Documentazione Health Manager

Francesco Bonistalli

January 2025

Indice

1 Introduzione **2**
1.1 Caso d'uso . . . 2
1.2 Struttura dell'applicazione . . . 2

2 Client **3**
2.1 Guida alle pagine dell'applicazione . . . 3
2.1.1 Start Page . . . 3
2.1.2 Login Page . . . 3
Registration page . . . 4
2.1.3 Home Page . . . 5
Management Pages . . . 5
Insert Pages . . . 6

3 Server **7**

4 Codice generato da ChatGPT **8**
4.1 Styling pagine JavaFX . . . 8
4.2 Lettura da file .csv . . . 8
4.3 Corpo richiesta POST . . . 9

Capitolo 1

Introduzione

1.1 Caso d'uso

HealthManager è un'applicazione java pensata per permettere all'utente di poter tenere traccia e gestire tutto ciò che riguarda la propria salute. In particolare si riferisce a persone che praticano attività sportiva e vogliono tenere traccia di quelli che sono i più importanti fattori che determinano lo stato di forma di un atleta:

- **Allenamento:** l'applicazione permette di registrare le informazioni di una sessione di allenamento.
- **Alimentazione:** l'applicazione permette di registrare i pasti, fondamentali per reintegrare gli sforzi dell'allenamento.
- **Recupero:** l'applicazione permette di registrare i sonni, fondamentali per permettere al corpo di recuperare dagli sforzi.

1.2 Struttura dell'applicazione

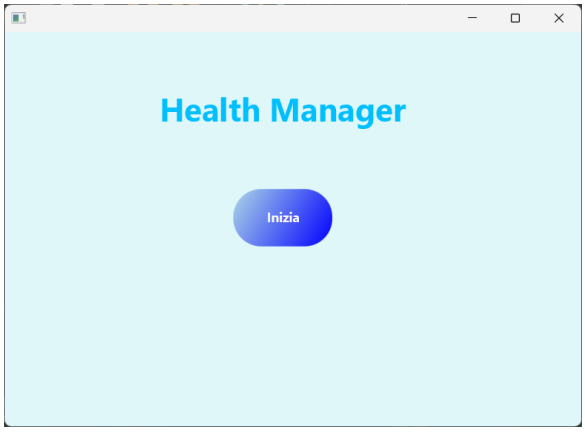
L'applicazione è divisa tra client e server. La parte client è costituita da un'interfaccia grafica costruita con JavaFX che interagisce con la parte server tramite richieste http. Il server a sua volta contiene tutte le funzioni per la gestione dei dati e delle funzioni richieste dal client, oltre che all'interazione con il database.

Capitolo 2

Client

2.1 Guida alle pagine dell'applicazione

2.1.1 Start Page



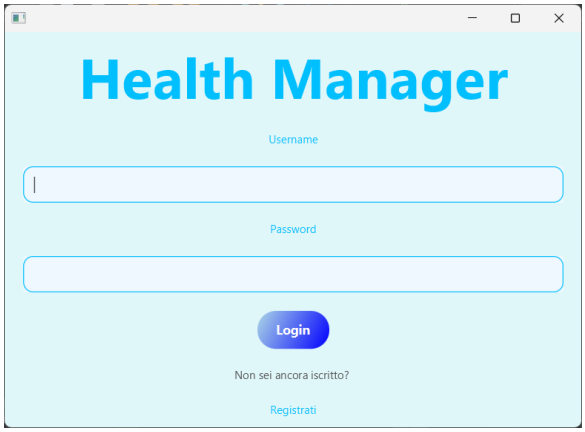
Il client, una volta avviato, apre una pagina iniziale che permette l'inserimento, tramite la pressione del tasto "Inizia", nel database di alcuni dati[1] a partire da dei file *.csv* presenti nella cartella "filePopolamento" localizzata, nel server, all'indirizzo:

$$"\backslash src\backslash main\backslash resources\backslash filePopolamento\backslash"$$

2.1.2 Login Page

Dalla pressione del tasto nella pagina precedente si viene portati a una schermata di login: l'applicazione è infatti pensata per essere utilizzata da più utenti in modo che ognuno possa tenere conto delle proprie attività.

