Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Informatica
A.a. 2022/2023
Sessione di Laurea Dicembre 2023

HealthApp - Progettazione e Sviluppo dell'Interfaccia Utente di una Web Application per il Monitoraggio del Benessere degli Assistiti Medici

Relatore:
Prof. Maurizio Morisio

Candidato:
Francesco RUSSO

INDICE

1. PRESENTAZIONE

Health App è un'iniziativa nata dalla collaborazione tra i medici dell'Azienda Ospedaliera di Verona e il Dipartimento di Automatica e Informatica (DAUIN) del Politecnico di Torino. Il progetto mira a sviluppare un software medico per il monitoraggio dei parametri salutari dei pazienti offrendo consulenze personalizzate al fine di migliorare il loro stato di salute.

L'idea di base di questo esperimento è valutare se sia possibile avere un impatto positivo sullo stile di vita di una particolare categoria di persone. L'obiettivo è incoraggiare una maggiore attività fisica, migliorare la qualità del sonno e promuovere una dieta sana. L'applicativo è pensato per soddisfare le esigenze di diverse tipologie di utenti: i pazienti possono consultare i propri parametri salutari e ricevere raccomandazioni tramite l'app mobile, mentre i medici gestiscono e monitorano i loro assistiti tramite un'interfaccia web.

È evidente che per condurre analisi accurate e affidabili sia necessario raccogliere una vasta gamma di dati. In quest'ottica, il tracker Fitbit gioca un ruolo chiave, poiché permette ai pazienti di monitorare il sonno, i passi e i battiti cardiaci.

Tuttavia, per avere a disposizione un quadro completo delle abitudini quotidiane dei pazienti, è altrettanto importante acquisire informazioni riguardanti la loro alimentazione. Inoltre, il sistema offre anche delle funzionalità che consentono ai medici di inserire valutazioni personalizzate per ciascun paziente. Questa combinazione di fattori offre una visione a tutto tondo dello stato di salute dei pazienti e garantisce un supporto medico su misura, contribuendo al miglioramento della loro qualità di vita.

1.1. OBIETTIVO DELLA TESI

In questo lavoro di tesi l'attenzione è stata posta sulla progettazione dell'interfaccia utente dell'applicazione web destinata ai medici. Oggi, sviluppare un'applicazione che risponda pienamente alle esigenze degli utenti è una sfida sempre più impegnativa. Ciò è stato possibile grazie al costante coinvolgimento dei medici, che hanno fornito un prezioso feedback in fase di progettazione, contribuendo così a garantire la realizzazione di una UI estremamente intuitiva e di facile utilizzo.

Il raggiungimento di questo obiettivo è stato facilitato dall'uso della libreria React che ha consentito di sviluppare un'applicazione web altamente reattiva e user-friendly, fornendo la flessibilità necessaria per implementare rapidamente le funzionalità richieste e adattarle alle necessità del nostro pubblico target, garantendo così un'esperienza utente ottimale.

In fase di sviluppo sono state inoltre utilizzate alcune librerie chiave per migliorare la qualità del progetto:

- Redux, per gestire in modo efficiente lo stato dell'applicazione, assicurando un'organizzazione chiara dei dati.
- Material-UI, per creare un'interfaccia utente intuitiva e attraente, che non solo soddisfi gli standard estetici, ma migliori anche l'usabilità complessiva.
- Axios per effettuare richieste HTTP da un client web a un server.

1.2. METODOLOGIA

La metodologia scelta per il nostro processo di sviluppo è stata quella Agile [2]. La scelta di adottare tale approccio è stata guidata dalla necessità di avere un processo di sviluppo incrementale, con un'attenzione mirata a specifiche funzionalità e un costante coinvolgimento dei medici al fine di ottenere il loro prezioso riscontro. Di conseguenza, vengono programmati meeting periodici con il team e il professore al fine di delineare i task e le attività da svolgere. Questi incontri si svolgono su base settimanale, in sintonia con la durata tipica di uno sprint all'interno del nostro processo di sviluppo. Questa metodologia si sposa bene con contesti in cui l'obiettivo è chiaramente definito ma il risultato non lo è ancora. Nel corso del tempo il progetto ha infatti subito una forte evoluzione grazie agli input del team e al feedback costante degli stakeholder. Il risultato finale rappresenta dunque una sintesi di questi contributi.

1.3. ARCHITETTURA

Health App adotta il tradizionale pattern architetturale a tre livelli (three-tier). L'architettura a tre livelli [1] offre diversi vantaggi, tra cui la separazione chiara delle responsabilità, la scalabilità e la manutenibilità. Ogni livello può essere infatti sviluppato e gestito in modo indipendente, semplificando quindi l'aggiornamento e la manutenzione del sistema.

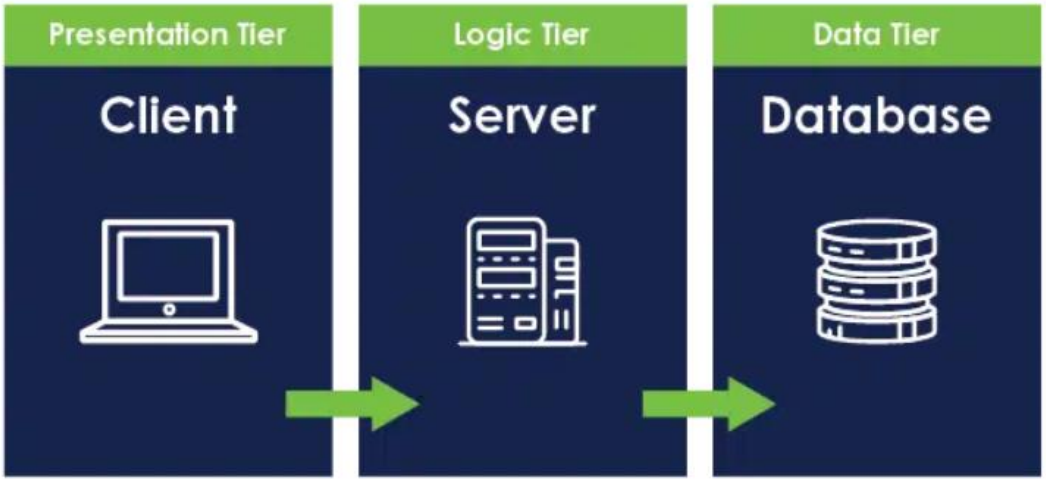

Figura 1 - Three Tier Architecture

Ecco una panoramica di questi tre livelli:

- Presentation tier: è responsabile dell'interazione diretta con gli utenti. Gestisce la user interface e la presentazione dei dati agli utenti.
- Application/Logic tier: è il cuore dell'applicazione, infatti contiene la logica di business e le regole che guidano il comportamento del software. È responsabile dell'elaborazione dei dati e delle operazioni che l'applicazione deve eseguire.
- Data tier: è il livello dedicato alla gestione dei dati. Si occupa di garantire la persistenza e l'accessibilità dei dati necessari all'applicazione.

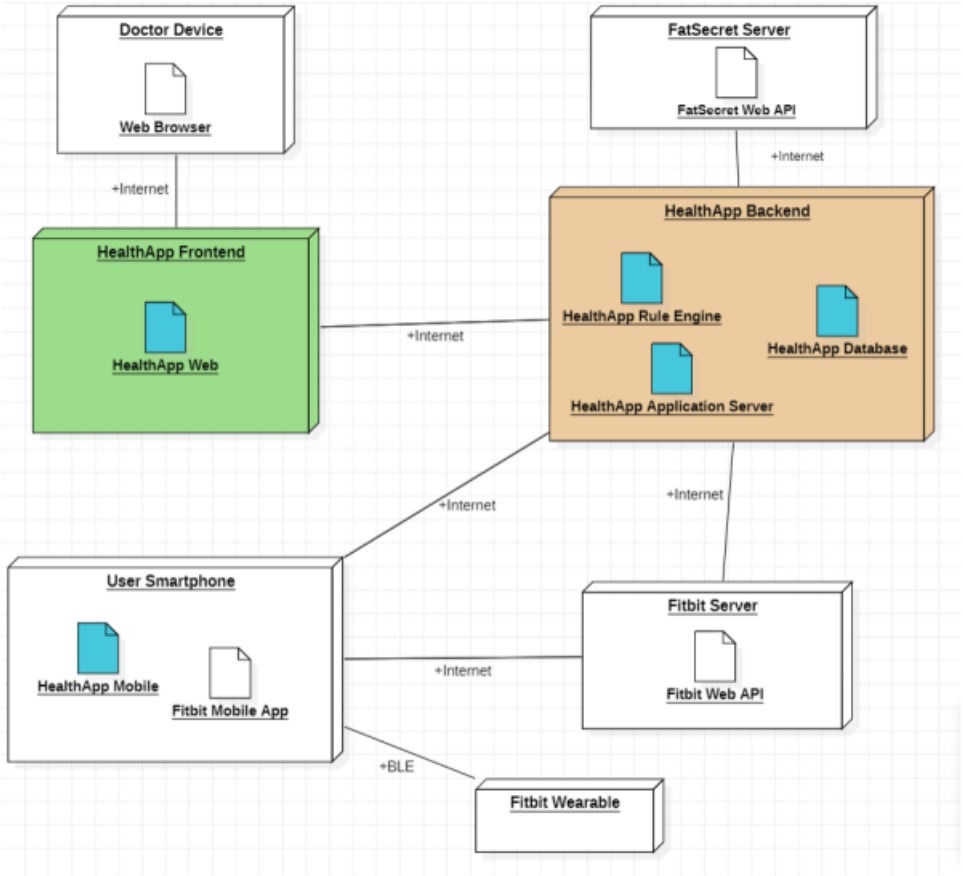

Figura 2 - Deployment Diagram

1.4. BACKEND

Molto spesso, application tier e data tier vengono considerati come un unico elemento, noto come “backend”, che rappresenta la parte di un’applicazione che si occupa della gestione dei dati e della logica di business. Nel progetto Health App esso è stato sviluppato utilizzando Spring Boot [3], un framework noto per la sua facilità d'uso e la capacità di accelerare il processo di sviluppo, rendendo più semplice la creazione di applicazioni robuste, scalabili e altamente performanti.

Spring Boot è ampiamente utilizzato per sviluppare una vasta gamma di applicazioni e deve la sua popolarità ad una serie di vantaggi che lo contraddistinguono, infatti:

- semplifica notevolmente lo sviluppo di applicazioni Java fornendo configurazioni predefinite e basate su convenzioni. Questo riduce la presenza di codice boilerplate.

- ha un server web integrato (Tomcat o Netty), che rende il rilascio più semplice.
- gestisce automaticamente le dipendenze delle librerie, utilizzando Maven o Gradle per la gestione delle build.
- offre degli starter preconfezionati, ossia delle dipendenze pronte all'uso specifiche per scenari comuni.
- garantisce supporto per la sicurezza tramite Spring Security [5].

Per queste ragioni, si è scelto di utilizzare Spring Boot con Kotlin [6], un moderno linguaggio di programmazione che eleva la produttività, migliora la leggibilità del codice e rafforza la sicurezza delle applicazioni [7].

Una possibile soluzione alternativa è rappresentata dall'adozione del framework Express [8] in ambiente Node.js. Sebbene questo approccio sia noto per le sue prestazioni elevate e la scalabilità, può incontrare difficoltà se usato in applicazioni che richiedono molte risorse o elaborazioni intensive. Questo può avvenire perché Node.js è basato su un modello di gestione asincrona [9], il che significa che è particolarmente efficace per gestire un gran numero di richieste simultanee leggere, ma potrebbe subire un rallentamento quando si tratta di operazioni CPU-intensive o richieste di lunga durata.

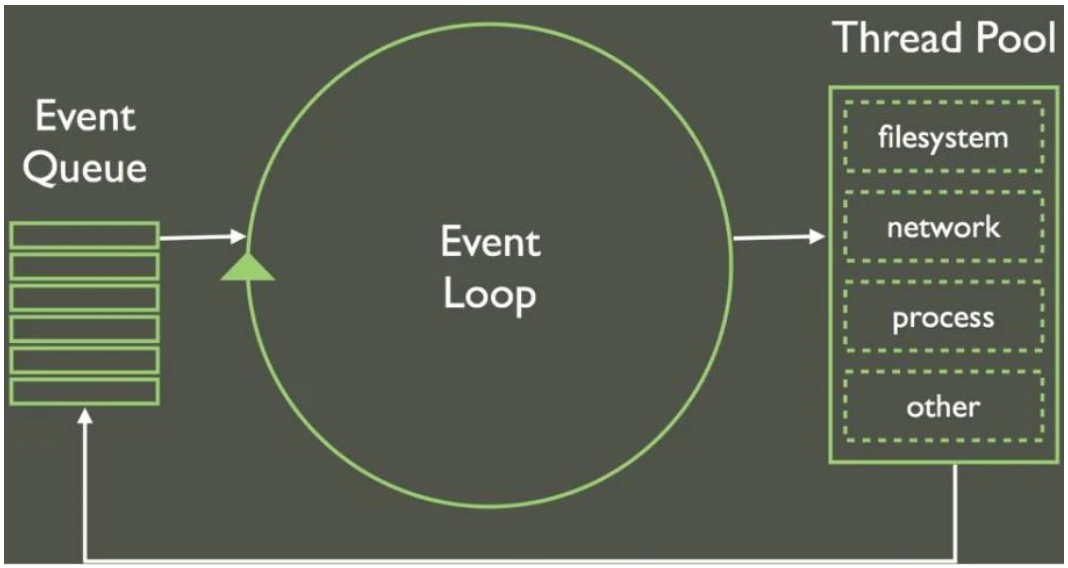

Figura 3 - Node Single Threaded Event Loop

Spring Boot, al contrario, sfrutta il multithreading. Infatti, grazie al server web integrato, è in grado di gestire le richieste in modo concorrente assegnando un thread separato a ciascuna richiesta in arrivo.

Per concludere, questa parte dell'applicazione ha il compito di comunicare con servizi esterni per raccogliere informazioni sui pazienti e mettere a disposizione delle API REST. Da segnalare anche l'integrazione di Drools, un Business Rules Engine che, basandosi sui dati raccolti nel tempo, offre consulenze e suggerimenti personalizzati ai pazienti.

1.5. FRONTEND

Come anticipato precedentemente, il sistema è stato progettato per soddisfare le diverse esigenze degli utenti, pertanto l'obiettivo è quello di realizzare sia la Web App che la Mobile App, entrambe basate sulla libreria React.

