell'utente, come ad esempio le funzioni login, registrazione, modifica profilo, get_scheda, authenticate_admin, aggiunta o eliminazione di un esercizio dalla scheda.
- *exercise_manager*
  In exercise_manager.py sono state inserite le funzioni utili alla gestione degli esercizi, come ad esempio il get_scheda, getAllMuscles, get_exercises, get_preferred, get_recent e altri.
  In particolare, troviamo le funzioni:
  - get_consigliati
    Tale funzione, ottenendo informazioni sui muscoli d'interesse dell'utente da user_manager e ottenendo una lista di tutti gli esercizi da get_exercises, fa l'elaborazione per creare e ritornare una lista di esercizi consigliati sulla base dei muscoli d'interesse dell'utente.
  - get_trascurati
    Tale funzione, ottenendo i muscoli allenati dall'utente e la lista di tutti gli esercizi, si occupa di elaborare e ritornare una lista di esercizi consigliati sulla base dei muscoli che l'utente sta trascurando (nessun esercizio nella sua scheda lo allena).

Modulo Go

Il modulo Go si occupa della gestione del database MySQL; in cui sono memorizzate le informazioni riguardanti gli utenti, vari esercizi di allenamento, le tipologie di muscoli e le schede create dagli utenti stessi. La gestione del DB MySQL in Go è stata possibile grazie all'uso del *package* esterno *github.com/go-sql-driver/mysql* (driver) assieme all'uso del package *database/mysql*. Il codice del modulo Go è stato organizzato in quattro diversi *packages*, a seconda del loro scopo: *main* (*main.go*), *database* (*database.go*), *types* (*types.go*) e *utility* (*utility.go*).

- *main*
  Tale *package* si occupa di avviare il server go (indirizzo: *localhost*, porta: 8080) e di definire i vari endpoints del server, gestiti dal *multiplexer*/*router* "*mux*" del package go *"net/http".* Inoltre, in corrispondenza di ogni definizione di un *endpoint* viene definito anche l'*handler* corrispondente attraverso l'uso delle funzioni anonime. Tutti gli handler di ciascun *endpoint* accedono alle informazioni che ricevono dalla richiesta HTTP (inviate dal modulo python in formato *json*) attraverso il puntatore *r* del tipo struct "*http.Request*" del package "*net/http*". Quindi, deserializza le informazioni contenute nel json (tramite un *decoder* del package "*encoding/json*") in apposite struct definite in go, definisce un canale, invia i dati ad una *goroutine*, in modo da eseguire più operazioni in modo concorrente, e attende su un canale, definito all'interno della route, le informazioni resituite dalla *goroutine*. Infine, risponde al modulo python secondo due modalità: in caso di stringhe, attraverso la funzione Write dichiarata dall'interfaccia *http.ResponseWriter*; in caso di risposte *json* codifica le informazioni in tale formato tramite un *encoder* del package *encoding/json*.
- *database*
  Tale package definisce tutte le *goroutines* richiamate dal *main.go*, le quali si occupano dell'accesso e della gestione delle informazioni memorizzate sul database; in particolare gestisce le operazioni di registrazione e di autenticazione degli utenti, dell'admin; delle operazioni di modifica delle informazioni del profilo degli utenti iscritti; delle operazioni di gestione dei dati degli esercizi (aggiunta, eliminazione, recupero); dell'operazione di recupero dei muscoli presenti nel database; delle operazioni di gestione dei dati relativi alle schede di allenamento degli utenti (aggiunta ed eliminazione di un esercizio, recupero degli esercizi della scheda); infine, l'accesso ai muscoli preferiti dell'utente (reperibili dall'interfaccia utente se scelti in fase di registrazione). Infine, sono definite delle funzioni di utilità relative alle varie operazioni effettuate sul database come quella di connessione (*ConnectDB()*) o quelle di recupero degli *id* di specifiche *entries* delle tabelle *users*, *exercises* e *muscles*.
- *types*
  Tale package definisce i vari tipi di dato utilizzati per le fasi di codifica e decodifica *json*, rispettivamente per le richieste in ingresso effettuate dal server python e per le risposte generate dal server Go. A tale scopo, i vari campi delle struct presentano scritto accanto il cosiddetto "*struct tag* "per le operazioni di serializzazione/deserializzazione dei *json*.
- *utility*
  Tale package implementa la funzione di utilità "*FindDifferentStrings*" che, dati due slice di tipo stringa in ingresso, restituisce due *slices* di stringhe: il primo contiene le stringhe presenti nel primo *slice* in ingresso, ma non nel secondo; il secondo *slice* restituito contiene

le stringhe presenti nel secondo *slice* in input, ma non nel primo. Tale funzione viene utilizzata nel package *database* all'interno della funzione *ModifyPreferredMuscles*, la quale si occupa di rimuovere dal DB i muscoli non presenti nella richiesta di modifica in ingresso dal server Python e di inserire nel DB i muscoli presenti nella richiesta e non nel DB stesso.