[1] I dati iniziali presenti nel database sono fasulli e generati casualmente



Questa schermata avvisa l'utente che sta provando ad accedere nel caso in cui la password inserita sia errata o nel caso in cui l'username inserito non sia presente nel database.

Figura 2.1: In questi esempi vediamo come: nel primo, una password vuota risulta errata, nel secondo, invece, l'user "user" non sia registrato nel sistema ("user" non è un utente tra quelli caricati inizialmente nel database)

Registration page

Per ovviare al problema visto nell'ultimo caso, ovvero quello di utente non registrato, si può accedere alla pagina di registrazione, come suggerito dal testo presente in basso alla pagina di login, tramite un click sulla scritta *"Registrati"*.

Anche in questo caso l'utente viene avvisato di problemi quali il mancato riempimento di alcuni campi o l'indisponibilità dell'username, quest'ultimo deve essere infatti univoco per ogni utente.

2.1.3 Home Page

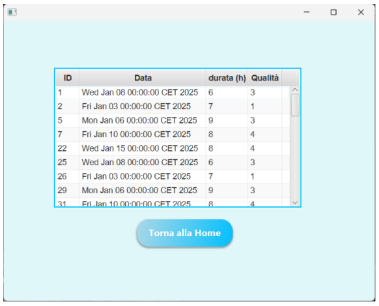
Nel caso il login avvenga correttamente l'utente viene portato alla pagina principale dell'applicazione dove ha accesso a tutte le funzionalità che gli permettono di gestire le proprie attività sportive e ciò che ne concerne.

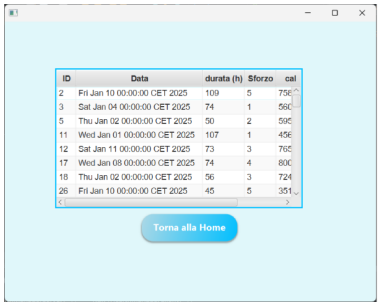
In questa pagina, tramite la pressione del tasto in basso *"Torna al login"* è possibile tornare alla pagina precedente dalla quale sarà nuovamente necessario inserire username e password per essere rimandati a questa schermata.

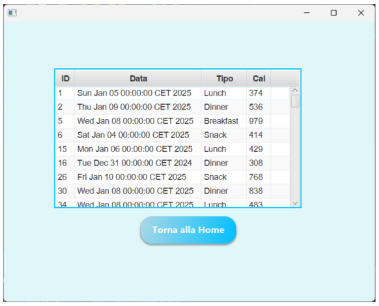
Effettuando invece un click su uno dei tre pannelli si verrà rimandati alle rispettive pagine di gestione mentre, sfruttando i vari tasti *"Aggiungi"*, si può registrare una nuova istanza di ognuno dei tre elementi chiave dell'attività sportiva.

Management Pages

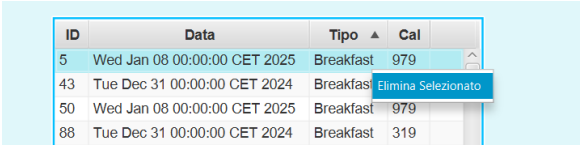
Tramite queste pagine è possibile visualizzare, tramite una tabella, tutti gli elementi aggiunti dall'utente per ogni specifico utilizzo.





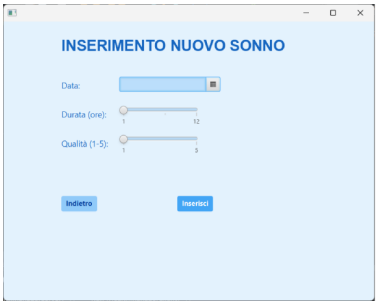


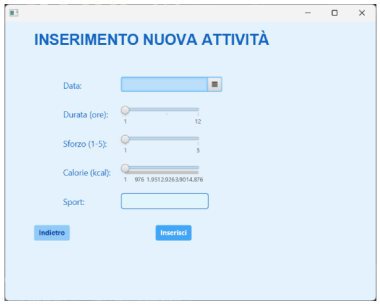
Tramite la pressione del tasto *"Torna alla Home"* è possibile tornare alla schermata principale mentre tramite un click con il tasto destro, a seguito della selezione di un elemento sulla tabella, è possibile aprire un menu a scomparsa che permette la rimozione di un elemento dalla tabella (e di conseguenza anche dal database).