1.5.1. MOBILE APPLICATION

L'applicazione Mobile, come accennato in precedenza, è esclusivamente dedicata agli assistiti medici coinvolti nell'esperimento Health App, sebbene per ora sia stata messa in stand-by. E' stata sviluppata utilizzando React Native [25], un framework cross-platform in grado di creare app native per iOS e Android basate su una singola code base, sfruttando comunque la potenza e le principali features che hanno reso famosa la libreria React. Questo approccio permette agli sviluppatori di scrivere il codice una sola volta e di riutilizzarlo per entrambi i sistemi operativi, riducendo così i costi e i tempi di sviluppo.

1.5.2. WEB APPLICATION

La Web App di Health App è stata sviluppata usando React [10], una libreria open-source javascript utilizzata per la creazione di interfacce utente basate su componenti.

React si distingue per alcune caratteristiche chiave:

Component-Based: React si basa su un'architettura a componenti, suddividendo quindi l'interfaccia utente in componenti riutilizzabili che possono essere sviluppati separatamente e riuniti per creare l'intera UI. Questo approccio favorisce la modularità, la manutenibilità e la facilità di sviluppo delle applicazioni;

Virtual DOM: React utilizza un Virtual DOM [23], ossia una rappresentazione virtuale della struttura DOM dell'applicazione. Quando avviene un cambiamento nello stato di un componente, React compara il Virtual DOM con il DOM reale e aggiorna solo le parti che sono state effettivamente modificate. Questo approccio

ottimizza le prestazioni dell'applicazione, riducendo la necessità di aggiornare l'intero DOM.

Unidirectional Data Flow: il flusso dei dati si muove in un'unica direzione [24], tipicamente dall'alto verso il basso. Questo rende più facile comprendere come i dati vengano aggiornati e gestiti nell'applicazione, migliorando la tracciabilità dello stato.

JSX (JavaScript XML): utilizzato in React per definire la struttura dell'interfaccia utente all'interno del codice JavaScript. JSX [22] è un'estensione di JavaScript che consente di dichiarare gli elementi UI in modo simile all'HTML. Questo rende la sintassi più chiara e familiare per gli sviluppatori.

In React, i componenti possono essere implementati come classi o funzioni. Quest'ultimo rappresenta l'approccio più semplice e intuitivo.

```
function Ciao(props) {
  return <h1>Ciao, {props.nome}</h1>;
}
```

Figura 4 - Esempio di functional component

```
class Ciao extends React.Component {
  render() {
    return <h1>Ciao, {this.props.nome}</h1>;
  }
}
```

Figura 5 - Esempio di class component

Dal punto di vista concettuale, i componenti [11] sono a tutti gli effetti come delle funzioni, in quanto accettano dati arbitrari in input, noti come "props", e restituiscono elementi React che specificano cosa dovrebbe essere visualizzato sullo schermo. La funzione definita nella figura precedente, ad esempio, costituisce un componente valido di React in quanto accetta un singolo oggetto "props" come argomento con dati e ritorna un elemento React. Nello specifico notiamo che, attraverso la sintassi JSX, un componente funzionale restituisce sempre del codice HTML.

Come accennato precedentemente, React ha degli oggetti speciali chiamati "props" [11] (da proprietà) che si usano per trasportare dati da un componente all'altro. Dati che però possono essere trasportati soltanto in una direzione, ossia

dal componente genitore al figlio. Ecco quindi un esempio pratico del concetto di Unidirectional Data Flow di cui sopra.

Un tratto distintivo dei componenti React è la loro necessità di agire come funzioni pure. Per funzione pura si intende una funzione che, a partire dallo stesso input (le props in questo caso), restituisce sempre il medesimo output, non modificando mai i suoi argomenti di input (le props sono read-only) e non generando effetti collaterali, come la modifica di variabili globali o l'interazione con il DOM.

Naturalmente, le interfacce utente sono dinamiche e subiscono variazioni nel corso del tempo. Di conseguenza, i componenti funzionali in React sono progettati per essere delle "quasi" funzioni pure, introducendo il concetto di "stato". Gli stati in React rappresentano dati mutabili che possono essere utilizzati all'interno di un componente per tenere traccia delle informazioni che cambiano nel tempo.

Hook

In React, un hook [26] è una funzione speciale che permette di "agganciarsi" o utilizzare le funzionalità di React in componenti basati su funzioni anziché componenti basati su classi. Gli hooks sono stati introdotti in React 16.8 per consentire ai componenti funzionali di gestire lo stato locale, effetti collaterali e altri aspetti precedentemente disponibili solo nei componenti di classe.

```
import React, { useState } from 'react';

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Figura 6 - esempio di State Hook

L'hook “useState” è un hook che permette ai componenti basati su funzioni di gestire lo stato locale. Lo si utilizza quando si desidera memorizzare e aggiornare lo stato all'interno di un componente.

```
import React, { useState, useEffect } from 'react';

function Esempio() {
  const [contatore, setContatore] = useState(0);

  // Simile a componentDidMount e componentDidUpdate:
  useEffect(() => {
    // Aggiorna il titolo del documento usando le API del browser
    document.title = `Hai cliccato ${contatore} volte`;
  });
```

Figura 7 - Esempio di Effect Hook

L'hook “useEffect” permette di gestire gli effetti collaterali all’interno dei componenti. Gli effetti collaterali possono essere azioni come richieste API, aggiornamenti del DOM, sottoscrizioni a eventi, o qualsiasi altra operazione che coinvolge l'interazione con il mondo esterno.