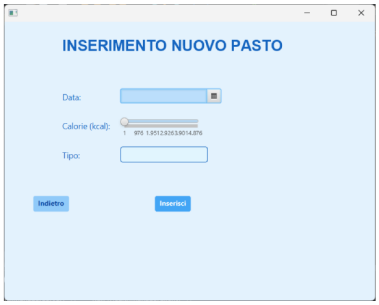


Insert Pages

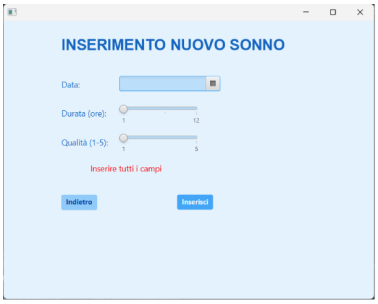
Le pagine di inserimento appaiono in questo modo:

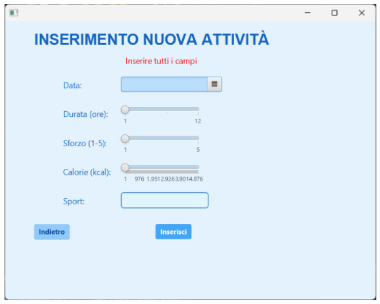


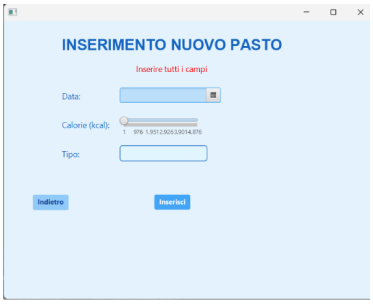




Anche queste presentano messaggi di errore visibili all'utilizzatore in caso di mancanza di parametri in inserimento.







Capitolo 3

Server

Lato server l'applicazione è strutturata per gestire tutte le richieste effettuabili dal client.
Queste rispondono tutte al *path* di @RequestMapping corrispondente a

$$/HealthManager$$

Si dividono poi le richieste tramite l'utilizzo di tre controllori:

1. **MainController**: risponde solo alle richieste *GET* relative al **path = "/stato"** utlizzato per testare lo stato delle richieste al momento della prima connessione nella pagina iniziale.
2. **GetDataController**: risponde anche questo a sole richieste *GET*, in particolare è composto dai *path*:
   - */utente* che si occupa di recuperare un utente dato l'username.
   - */utenti/tutti* che si occupa di recuperare tutti gli utenti sotto forma di *List<Utente>*
   - */attivita* che si occupa di recuperare tutte le attivita sotto forma di *List<Attivita>*
   - */sonno* che si occupa di recuperare tutti i sonni sotto forma di *List<Sonno>*
   - */pasto* che si occupa di recuperare tutti i pasti sotto forma di *List<Pasto>*
3. **InsertDataController**: risponde invece a sole richieste *POST*, in particolare si compone dei *path*:
   - */nuovoUtente* che si occupa di salvare un nuovo utente.
   - */nuovoSonno* che si occupa di salvare un nuovo sonno.
   - */nuovoAttivita* che si occupa di salvare una nuova attività.
   - */nuovopPasto* che si occupa di salvare un nuovo pasto.

Capitolo 4

Codice generato da ChatGPT

4.1 Styling pagine JavaFX

Tutte le pagine create con javaFX, dopo essere state progettate e strutturate (compresi inserimenti di campi #id o campi #action) sono state modificate nello stile da ChatGPT a seguito del seguente prompt: *make this appear more nice to the view, using blue/lightblue as the theme color*

4.2 Lettura da file .csv

Prompt: *scrivi una funzione "utenteCsvToList" che dato un file csv contente in ogni riga dati utente secondo il seguente formato:*
username,nome,cognome,password,altezza,peso,eta,calorie_giornaliere
li legga e converta il file il una List di dati "Utente" dove "Utente" è una classe con i seguenti campi:
private String username;
private String nome;
private String cognome;
private String password;
private String eta;
private String altezza;
private String peso;
private String calorie_giornaliere;
che ha ritornato il codice:

```
1     public static List<Utente> utenteCsvToList(String filePath) {
2       List<Utente> utenti = new ArrayList<>();
3       try (BufferedReader br = new BufferedReader(new
           FileReader(filePath))) {
4           String line;
5           while ((line = br.readLine()) != null) {
6               // Split della riga in base alla virgola
7               String[] fields = line.split(",");
8
9               // Creazione di un oggetto Utente con i dati della
                   riga
```

```
                if (fields.length == 8) {  // Assicurarsi che ci siano 8 campi
                    Utente nuovoUtente = new Utente(
                        fields[0], // username
                        fields[1], // nome
                        fields[2], // cognome
                        fields[3], // password
                        Integer.valueOf(fields[4]), // eta
                        Integer.valueOf(fields[5]), // altezza
                        Integer.valueOf(fields[6]), // peso
                        Integer.valueOf(fields[7])  // calorie_giornaliere
                    );
                    utenti.add(nuovoUtente);  // Aggiungi l'utente alla lista
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return utenti;
    }
```

4.3 Corpo richiesta POST

Non trovando sulle slide del laboratorio come settare i parametri visti nel comando *curl* (usato a lezione per testare il funzionamento delle richieste dal prompt dei comandi) per una richiesta POST ho inserito il seguente prompt: *sto effettuando una richiesta POST al server al quale vorrei passare i parametri di nome nuovo... come posso fare per inserirli nella richiesta e non nell'url?* che mi ha restituito il codice:

```
    try {
    // Ottieni i dati da inviare
    Date dataNuovo =
    Date.from(DataNuovoSonno.getValue().
            atStartOfDay(ZoneId.systemDefault()).toInstant());
    int durataNuovo = (int) SliderDurataNuovoSonno.getValue();
    int qualitaNuovo = (int) SliderQualitaNuovoSonno.getValue();
    String userNuovo = utenteAttivo.getUtenteAttivo();

    // Creazione URL per il server
    URL url = new
        URL("http://127.0.0.1:8080/HealthManager/nuovoSonno");

    // Configurazione connessione HTTP
    HttpURLConnection connessioneHttp = (HttpURLConnection)
        url.openConnection();
    connessioneHttp.setRequestMethod("POST");
    connessioneHttp.setRequestProperty("Content-Type",
        "application/json; utf-8");
    connessioneHttp.setRequestProperty("Accept",
        "application/json");
```

```
18      connessioneHttp.setDoOutput(true); // Abilita il flusso di
            output
19
20      // Creazione del corpo della richiesta come JSON
21      String jsonBody = String.format(
22          "{\"data\":\"%s\", \"durata\":%d, \"qualita\":%d,
                \"user\":\"%s\"}",
23          dataNuovo.toString(), durataNuovo, qualitaNuovo, userNuovo
24      );
25
26      // Scrittura del corpo della richiesta nel flusso di output
27      try (OutputStream os = connessioneHttp.getOutputStream()) {
28          byte[] input = jsonBody.getBytes("utf-8");
29          os.write(input, 0, input.length);
30      }
31
32      // Invio della richiesta e lettura della risposta
33      int responseCode = connessioneHttp.getResponseCode();
34      System.out.println("Codice risposta: " + responseCode);
35
36      if (responseCode == HttpURLConnection.HTTP_OK) {
37          System.out.println("Richiesta inviata con successo!");
38      } else {
39          System.out.println("Errore nella richiesta.");
40      }
41
42  } catch (Exception e) {
43      e.printStackTrace();
44  }
```