```
useEffect(() => {
  // codice da eseguire quando è attivato l'effetto
}, [dependencies]);
```

Figura 8 - useEffect() dependencies

- Il blocco di codice all'interno della callback [27] viene eseguito ogni volta che il componente viene reindirizzato o quando cambia il valore di una delle dipendenze. In questa sezione, vengono gestiti gli effetti collaterali associati alle interazioni con il componente, come la modifica dello stato, l'interazione con API esterne o altre operazioni asincrone.
- L'array delle dipendenze [27] rappresenta la lista dei valori da monitorare. Ogni volta che uno degli elementi di questa lista viene modificato, la callback viene eseguita.

Per quanto riguarda l'interfaccia utente di Health App, sono stati impiegati prevalentemente componenti funzionali con un ampio utilizzo di state hook e effect hook.

2. ASPETTI FUNZIONALI DELLA WEB APPLICATION

In questo capitolo verranno illustrate le principali funzionalità messe a disposizione dei medici all'interno dell'applicazione. Vedremo come quest'ultima faciliti il monitoraggio dei dati dei pazienti da parte dei medici e fornisca ai pazienti un quadro chiaro e completo del loro stato di salute. Scopriremo inoltre le diverse opzioni e gli strumenti a disposizione, evidenziando come questa sia una soluzione semplice e intuitiva per offrire consulenze su misura per migliorare il benessere dei pazienti.

2.1. REGISTRAZIONE DEL DOTTORE

La registrazione del dottore rappresenta una fase preliminare che viene completata durante la configurazione iniziale dell'applicazione. Questo compito è affidato a un amministratore, che, come vedremo in seguito, avrà anche la responsabilità di creare i questionari.

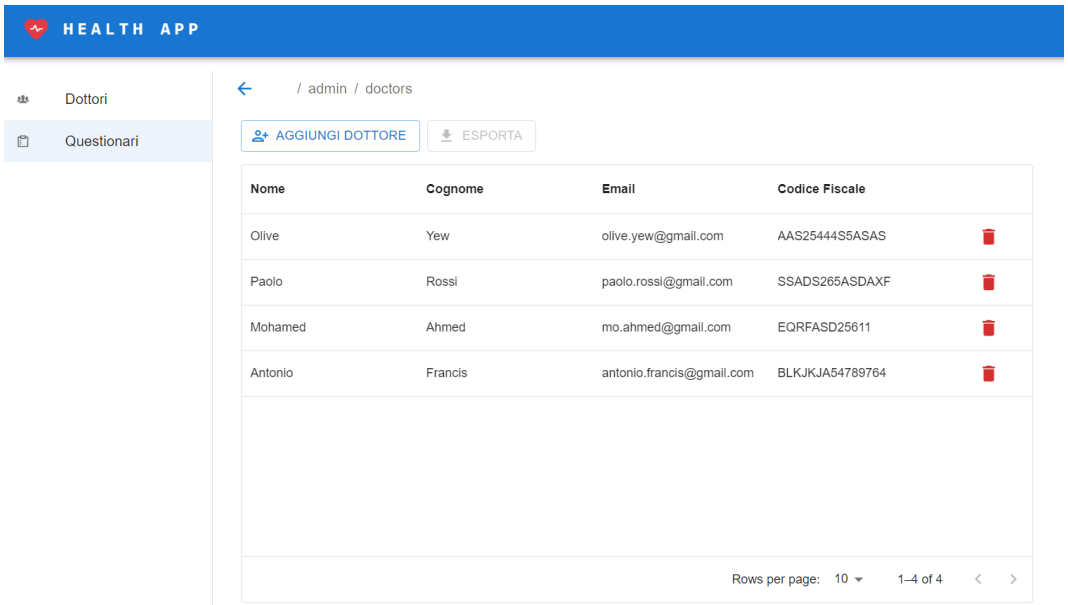

Figura 9 - Administrator Home Page

Subito dopo aver eseguito la procedura di login, è possibile creare un nuovo dottore cliccando sul pulsante 'Aggiungi Dottore' presente nella home page (figura 9).

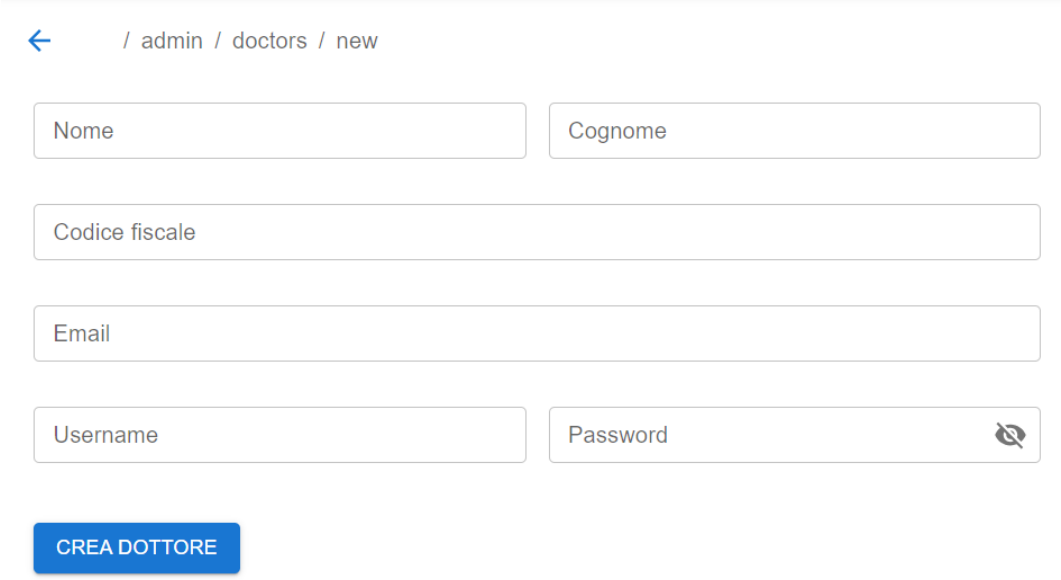

Figura 10 - Form di inserimento di un nuovo dottore

È importante notare che, anche se questa procedura viene eseguita in fase di configurazione del sistema, è necessario inserire anche le credenziali che il dottore utilizzerà per accedere autonomamente all'applicazione. Tuttavia, va sottolineato che il dottore ha la possibilità di modificare la password in qualsiasi momento. Pertanto, la password inserita in fase di registrazione può essere considerata come una password provvisoria, finalizzata a garantire il primo accesso e completare il processo di registrazione.

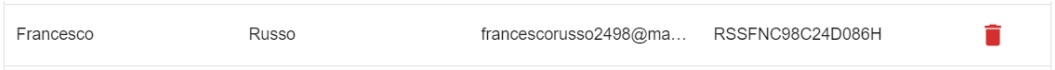

Figura 11 - Esempio di visualizzazione dei dati riferiti al singolo dottore

Dopo aver inserito con successo tutti i dati necessari, il dottore appena registrato sarà immediatamente visualizzato nell'elenco dei dottori presente nella home page dell'amministratore. Una volta conclusa questa procedura, la registrazione sarà completata con successo, consentendo al dottore di accedere alla propria area personale, aggiornare la password e inserire i propri assistiti.

2.2. REGISTRAZIONE DEL PAZIENTE

La registrazione del paziente rappresenta una delle funzionalità fondamentali dell'applicazione poiché permette ai pazienti di diventare parte integrante del sistema Health App. Questo task è svolto dai medici e necessita della raccolta dei dati identificativi del soggetto da registrare.

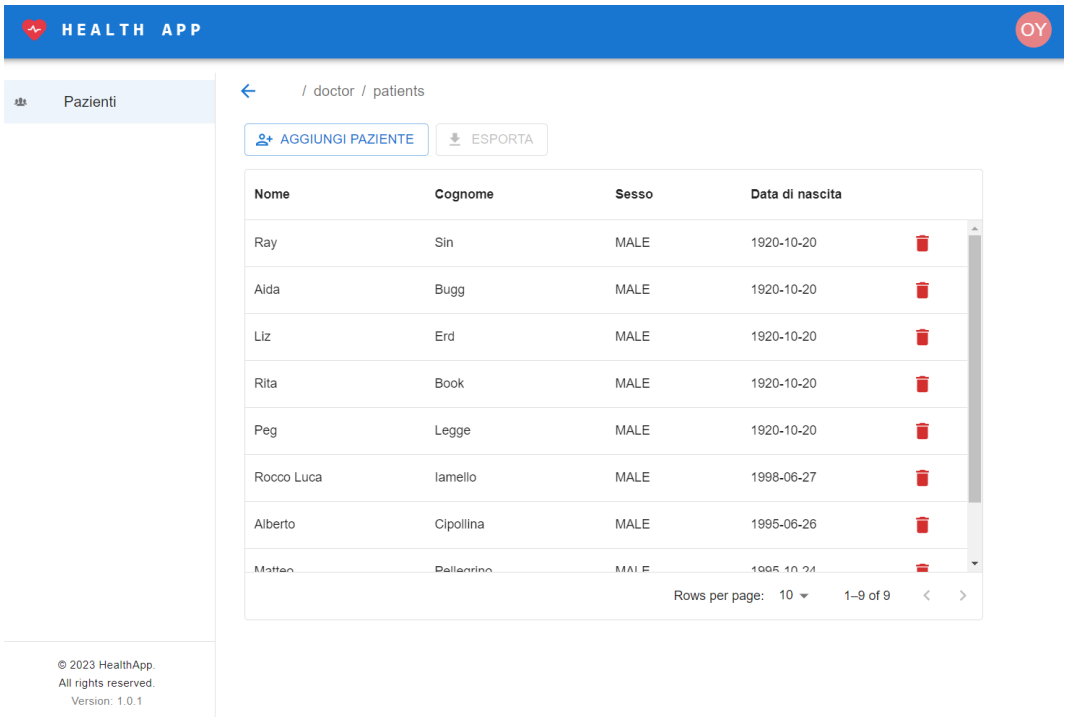

Figura 12 - Doctor Home Page

Subito dopo aver eseguito la procedura di login, il medico può inserire un nuovo assistito cliccando sul pulsante 'Aggiungi Paziente' presente nella home page (figura 12).

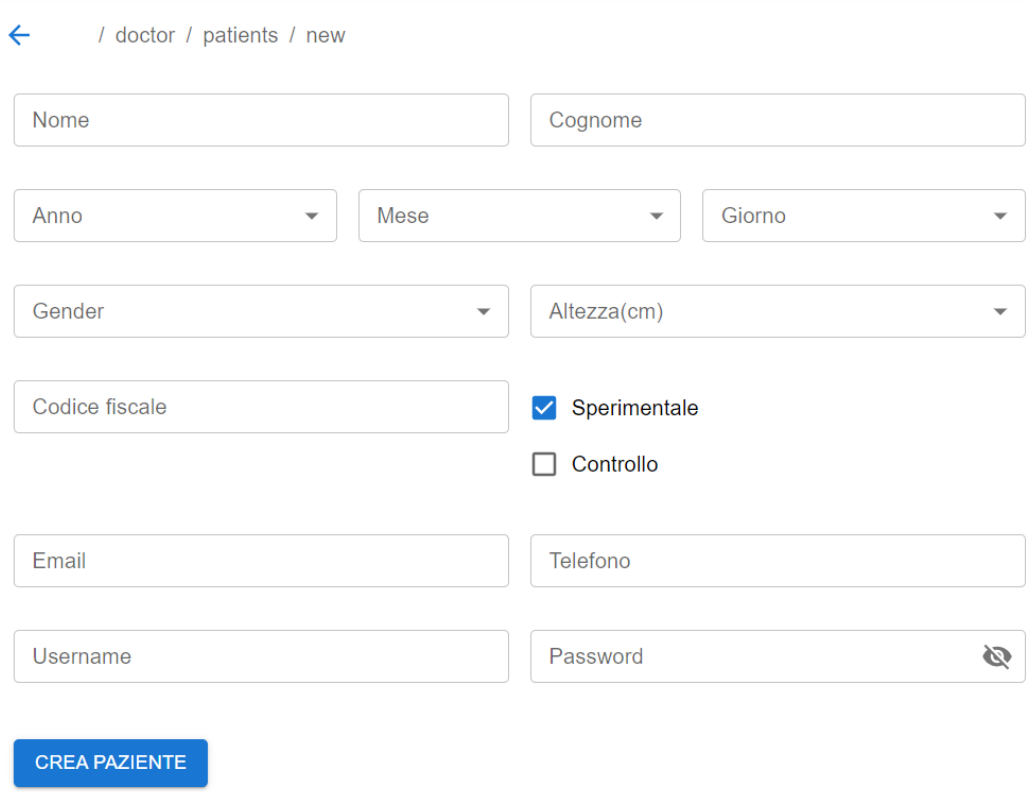

Figura 13 – Form di inserimento di un nuovo paziente

È importante notare che, anche se questa procedura viene eseguita dal medico in presenza del paziente, è necessario inserire anche le credenziali che il paziente utilizzerà per accedere autonomamente all'applicazione. Tuttavia, va sottolineato che il paziente può cambiare la password in qualsiasi momento. Pertanto, la password inserita dal medico durante la fase di registrazione può essere considerata come una password provvisoria, finalizzata a garantire il primo accesso e completare il processo di registrazione. Prima di confermare, è possibile indicare se il paziente partecipa all'esperimento Health App o se utilizza l'applicazione unicamente per monitorare i propri parametri biometrici. A tal fine, è necessario selezionare l'opzione "Sperimentale" nel primo caso e "Controllo" nel secondo.

Figura 14 - Esempio di visualizzazione dei dati di un paziente nella Home Page dei medici

Dopo aver inserito con successo tutti i dati necessari, il paziente appena registrato sarà immediatamente visualizzato nell'elenco dei pazienti presente nella home page dei medici.

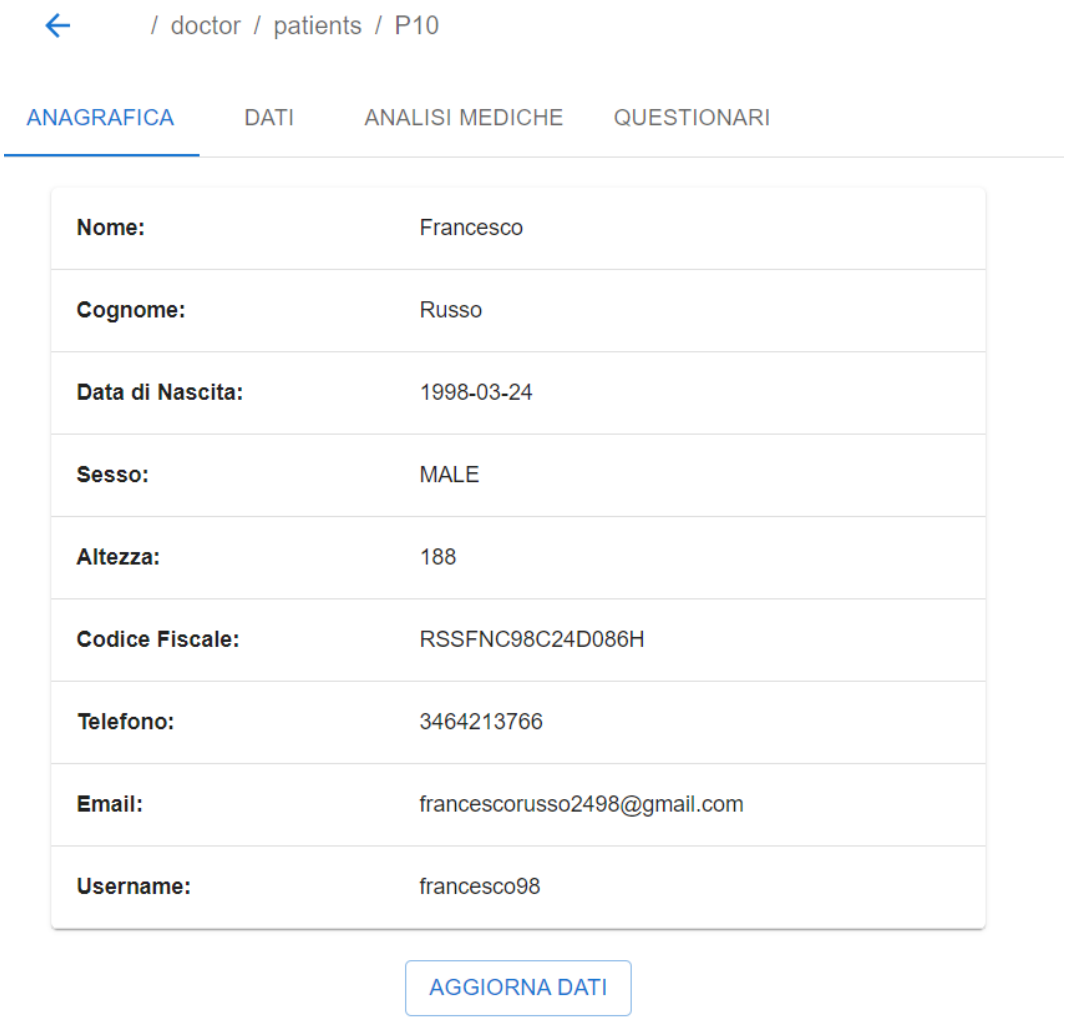

Figura 15 - Anagrafica di un paziente

Nella fase di creazione di un paziente, vengono raccolti numerosi dati. Tuttavia, nella homepage del dottore vengono mostrate solamente le informazioni identificative di ciascun paziente, come nome, cognome, sesso e data di nascita. Questo approccio è stato adottato per rendere la schermata iniziale più concisa e meno dispersiva.

Ad ogni modo, se un utente fa clic su un paziente specifico, sarà in grado di visualizzare in dettaglio tutti i dati relativi a quel paziente (figura 15), avendo anche la possibilità di aggiornare le varie informazioni se necessario. Per poter

completare la registrazione del paziente, è necessario che quest'ultimo effettui il primo accesso da web app utilizzando le credenziali appena inserite.

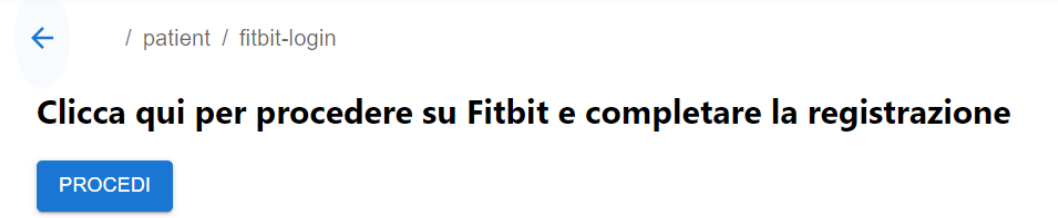

Figura 16 - patient Home Page

Dopo aver effettuato l'accesso, il paziente verrà invitato a completare la registrazione collegando il proprio account Health App a quello di Fitbit. Per farlo, l'utente dovrà cliccare su 'Procedi', quindi effettuare il login su Fitbit, accettando tutti i consensi necessari per la raccolta dei dati biometrici lato Health App. Una volta conclusa questa procedura, la registrazione sarà completata con successo, consentendo al paziente di accedere alla propria area personale tramite l'applicazione mobile.

2.3. CREAZIONE DI UN QUESTIONARIO

Al fine di ottenere una visione completa della routine quotidiana dei pazienti, è importante raccogliere anche dati di diversa natura rispetto a quelli ottenuti tramite il tracker, come le loro abitudini alimentari e altri aspetti rilevanti. Si è quindi deciso, in accordo con i medici, di mettere a disposizione dei partecipanti all'esperimento dei questionari, in modo da acquisire ulteriori informazioni preziose per valutare il loro stile di vita e il loro stato di salute in modo approfondito. Nello specifico, abbiamo focalizzato la nostra attenzione su questionari derivanti dalla letteratura scientifica sulla salute, come il MEDIAS, uno strumento molto utilizzato per valutare quanto una persona segua la dieta mediterranea.

Tuttavia, abbiamo riconosciuto che i questionari standard come il MEDAS potrebbero risultare limitanti, poiché molto generici e non sempre adatti al nostro specifico contesto sperimentale. Pertanto, è stata sviluppata una funzionalità che permette di creare questionari personalizzati, utilizzando il MEDAS come base e punto di riferimento. Questo approccio consente di definire questionari su misura per affrontare diverse tematiche, consentendo ai pazienti di condividere informazioni più accurate sul loro stile di vita.

Inoltre, per assicurare che gli utenti finali si sentano a loro agio, abbiamo cercato di emulare l'esperienza cartacea sia durante la creazione che durante il completamento dei questionari.

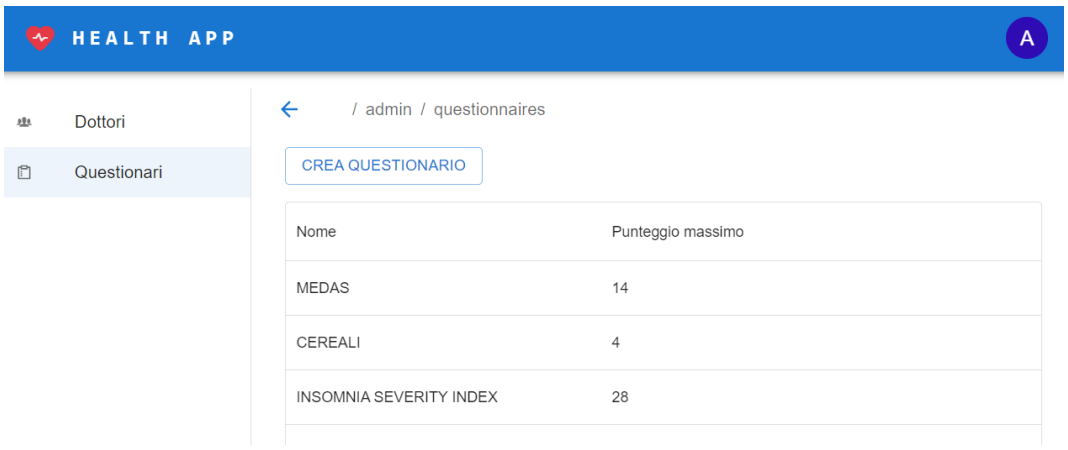

Figura 17 - Lista dei questionari compilabili

Lato amministratore, è possibile consultare l'elenco dei questionari esistenti o crearne di nuovi. Per far ciò è sufficiente fare clic sul pulsante 'Crea Questionario'.

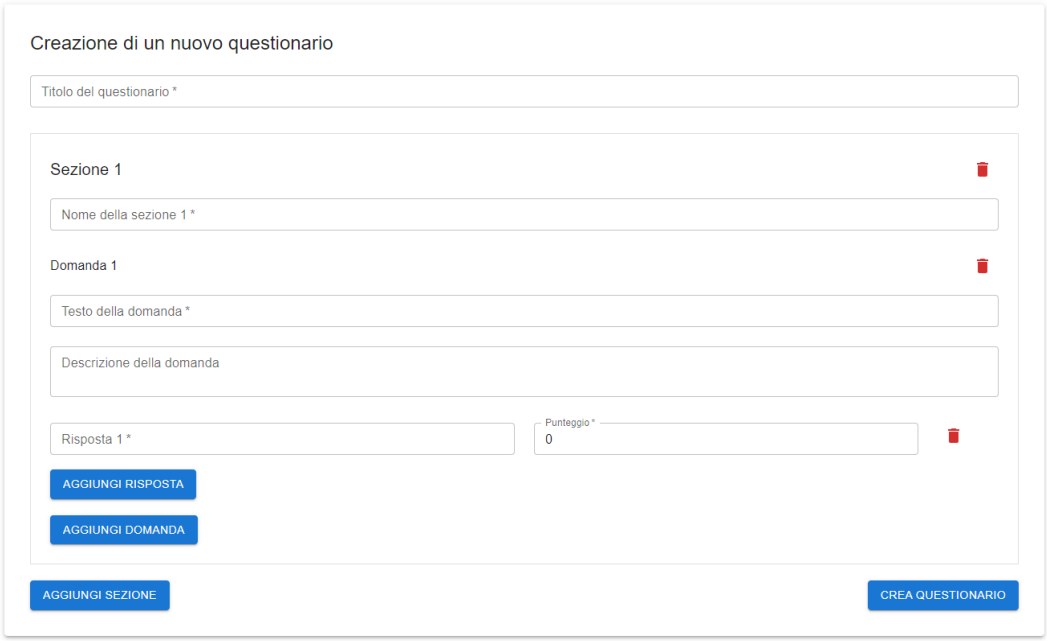

Figura 18 - creazione di un nuovo questionario

Il form per la creazione di un nuovo questionario è altamente intuitivo poiché mostra in una sola pagina tutti gli step necessari per completare il task con successo. Ciascun questionario deve avere un titolo obbligatorio e può contenere diverse domande. Poiché le domande possono riguardare tematiche differenti, si è pensato di raggrupparle in sezioni. Ogni sezione è dedicata a una specifica area di interesse.

Figura 19 - inserimento di una domanda con possibili risposte associate

Per ogni domanda è necessario fornire le possibili risposte, ognuna delle quali deve essere associata a un punteggio corrispondente (figura 19). Dopo aver salvato il questionario, è possibile consultarlo nella sezione 'Questionari' del pannello amministratore, dove è disponibile un riepilogo di quanto è stato inserito.

Domanda	Descrizione	Sezione	Risposte	Punti
Quanti cucchiai di olio d'oliva consumi al giorno?	Compreso l'olio d'oliva che usi per cucinare, per condire l'insalata, che hai consumato nei pasti fuori casa, ecc.	Sezione 1	Uno o meno di uno Due o tre Quattro o più	0 0 1
Quante porzioni di verdura consumi al giorno?	Sono comprese tutte le verdure cotte e crude escluse le patate e i legumi. Una porzione = una tazza grande o metà piatto grande; Una porzione = 200 g. Considera i contorni come 1/2 porzione.	Sezione 1	Meno di una Una Due Tre o più	0 0 1 1
Quante porzioni di frutta fresca consumi al giorno?	Una porzione = una unità di frutta di medie dimensioni, una grande tazza di frutta a fette, una fetta di melone o anguria di medie dimensioni, o una tazza di succo appena spremuto.	Sezione 1	Meno di una Una Due Tre o più	0 0 0 1

Figura 20 - resoconto del questionario appena creato

2.4. COMPILAZIONE DI UN QUESTIONARIO

Come accennato in precedenza, la compilazione dei questionari fornisce ai medici ulteriori informazioni sulle abitudini e lo stato di salute dei pazienti, che si aggiungono ai dati raccolti tramite il dispositivo Fitbit.

Questa attività è di fondamentale importanza per il successo complessivo dell'esperimento. Di solito, la compilazione avviene durante gli incontri periodici tra il medico e il paziente, in modo che possano affrontare insieme il processo. Questa collaborazione consente loro di chiarire dubbi e evitare possibili fraintendimenti, poiché l'obiettivo è ottenere un quadro chiaro e completo del livello di benessere del paziente. Quindi, a partire dalla home page del medico, si seleziona il paziente con cui si vuole compilare il questionario.

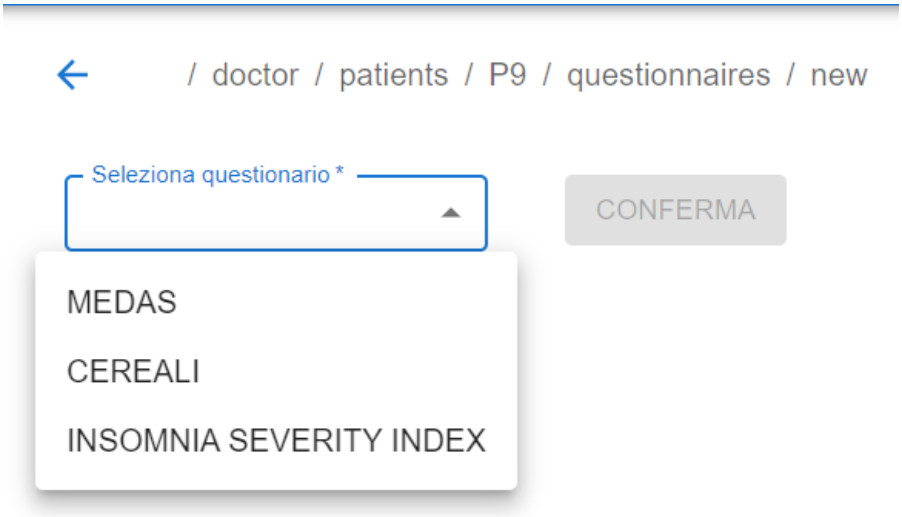

Figura 21 - selector per la scelta del questionario da compilare

In seguito, è necessario scegliere il questionario che si desidera compilare (come illustrato nella figura 21). La selezione avviene tra una lista di questionari precedentemente definiti nell'area amministratore, creati avendo il questionario MEDAS come modello di riferimento.

Figura 22 - esempio di questionario da compilare

Ciascun questionario si compone di una serie di domande a risposta chiusa, ognuna delle quali offre opzioni rappresentate da caselle da spuntare (come evidenziato nella figura 22). Questo schema richiama l'idea di un questionario tradizionale con crocette su carta, enfatizzando la sua immediatezza e semplicità d'utilizzo.

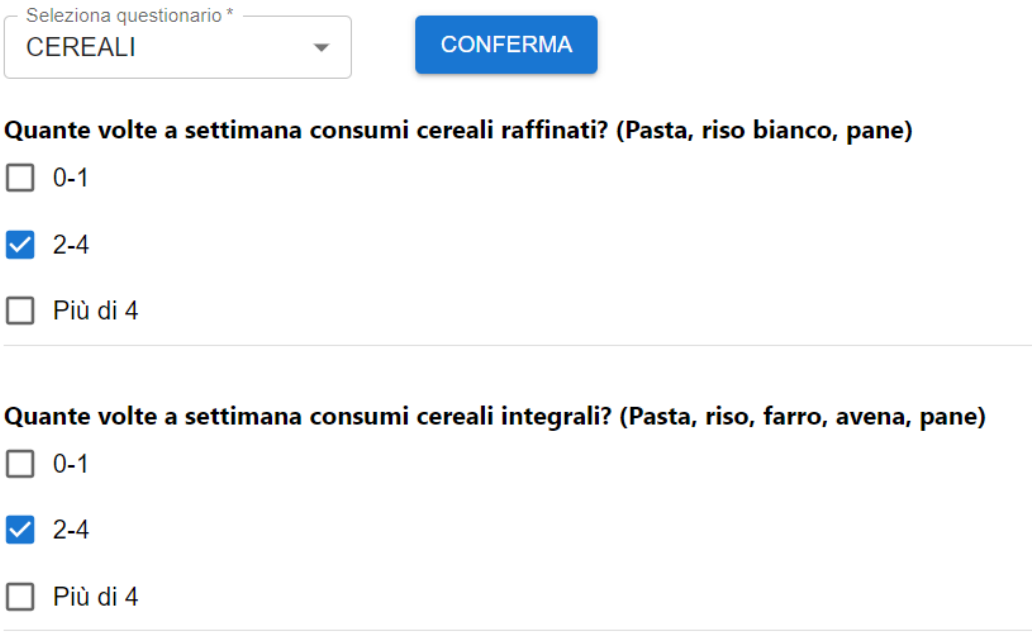

Figura 23 - esempio di questionario compilato

Il questionario può essere confermato solo dopo aver scelto una risposta per ciascuna domanda. In queste circostanze, il pulsante 'Conferma' (come illustrato nella figura 23) diventa cliccabile.

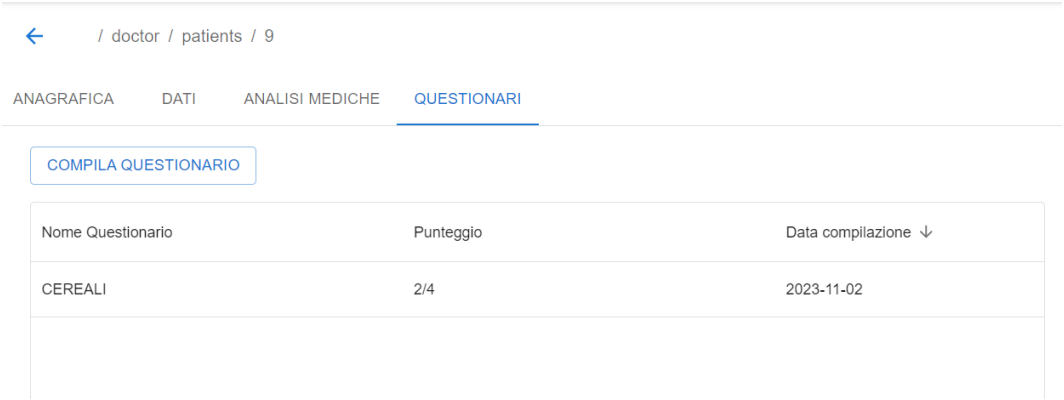

Figura 24 - elenco dei questionari compilati dal paziente

Quindi, il questionario appena completato verrà visualizzato nell'elenco dei questionari compilati dal paziente selezionato (come indicato nella figura 24). In particolare, verranno mostrati il nome del questionario, il punteggio totale ottenuto e la data di compilazione.

Domanda	Risposte	Punti	Risposta data	Punti risposta
Quante volte a settimana consumi cereali raffinati? (Pasta, riso bianco, pane)	0-1 2-4 Più di 4	2 1 0	2-4	1
Quante volte a settimana consumi cereali integrali? (Pasta, riso, farro, avena, pane)	0-1 2-4 Più di 4	0 1 2	2-4	1

Nome: CEREALI

Punteggio: 2/4

Figura 25 - resoconto del questionario compilato

Se si desidera ottenere un riepilogo più approfondito, è possibile fare clic sul questionario di interesse e visualizzare i dettagli, inclusa la risposta fornita a ciascuna domanda e il punteggio ottenuto per ciascuna di esse (come mostrato nella figura 25). È fondamentale evidenziare che un questionario può essere compilato più volte dallo stesso paziente durante il corso dell'esperimento. Questo consente ai medici di monitorare i progressi dei pazienti e di osservare come le loro abitudini cambino nel periodo preso in esame.

2.5. VISUALIZZAZIONE DEI PARAMETRI BIOMETRICI

Uno dei passaggi cruciali per un paziente in fase di registrazione è quello di collegare ad Health App il proprio account Fitbit, in maniera tale da rendere fruibili anche sulla nostra app i dati raccolti dal tracker.

A primo impatto questa scelta potrebbe quasi sembrare ridondante, ma in realtà rappresenta una rielaborazione dei dati stessi, al fine di offrire ai medici una visione più approfondita della salute dei pazienti e abilitarli a offrire consigli personalizzati per migliorare il benessere dei loro assistiti. Per consultare i parametri biometrici di un paziente specifico tramite la piattaforma Health App, il medico deve inizialmente selezionare il paziente desiderato.

Successivamente, può esplorare la sezione 'Dati', suddivisa a sua volta in quattro sottosezioni, una per ciascuna area di interesse:

- **Passi**: questa sottosezione visualizza il numero di passi compiuti dal paziente in un determinato intervallo di tempo.
- **RHR (Frequenza Cardiaca a Riposo)**: rappresenta il numero di battiti cardiaci registrati in un arco di tempo specifico.
- **HR Zone (Zone di Frequenza Cardiaca)**: questa parte fornisce un'indicazione pratica delle diverse zone di frequenza cardiaca in cui il paziente si trova durante l'attività fisica.
- **Sonno**: questi grafici presentano dati sulla durata totale del sonno, ponendo una particolare attenzione sulle diverse fasi che lo contraddistinguono e gli eventuali risvegli notturni. Questi dati sono essenziali per valutare la qualità del sonno del paziente.

Queste quattro sottosezioni offrono un quadro completo dei parametri biometrici del paziente, consentendo al medico di valutare diversi aspetti della sua salute e benessere.

Figura 26 - vista giornaliera

Poiché per i medici è importante osservare l'evoluzione dei parametri nel corso del tempo, la piattaforma offre la flessibilità di visualizzare questi dati sia su base giornaliera (come illustrato nella figura 26) che settimanale (come mostrato nella figura 27), consentendo agli utenti di spostarsi liberamente da una data all'altra grazie alla presenza di un date picker.

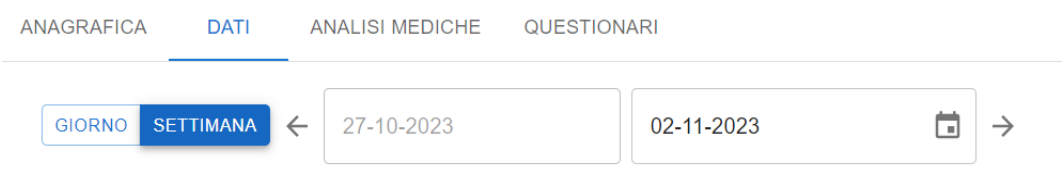

Figura 27 - vista settimanale

I dati sono visualizzati tramite grafici a barre, in cui i valori lungo l'asse verticale rappresentano misurazioni specifiche legate all'argomento di interesse selezionato, mentre lungo l'asse orizzontale è indicato l'intervallo temporale. Ogni grafico comprende inoltre un valore medio, che si rivela prezioso per il medico nell'individuare tendenze o variazioni significative nel periodo preso in esame.

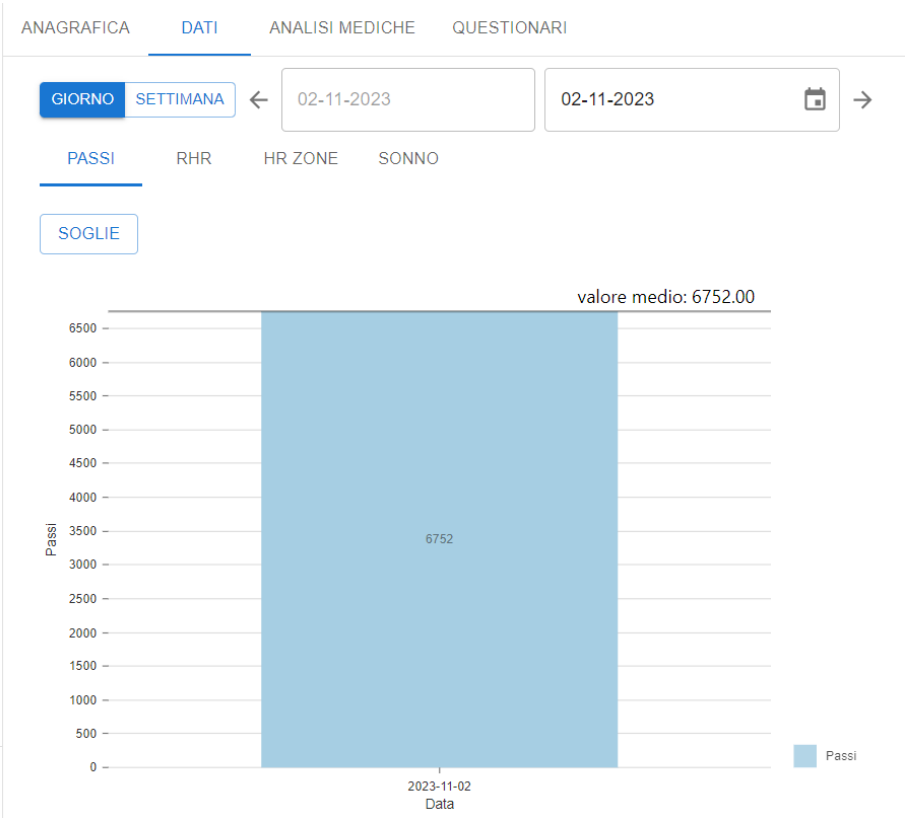

Figura 28 - grafico che mostra i passi effettuati in un singolo giorno

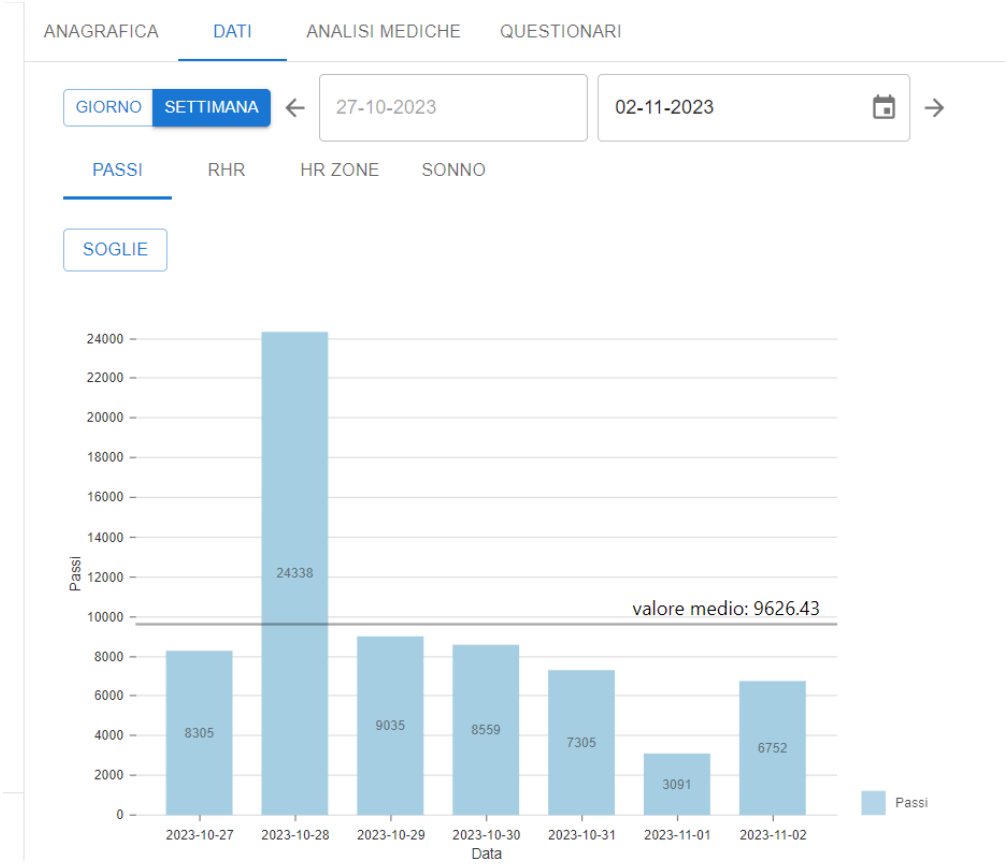

Figura 29 - grafico che illustra i passi effettuati in una settimana

Figura 30 - grafico che mostra i BPM a riposo giornalieri

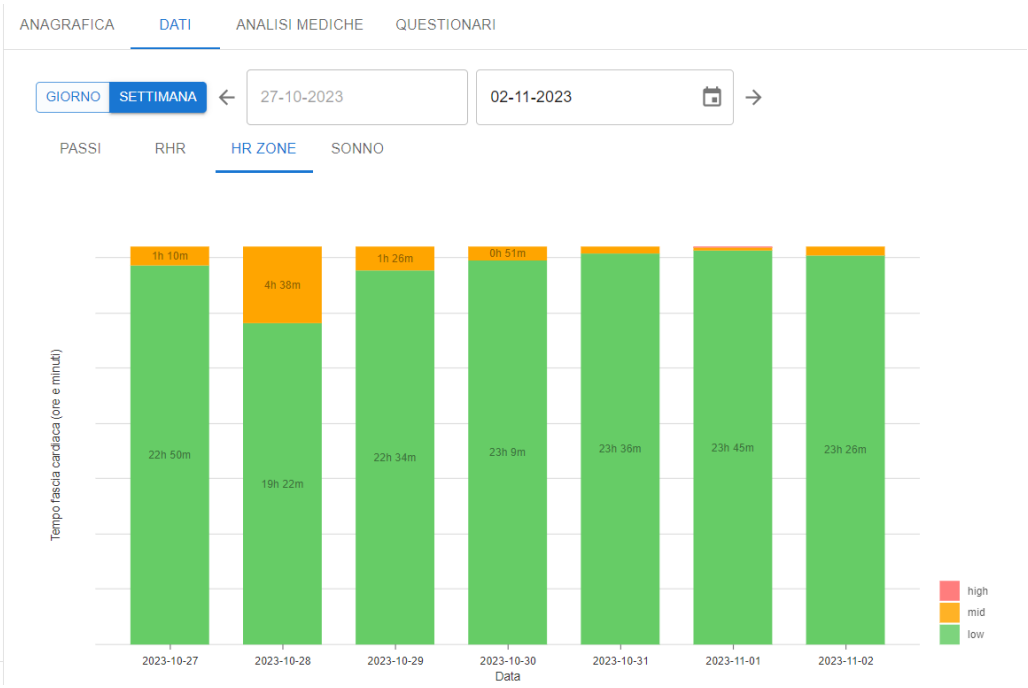

Figura 31 - grafico che mostra le zone di frequenza cardiaca settimanali

In merito alla frequenza cardiaca, si è deciso di rappresentarla mediante un'unica barra, differenziando le diverse fasce mediante colori distinti, mostrando la quantità complessiva di tempo trascorso (espresso in ore) in ogni fase durante la giornata.

Le tre fasi individuate sono le seguenti:

- **Fase di Riposo**: in questa fase, la frequenza cardiaca è inferiore al 60% del valore massimo registrato. Essa è comunemente associata a momenti di riposo o ad attività fisiche leggere.
- **Fase di Attività Moderata**: qui, la frequenza cardiaca si colloca in un intervallo compreso tra il 60% e l'80% del valore massimo registrato. Questa fascia indica periodi di attività fisica a intensità medio-bassa.
- **Fase di Attività Intensa**: questa fase comprende tutte le letture della frequenza cardiaca che superano l'80% del valore massimo registrato. Rappresenta momenti di intensa attività cardiaca, come l'esercizio fisico vigoroso o momenti di stress elevato.

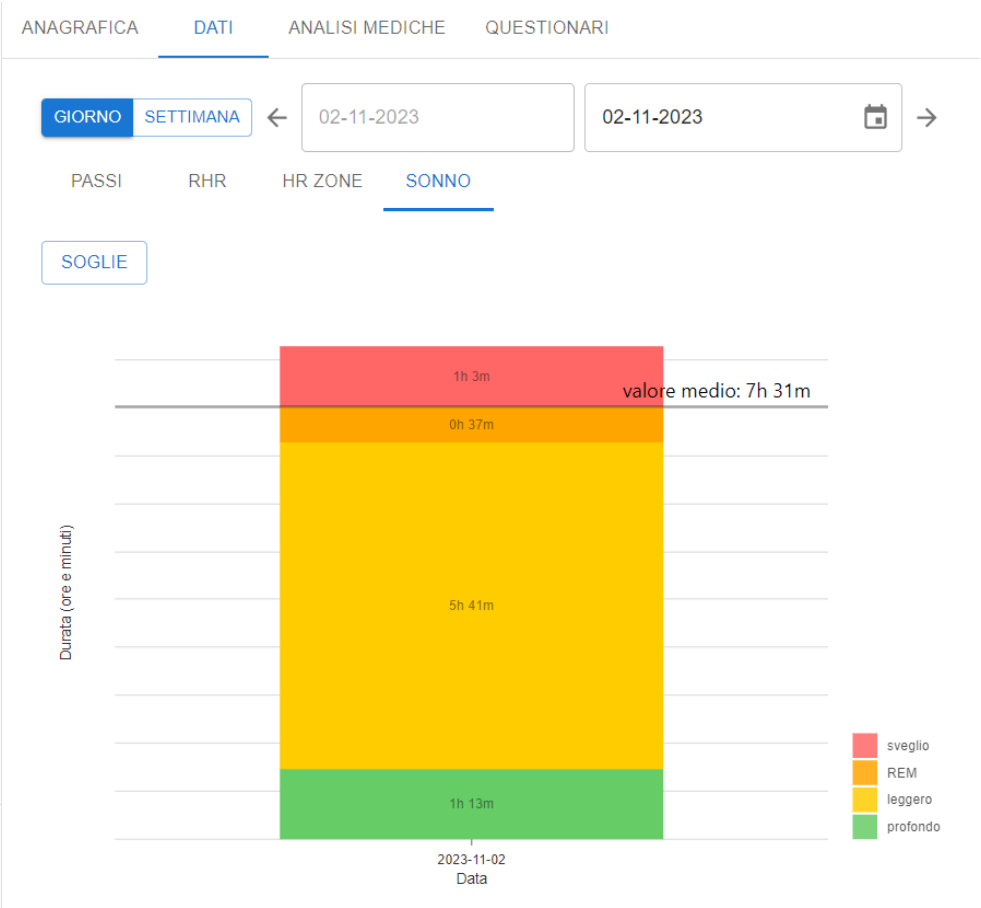

Figura 32 - grafico che mostra le fai del sonno di un singolo giorno

Per quanto riguarda il monitoraggio del sonno, abbiamo adottato un approccio simile a quello utilizzato per la rappresentazione della frequenza cardiaca. Tuttavia, in questo caso, le fasi sono quattro:

- **Fase di Rilassamento**: questa fase rappresenta il momento in cui la persona è ancora sveglia e si sta rilassando prima di addormentarsi.
- **Fase di Sonno Leggero**: qui si rileva il sonno in una fase meno profonda, caratterizzata da una maggiore sensibilità agli stimoli esterni.
- **Fase di Sonno Profondo**: questa fase indica un sonno più profondo e riposante, in cui la persona è meno suscettibile agli input esterni.
- **Fase REM**: La fase REM [28] è la fase di sonno più profondo e coinvolge movimenti oculari rapidi. Durante questa fase, solitamente si verificano i sogni.

Questo approccio fornisce una chiara rappresentazione delle diverse fasi del sonno, aiutando a comprendere nel dettaglio il profilo del sonno del paziente.

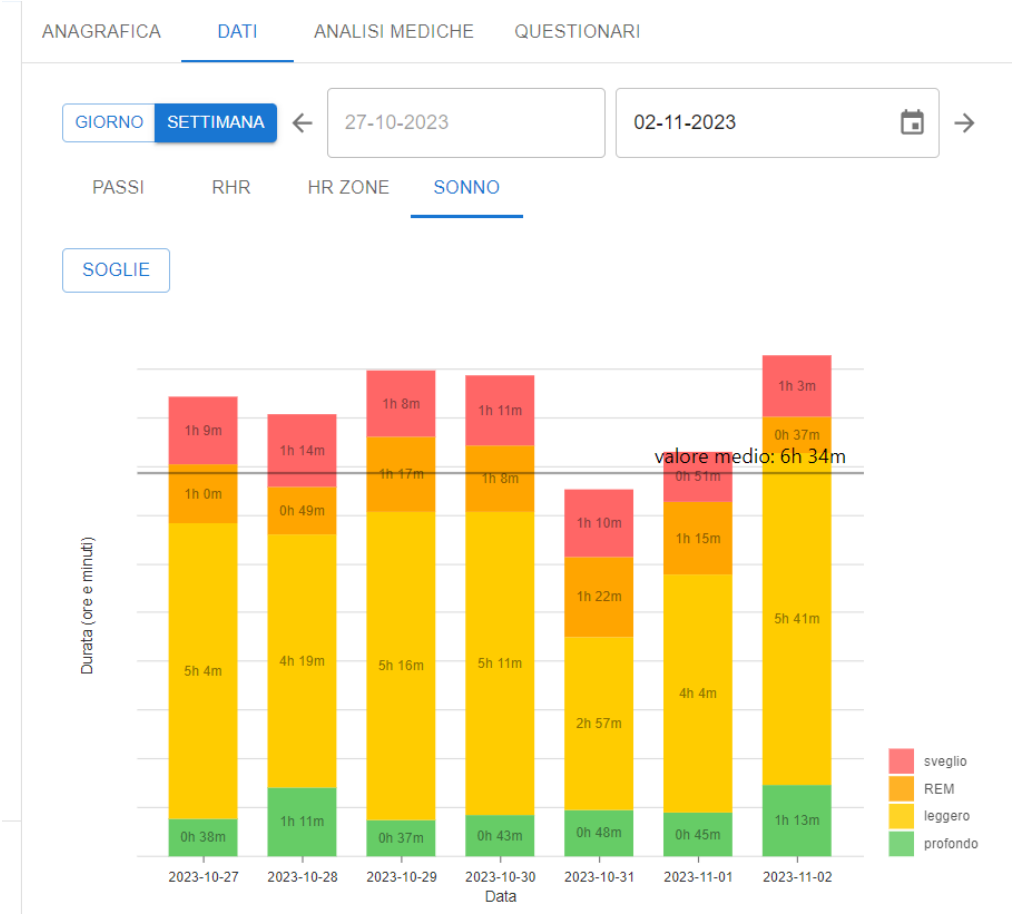

Figura 33 - grafico che mostra le fasi del sonno selezionando un range settimanale

2.6. INSERIMENTO E CONSULTAZIONE DELLE ANALISI MEDICHE

Durante le visite di routine, il medico ha la facoltà di richiedere al paziente di sottoporsi a specifici esami del sangue, i cui risultati vengono poi registrati su Health App, per mantenere un riferimento in vista delle analisi successive. Inoltre, in ogni incontro, è possibile aggiornare i dati relativi al peso del paziente. Il medico può annotare il peso attuale, con la data predefinita impostata sulla data dell'incontro, ma ha anche la possibilità di modificarla manualmente tramite un date picker se i dati si riferiscono a date diverse. Lo stesso vale per la registrazione dei referti delle analisi del sangue.

Questa flessibilità garantisce un monitoraggio dettagliato e preciso dell'evoluzione del peso e dei valori sanguigni del paziente nel corso del tempo. L'insieme di tutti questi dati permette al medico di valutare i progressi del paziente durante l'esperimento, permettendogli di determinare se le strategie di intervento raccomandate hanno effettivamente prodotto gli effetti desiderati, contribuendo così a migliorare il benessere del paziente.

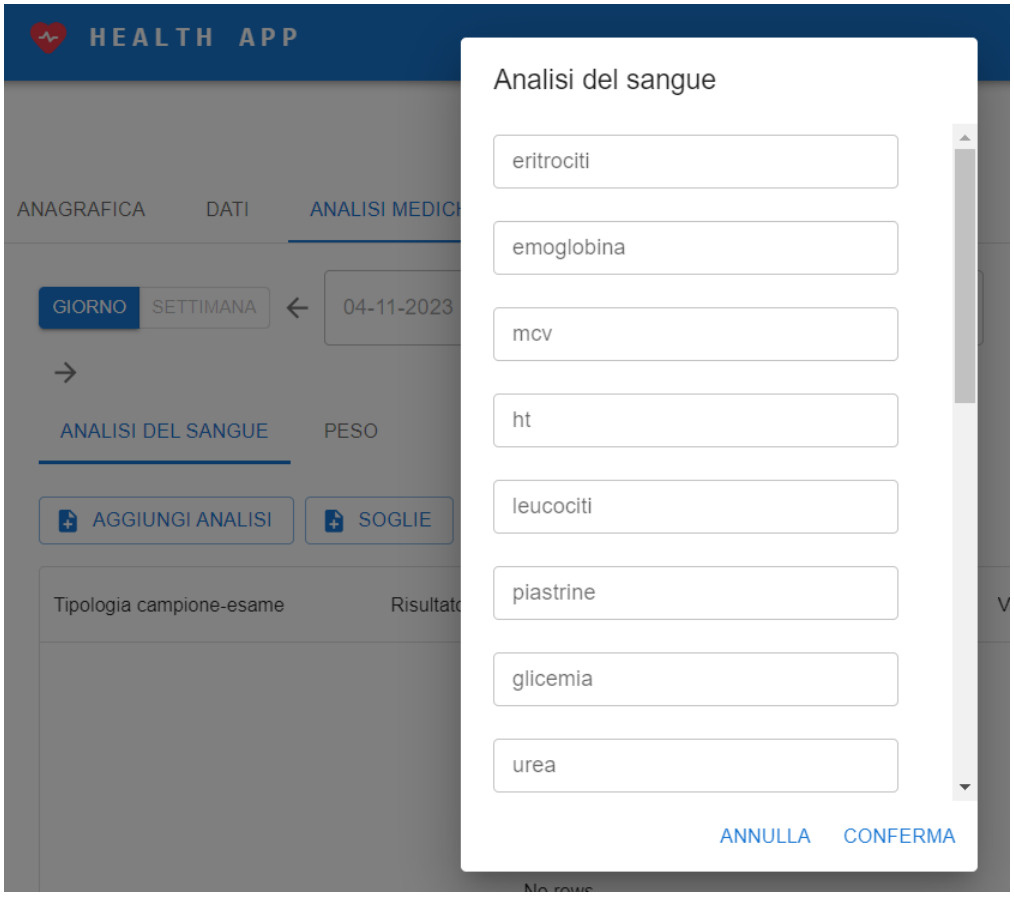

Figura 34 - schermata di inserimento dei valori degli esami sanguigni

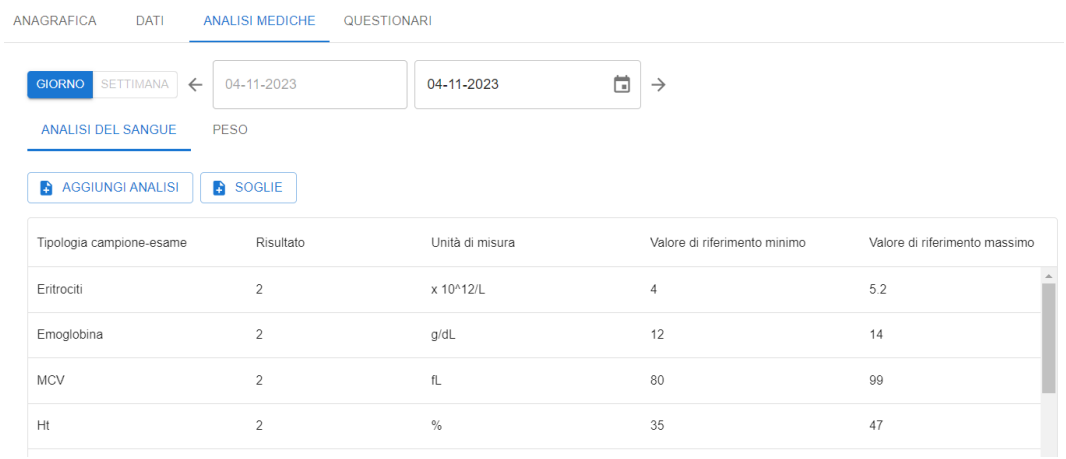

Figura 35 - visualizzazione dei dati delle analisi del sangue

Per agevolare la consultazione dei dati, vengono forniti dettagli per ciascun parametro sanguigno, compresi l'unità di misura, l'intervallo di valori di riferimento (minimo e massimo) inseriti come soglie e il risultato specifico dell'analisi (figura 35).

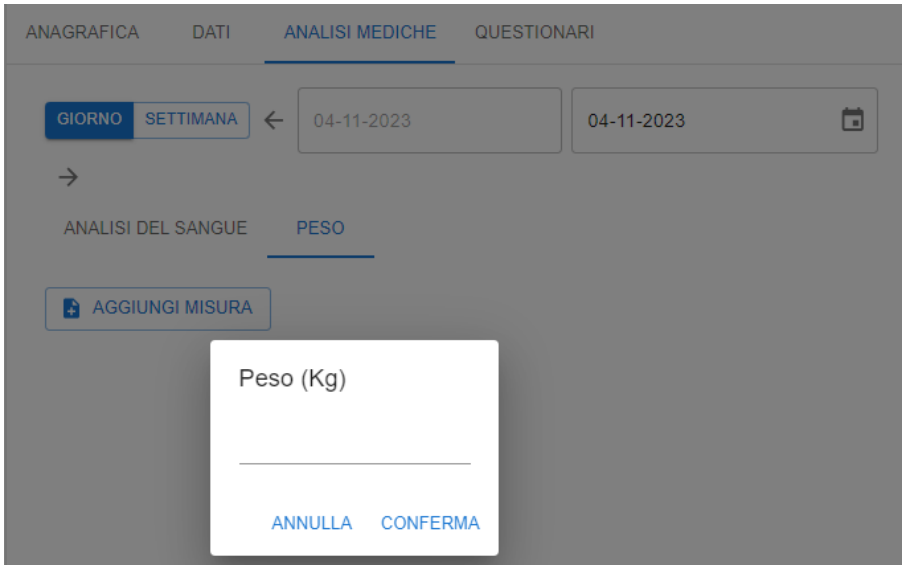

Figura 36 - inserimento del peso per un dato giorno

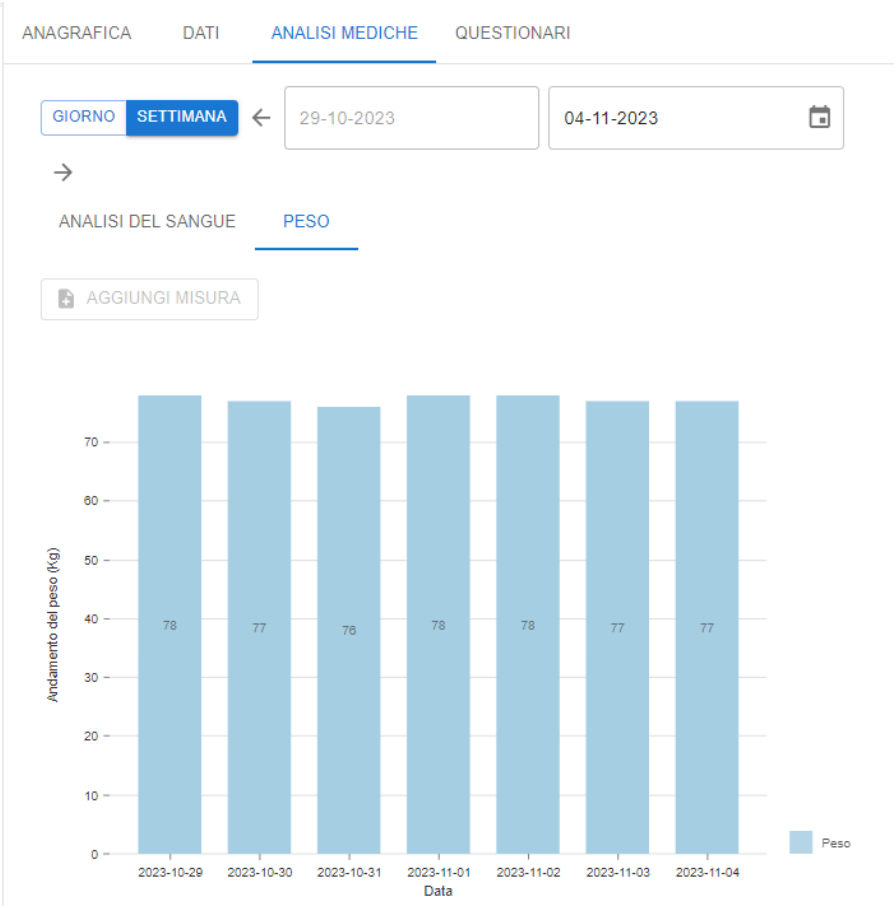

Figura 37 - visualizzazione dell'andamento settimanale del peso

Il grafico evidenzia chiaramente le variazioni del peso giornaliero del paziente, fornendo un'istantanea dei cambiamenti nel corso della settimana. Questa analisi dettagliata è fondamentale per il monitoraggio del progresso verso gli obiettivi di gestione del peso prefissati.

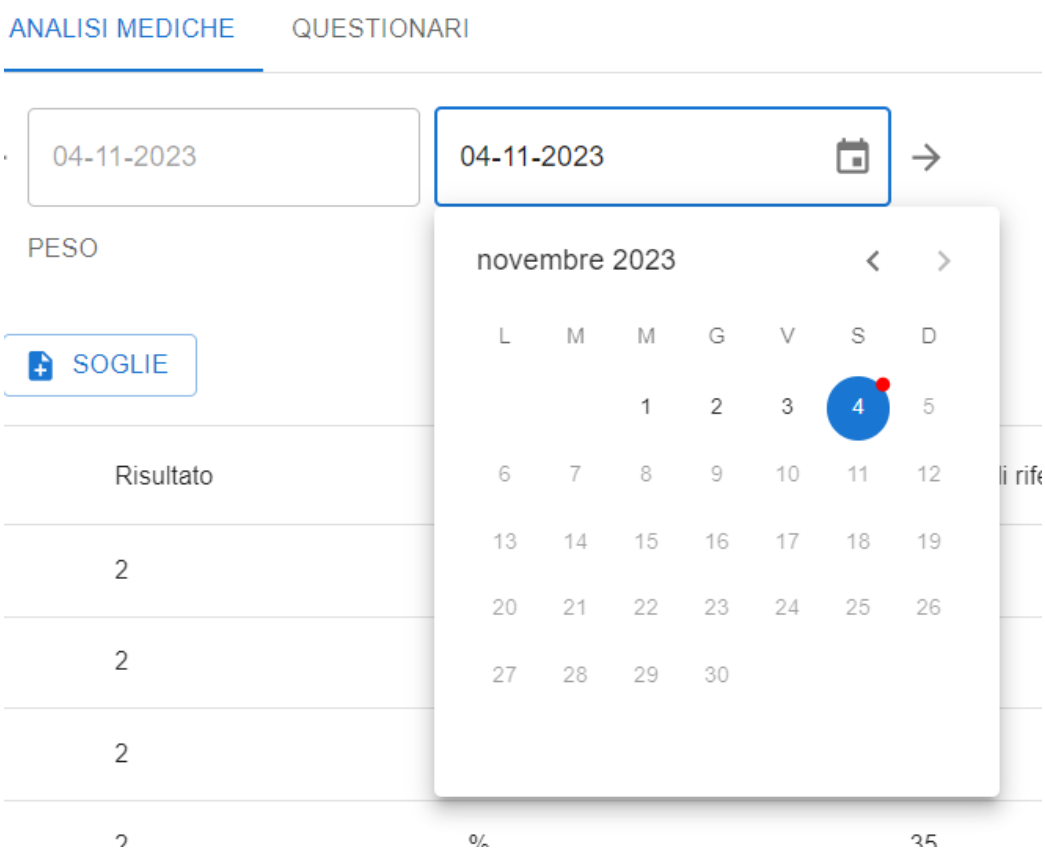

Figura 38 - animazione nel calendario che sottolinea la presenza di un evento in tale data

Dato che l'esperimento si estende su un ampio arco temporale, è importante mettere in evidenza i giorni che contengono informazioni degne di nota, come la registrazione dei referti medici. Per agevolare quindi la navigazione nell'applicazione da parte dei medici, i giorni con esami clinici da consultare sono contrassegnati da un distitivo pallino rosso.

2.7. INSERIMENTO E CONSULTAZIONE DEI VALORI DI SOGLIA

Sinora, abbiamo spesso sottolineato come uno degli obiettivi dell'esperimento sia quello di offrire consigli personalizzati per migliorare lo stato di salute di ciascun paziente. In virtù di ciò, lato backend è stato integrato un Rule Engine che, tramite l'applicazione di alcune regole definite dai medici, è in grado di definire lo stato di salute del paziente.

I medici osservano una vasta gamma di dati, dai monitoraggi biometrici forniti dai tracker fino ai dati relativi ai parametri sanguigni inseriti nel sistema. Per ogni parametro, essi stabiliscono delle soglie o intervalli di valori ideali che dovrebbero essere rispettati. Queste regole consentono al sistema di valutare e determinare lo stato di salute del paziente, garantendo un trattamento personalizzato per migliorare il loro stato di salute.

Per configurare queste regole, i medici selezionano un paziente e dall'apposita area dedicata inseriscono i valori desiderati per ciascuna sezione.

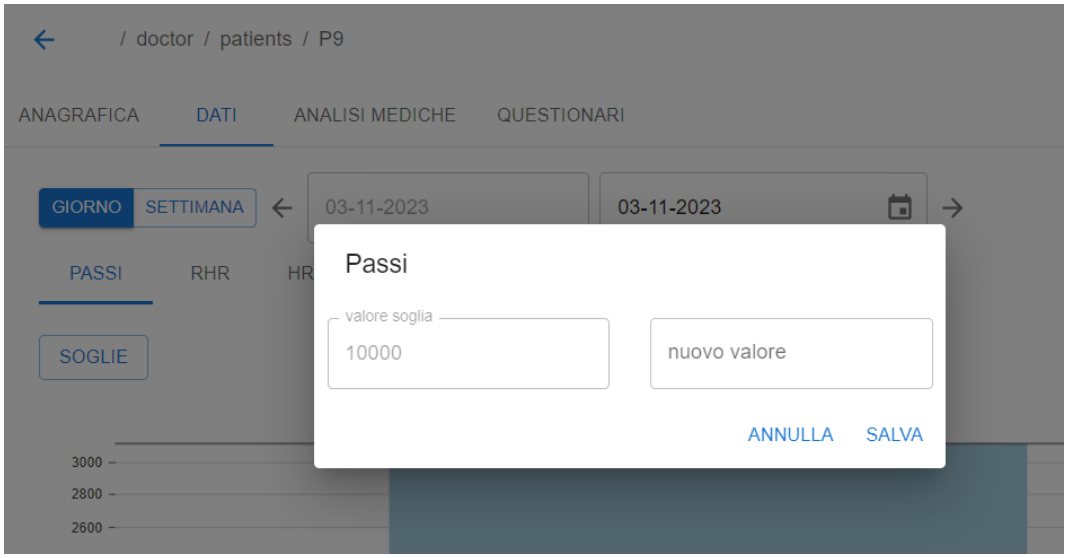

Figura 39 - inserimento del valore minimo di passi da fare giornalmente

Come illustrato in figura 39, per quanto riguarda il conteggio dei passi è necessario inserire il numero minimo di passi giornalieri richiesti.

Figura 40 - inserimento dei valori di soglia per la frequenza cardiaca a riposo

Mentre, relativamente ai battiti al minuto (bpm) medi, è richiesto di definire un intervallo, indicando i valori minimi e massimi consentiti.

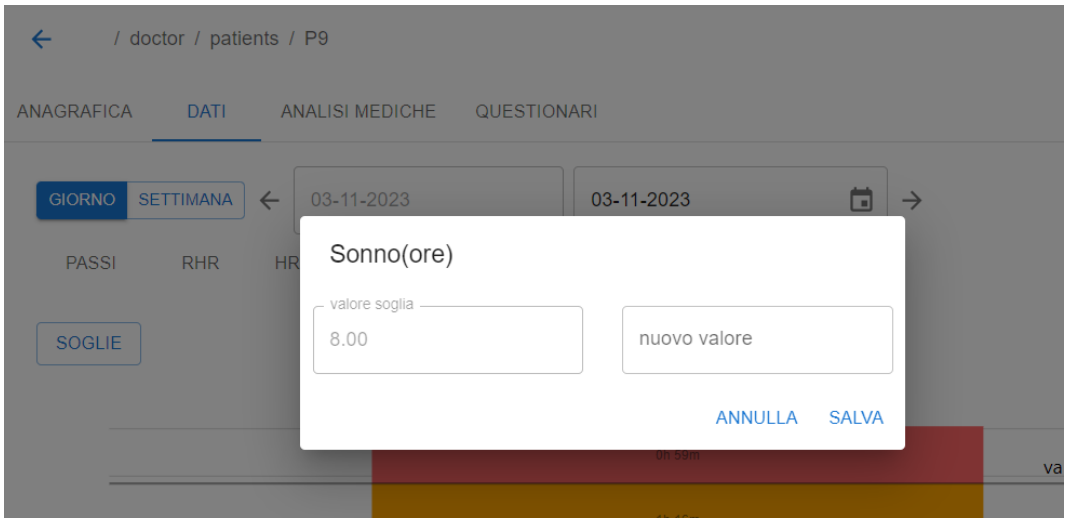

Figura 41 - inserimento del numero minimo di ore di sonno consigliate

Figura 42 - valori di soglia per i parametri sanguigni

L'uso delle soglie si applica anche alle analisi del sangue e, come per i bpm medi, le regole utilizzate nel sistema di raccomandazioni richiedono la definizione di una coppia di valori, rispettivamente un minimo e un massimo consentito.

Questi parametri di riferimento sono poi resi accessibili durante la consultazione delle analisi mediche precedentemente registrate nel sistema, come descritto nel paragrafo precedente.

3. TECNOLOGIE UTILIZZATE IN FASE DI IMPLEMENTAZIONE

In questo capitolo si esamineranno le tecnologie utilizzate in fase di implementazione.

3.1. REACT

Come menzionato in precedenza, la web application è stata realizzata utilizzando React [29], una famosa libreria open-source javascript.

Grazie alla sua struttura modulare basata su componenti e all'approccio dichiarativo, React favorisce la riutilizzabilità del codice, apportando notevoli semplificazioni nella gestione e manutenzione di interfacce utente.

3.1.1 VIRTUAL DOM

Il Virtual DOM (Document Object Model) costituisce il principio cardine alla base della strategia di rendering di React. Il DOM convenzionale presenta un'organizzazione che ricalca la struttura gerarchica di un documento HTML o XML, favorendo l'interazione tra il browser e i diversi elementi della pagina web. Questa gerarchia viene interpretata dal DOM come un albero di oggetti, in cui ciascun nodo rappresenta uno specifico elemento della UI. A differenza di ciò che avviene tradizionalmente, React crea una sua rappresentazione virtuale del DOM, piuttosto che manipolare direttamente i componenti nativi del DOM originale ogni volta che vengono riscontrati dei cambiamenti. Questa rappresentazione virtuale viene confrontata con la versione effettiva del DOM, applicando solo le modifiche necessarie. Questa operazione è detta 'Riconciliazione' [30].

Seguendo l'approccio tradizionale, per ogni modifica all'albero dei componenti sarebbe necessario costruirne uno ex novo. Tuttavia, questo approccio richiederebbe l'utilizzo di algoritmi con una complessità nell'ordine di O(n3), dove n rappresenta il numero di nodi dell'albero. L'algoritmo di Riconciliazione utilizzato da React, invece, ha una complessità nell'odine di O(n) grazie a due assunzioni:

- Elementi di diversa natura generano alberi differenti.
- L'utilizzo di una prop "key", assegnata ad ogni elemento dell'albero, permette all'algoritmo di valutare suggerimenti in merito a quali elementi potrebbero non subire variazioni tra una renderizzazione e l'altra.

Infatti, confrontando due alberi, l'algoritmo si concentra prima sui due nodi radice, differenziando l'azione scelta a seconda del loro tipo. Se questi ultimi non appartegono alla stessa tipologia di elemento, React abbatte il vecchio albero costruendone uno interamente nuovo. Ogni componente che sta sotto la radice verrà perciò smontato e il suo state distrutto. Invece, nel confrontare due elementi del Virtual DOM che hanno lo stesso tipo, React compara gli attributi di entrambi, mantenendo lo stesso nodo sottostante ed aggiornando solo gli attributi cambiati.

```
<div className="prima" title="cose" />

<div className="dopo" title="cose" />
```

Figura 43 - modifica di un parametro di un componente

Ad esempio, come illustrato nella figura 43, nel confronto tra questi due elementi, React sa di dover modificare solo il valore del parametro className. In seguito, quando il componente viene renderizzato l'algoritmo di confronto va in ricorsione sui nodi figli.

Di base, quando React va in ricorsione sui figli di un nodo DOM, opera su entrambe le liste (pre e post modifica) contemporaneamente e genera una mutazione ogni volta che trova delle differenze. In questi casi, per rendere efficiente la conversione tra alberi, React supporta un attributo key, che l'algoritmo di riconciliazione utilizza per confrontare i figli nell'albero originale con quelli nell'albero successivo.

```
<ul>
  <li key="2015">Duke</li>
  <li key="2016">Villanova</li>
</ul>

<ul>
  <li key="2014">Connecticut</li>
  <li key="2015">Duke</li>
  <li key="2016">Villanova</li>
</ul>
```

Figura 44 - modifica del numero di componenti figli

Questo nuovo approccio consente di ottimizzare le operazioni di aggiornamento e, di conseguenza, le performance complessive dell'applicazione.

3.1.2 JSX

Per quanto riguarda la progettazione dell'interfaccia utente, in React non è necessario creare un componente che abbia file HTML e CSS associati. Infatti, essa fa uso del formato JSX, un'estensione della sintassi JavaScript. Il principio chiave alla base di esso è che React riconosce che la logica di renderizzazione è intrinsecamente collegata a quelle che gestiscono l'interfaccia utente, come la gestione degli eventi, l'evoluzione dello stato nel tempo e la preparazione dei dati per la visualizzazione.

Quindi, invece di attuare una separazione sia fisica che concettuale inserendo il codice di markup e la logica in file separati, React utilizza i componenti come unità debolmente accoppiate che li incorporano entrambi, consentendo una gestione più coesa e organizzata delle responsabilità all'interno dell'applicazione.

```
const name = 'Giuseppe Verdi';
const element = <h1>Hello, {name}</h1>;
```

Figura 45 - esempio di JSX

In fase di sviluppo sono state inoltre utilizzate alcune librerie chiave per migliorare la qualità del progetto:

- Redux, per gestire in modo efficiente lo stato dell'applicazione, assicurando un'organizzazione chiara dei dati.

- Material-UI, per creare un'interfaccia utente intuitiva e attraente, che non solo soddisfi gli standard estetici, ma migliori anche l'usabilità complessiva.
- Axios, per effettuare richieste HTTP da un client web a un server.

3.2. REACT-REDUX

React è una libreria che si focalizza sull'interfaccia grafica, rendendosi agilmente integrabile con altre librerie. Redux [17], nello specifico, è un framework che si concentra sulla gestione dello stato globale dell'applicazione, avendo come obiettivo quello di rendere prevedibili le transizioni di stato definendo come e quando questo può avvenire.

L'utilizzo di Redux [29] offre diversi benefici in contesti complessi che richiedono uno stato condiviso tra diversi componenti. Esso consente di centralizzare lo stato dell'applicazione in un unico "store" globale, accessibile e modificabile da qualsiasi componente all'interno dell'applicazione.

Redux [18] offre diversi vantaggi, tra cui:

- Gestione centralizzata dello stato: Redux offre la possibilità di mantenere uno stato centralizzato dell'applicazione, a cui possono accedere tutti i componenti.
- Facilità di debugging: Redux permette di tenere traccia di tutte le azioni eseguite nell'applicazione, agevolando il processo di debug.
- Scalabilità: grazie alla presenza di actions e reducers, Redux semplifica notevolmente la gestione di applicazioni con una grande quantità di dati e componenti.
- Azioni reversibili: tutte le azioni sono annullabili e ripetibili.
- Flessibilità: è una libreria javascript utilizzabile non solo in combinazione con React, ma anche ad altri framework tipo Angular o, semplicemente, a javascript mediante vanilla js. Questa flessibilità consente una maggiore adattabilità a diversi contesti e tecnologie.

Tuttavia, la natura funzionale di React, insieme alla sua struttura dichiarativa, facilita l'integrazione con Redux. Quest'ultimo offre un modo per disaccoppiare lo stato dell'applicazione da React, creando un archivio globale indipendente che gestisce lo stato di tutti i componenti.

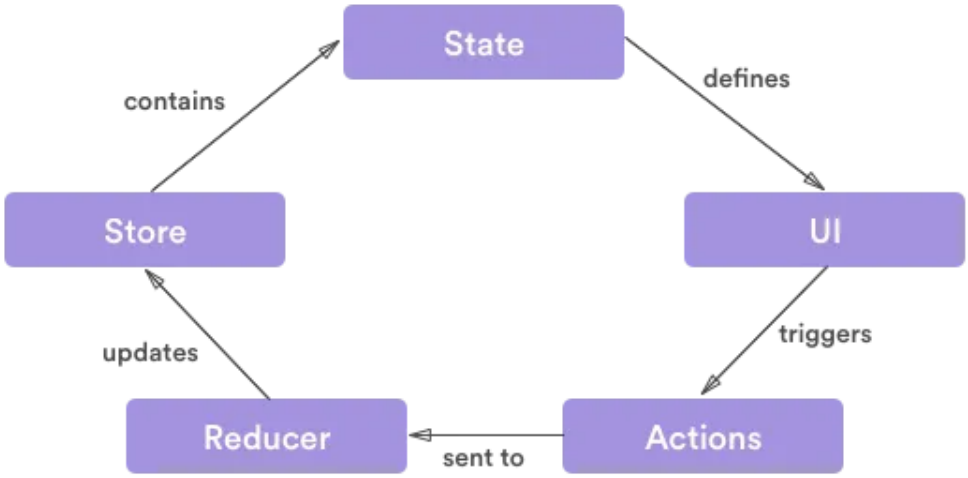

Figura 46 - React-Redux flow. (leggere da destra verso sinistra, partendo da 'UI')

Redux si fonda su un assunto: il mantenimento di un unico stato globale rappresentato da un oggetto JSON, archiviato in uno store. Quando l'utente interagisce con l'interfaccia utente compiendo delle azioni, scatena delle actions. Queste actions sono poi catturate dai reducers, i quali sono responsabili di apportare modifiche allo stato globale. Una volta che lo stato viene aggiornato, l'interfaccia utente si sincronizza automaticamente per riflettere tali cambiamenti. Inoltre, Redux si basa su tre principi:

- **Unica fonte di verità**: lo stato è l'unica fonte di verità a cui fa riferimento l'interfaccia utente.
- **Stato read-only**: lo stato è per definizione read-only e viene modificato dal reducer solo in seguito ad un'azione.
- **Cambiamenti di stato tramite funzioni pure**: le modifiche allo stato avvengono esclusivamente attraverso l'uso di funzioni pure.

3.2.1 STORE

Lo store è la parte dell'architettura che mantiene lo stato dell'applicazione e, come detto in precedenza, è manipolabile solo attraverso le actions.

```
import { applyMiddleware, legacy_createStore as createStore } from "redux";
import thunk from "redux-thunk";
import rootReducer from "./reducers";

const store = createStore(rootReducer, applyMiddleware(thunk));

export default store;
```

Figura 47 - creazione dello store in Health App

Lo store viene definito richiamando la funzione createStore (figura 47), che riceve come primo parametro il riferimento al root reducer e come secondo il riferimento al middleware.

3.2.2 MIDDLEWARE

Il middleware in React Redux consente di gestire l'elaborazione delle actions prima che queste raggiungano lo store. Può infatti intercettare, modificare o scartare le azioni inviate al reducer. Questo strumento offre supporto per una vasta gamma di funzionalità, tra cui la gestione delle chiamate asincrone, il logging e la registrazione delle azioni.

Esistono vari middleware che si integrano con Redux. Per quanto riguarda Health App, è stato adottato il middleware React Thunk, che risulta particolarmente utile per gestire operazioni asincrone. È essenziale quando ci si trova ad affrontare operazioni time-consuming, come le chiamate di rete.

La ragione che ci ha spinto ad utilizzare il middleware Redux Thunk è che, di norma, le azioni in Redux sono oggetti JavaScript che rappresentano modifiche di stato. Tuttavia, Redux Thunk ci consente di definire azioni come funzioni anziché oggetti. All'interno di queste funzioni thunk, è possibile gestire operazioni asincrone, come richieste API. Una volta che l'operazione asincrona è completata, la funzione thunk può inviare normali azioni utilizzando il metodo 'dispatch', influenzando lo stato dell'applicazione. Questo approccio semplifica notevolmente la gestione delle operazioni asincrone in Redux, contribuendo a mantenere il codice più chiaro e manutenibile.

```
export const getPatientQuestionnaires = (id) => (dispatch) => {
  return patientService.getPatientQuestionnaires(id).then(
    (data) => {
      if (data.data instanceof Array) {
        dispatch({
          type: "PATIENT_QUESTIONNAIRES",
          payload: data.data,
        });
      } else {
        dispatch({
          type: "LOGOUT",
        });
      }
    },
    (err) => {},
  );
};
```

Figura 48 - esempio di utilizzo di Redux Thunk in operazioni asincrone

Come evidenziato nell'esempio illustrato nella figura 48, l'azione è stata definita come una funzione thunk. Entrando più nel dettaglio, prima che l'azione 'PATIENT QUESTIONNAIRES' venga eseguita e processata dal relativo reducer, si attende il completamento della promise associata all'API che recupera i questionari di un paziente specifico.

3.2.3 REDUCERS

I reducers in Redux sono funzioni pure, responsabili di definire come lo stato dell'applicazione cambi in risposta alle azioni inviate. Queste funzioni ricevono lo stato attuale e un'azione, restituendo un nuovo stato. Per fare ciò, esaminano l'azione e il suo tipo per determinare come lo stato dovrebbe evolversi.

Nell'ambito di un'applicazione React-Redux, è possibile avere più reducers che operano su specifiche porzioni dello stato globale associato allo store. Questo è utile quando si ha un'applicazione molto grande e si desidera dividere la gestione dello stato in moduli più piccoli e gestibili.

```
1.  import { combineReducers } from 'redux'
2.  import contactsReducer from './contactsReducer';
3.  import uiReducer from './uiReducer';
4.
5.  const rootReducer =combineReducers({
6.
7.      contacts: contactsReducer,
8.      ui: uiReducer,
9.
10. })
11.
12. export default rootReducer;
```

Figura 49 - creazione di un root reducer mediante la funzione 'combineReducers'

Tuttavia, come anticipato in precedenza, durante la descrizione della creazione dello store, la funzione ‘createStore’ riceve un unico reducer, chiamato appunto ‘root reducer’. Per combinare più reducers in un reducer unico, si utilizza la funzione ‘combineReducers’. Questa funzione accetta un oggetto JavaScript in cui vengono mappate le singole funzioni "reducers" (come illustrato nella figura 49).

```
export default function reducer(state = initialState, action) {
  const { type, payload } = action;
  switch (type) {
    case "PATIENT_HRS":
      return {
        ...state,
        patientHRS: payload,
      };
    case "SLEEP_THRESHOLD":
      return {
        ...state,
        sleepThreshold: payload,
      };
    case "HR_THRESHOLD":
      return {
        ...state,
        hrThreshold: payload,
      };
    case "PATIENT_ID":
      return {
        ...state,
        patientId: payload,
      };
```

Figura 50 - funzione reducer (parziale) di Health App

Osservando la figura 50, è possibile vedere un estratto della funzione reducer definita nell'applicazione Health App.

Questa funzione in Redux è strutturata in modo da gestire il cambiamento di stato in base all'azione fornita. Solitamente, utilizza uno statement switch o una serie di if/else per identificare il tipo di azione e di conseguenza aggiornare lo stato.

3.2.4 ACTIONS

Le actions in Redux sono degli oggetti che vengono inviati allo store tramite il metodo 'dispatch'. Esse contengono una proprietà 'type' e, opzionalmente, dati aggiuntivi. Il campo 'type' specifica il tipo di azione da eseguire, mentre le altre proprietà opzionali trasportano dati supplementari correlati all'azione. In Health App, ad esempio, è presente un ulteriore campo 'payload' che rappresenta il valore associato allo stato a cui l'azione si riferisce.

```
export const getDoctors = () => (dispatch) => {
  authServices.getDoctors().then((data) => {
    try {
      var json = JSON.parse(JSON.stringify(data.data));
      if (!json || typeof json !== "object") {
        dispatch({
          type: "LOGOUT",
          payload: null,
        });
        return;
      }
    } catch (e) {
      dispatch({
        type: "LOGOUT",
        payload: null,
      });
    }
    dispatch({
      type: "DOCTORS",
      payload: data.data,
    });
  });
};
```

Figura 51 - esempio di action in Health App

Nello scenario in figura 51, il campo ‘payload’ memorizza il risultato della chiamata REST, che sarà poi elaborato dal ‘reducer’ appropriato per aggiornare la parte dello stato relativa all'elenco dei dottori. Successivamente, il componente dell'interfaccia utente contenente la lista dei dottori verrà nuovamente ricaricato e aggiornato per riflettere le modifiche.

Per concludere, Redux offre la possibilità di gestire uno stato globale, condiviso da tutta l'applicazione. Ma un componente React come può accedere a questo stato e sfruttarne il contenuto? Tramite l’utilizzo dell’hook 'useSelector' fornito da 'react-redux'. L'hook ‘useSelector’ richiede come argomento una funzione che riceve lo stato globale come parametro, consentendo così di estrarre e utilizzare selettivamente i dati di interesse. Questo approccio rappresenta un modo altamente efficace per integrare lo stato di Redux all'interno dei componenti React, contribuendo a mantenere il codice ben organizzato e garantendo un accesso agevole ai dati dell’applicazione.

3.3. MATERIAL DESIGN

Material Design [15] è un framework di design sviluppato da Google che si distingue per l'uso audace del colore, delle ombre e delle animazioni per creare un'esperienza utente coinvolgente e coerente.

Il focus di Material Design si concentra su tre punti fondamentali: creare, unificare e personalizzare. Con il suo approccio incentrato sulla riutilizzabilità dei componenti e sulla coerenza, questo framework crea un sistema di base che unifica l'esperienza dell'utente su diverse piattaforme, dispositivi e metodi di input, offrendo anche una solida base che consente di apportare personalizzazioni e innovazioni, permettendo alle aziende di esprimere la propria identità visiva.

3.3.1 MATERIAL UI

Material-UI [16] è un framework che offre una vasta gamma di componenti React, progettati per essere facilmente integrati e personalizzati, consentendo la rapida creazione di interfacce che seguono i principi del Material Design. Questi componenti sono indipendenti e funzionano senza richiedere configurazioni aggiuntive. Inoltre, Material-UI non si basa su fogli di stile globali, il che consente di applicare stili specifici solo ai componenti desiderati.

L'utilizzo di una libreria di componenti grafici predefiniti e ben testati semplifica notevolmente lo sviluppo e la distribuzione di un'applicazione, riducendo il tempo necessario per scrivere da zero l'HTML, il CSS e il JavaScript.

Ad ogni modo, se necessario, è possibile personalizzare i componenti predefiniti offerti da Material-UI, adattando due possibili strategie:

- Utilizzo delle props "style": È possibile modificare lo stile dei componenti predefiniti di Material-UI passando delle props al componente che si desidera personalizzare. Ciò consente di sovrascrivere gli stili predefiniti dei componenti, adattandoli alle esigenze specifiche dell'applicazione.

- Approccio con hook "styled": Material-UI fornisce un'API che sfrutta il concetto di hook per definire e applicare gli stili. Utilizzando questa API, è possibile gestire in modo dinamico e reattivo gli stili dei componenti, offrendo maggiore flessibilità nella personalizzazione.

3.4. AXIOS

Axios 14] è una libreria JavaScript ampiamente utilizzata per facilitare la comunicazione tra un client web e un server tramite il protocollo HTTP. Nell'ambito di Health App, Axios è stato impiegato all'interno delle azioni Redux per gestire le chiamate alle API esposte dal backend del sistema.

Per utilizzare Axios, è sufficiente richiamare l'istanza della libreria axios e indicare il tipo di richiesta HTTP che si desidera effettuare. Axios offre supporto per i quattro principali metodi HTTP: GET, POST, PUT e DELETE. Questa varietà di metodi consente agli sviluppatori di eseguire tutte le operazioni comuni coinvolte nell'interazione con un'API REST, come il recupero, l'inserimento, la modifica e la cancellazione dei dati.

Axios sfrutta il concetto di Promises [19]: ciascuna promise è un oggetto che contiene il risultato di un'operazione asincrona, la quale può avere esito positivo o negativo. Nel caso di un'esecuzione riuscita, il flusso del programma prosegue nel blocco then, in cui sono definite le azioni da eseguire con i dati ricevuti. Invece, se l'operazione non ha successo, il flusso del programma passa al blocco catch, in cui è possibile gestire l'errore e definire le azioni da intraprendere in caso di fallimento dell'operazione.

```
export const fitbitLogin = (patientId) => {
  return axios
    .get("/api/patients/" + patientId + "/fitbit-login", null)
    .then((response) => {
      window.location.replace(response.data);
    })
    .catch((error) => {
      console.error(error);
    });
};
```

Figura 52 - esempio di utilizzo della libreria Axios per effettuare il login su Fitbit

4. IMPLEMENTAZIONE DELLA WEB APPLICATION

Durante la fase implementativa, il nostro team ha collaborato seguendo i principi della metodologia agile, come accennato in precedenza. Abbiamo strutturato il lavoro in milestone, adottando un approccio iterativo attraverso sprint della durata di circa una settimana ciascuno. Al termine di ogni sprint, ci siamo dedicati attentamente al testing e al confronto diretto con i medici. Questo approccio ha favorito la ricezione di feedback immediati sullo sviluppo, contribuendo a garantire un processo di lavoro efficace e orientato alla soddisfazione delle esigenze del team medico.

In questo capitolo ci concentreremo sull'analisi della parte frontend della web application, realizzata utilizzando React e l'integrazione di altre librerie di terze parti. Questo approccio è stato adottato con l'obiettivo di ottimizzare la navigazione e facilitare la visualizzazione di ulteriori componenti grafici, contribuendo così a garantire un'esperienza utente chiara e intuitiva.

4.1. CONFIGURAZIONE DEL PROGETTO

4.1.1 CARATTERISTICHE GENERALI

Inizialmente, in fase di setup, è stato installato Node.js, un ambiente di runtime JavaScript. Grazie ad esso, tramite i comandi npm è possibile automatizzare l'installazione e l'aggiornamento di tutte le librerie di terze parti utilizzate nel progetto. Per quanto riguarda l'ambiente di sviluppo, si è scelto di usare Visual Studio Code come IDE principale.

Per il versionamento del codice è stato utilizzato Git, mentre il repository del progetto è ospitato su GitLab, che fornisce una piattaforma centralizzata per la collaborazione e il tracciamento delle modifiche. Questa infrastruttura ben integrata contribuisce alla gestione efficace del ciclo di vita del software, agevolando la collaborazione tra sviluppatori e garantendo un controllo preciso sulle versioni del codice.

4.1.2 CONFIGURAZIONE DEL PACKAGE.JSON

Durante la fase iniziale di configurazione del progetto, è stato definito il file package.json, che costituisce il manifesto dell'applicazione. Questo file non solo contiene i metadati relativi al progetto, ma elenca anche in modo dettagliato tutte le dipendenze necessarie, complete delle rispettive versioni, insieme agli script dedicati alla compilazione del codice sorgente.

Durante il processo di distribuzione di Health App lo script 'build', opportunamente inserito nel package.json, viene attivato. Questo script ha il compito di compilare il codice sorgente dell'applicazione, orchestrando la generazione di un pacchetto finale ottimizzato per la distribuzione. Tale pacchetto incorpora in maniera esaustiva tutti i file cruciali, tra cui HTML, CSS, JavaScript, e altre risorse essenziali. In questo modo, garantiamo che l'applicazione sia pronta per essere eseguita con successo in qualsiasi ambiente di distribuzione.

4.1.3 GESTIONE DELLE LIBRERIE ESTERNE

Le librerie elencate nella sezione ‘dependencies’ del file package.json costituiscono le dipendenze fondamentali per il corretto funzionamento dell'applicazione.

L'esecuzione del comando 'npm install’ semplifica notevolmente la gestione di tali dipendenze, consentendo al Node Package Manager (NPM) di recuperare e installare automaticamente tutti gli elementi necessari. Le risorse così ottenute vengono organizzate in modo ordinato nella cartella 'node_modules'. Inoltre, il file package-lock.json assume un ruolo essenziale nel garantire la stabilità del progetto, dettagliando con precisione le versioni specifiche delle dipendenze. Questa registrazione accurata contribuisce ad assicurare la coerenza e la riproducibilità dell'ambiente di sviluppo, evitando sorprese legate a possibili variazioni delle versioni. Questo approccio semplifica notevolmente la vita degli sviluppatori, consentendo loro di concentrarsi pienamente sulla scrittura del codice, senza preoccuparsi di intricati dettagli legati alla gestione delle risorse esterne.

Di seguito, una panoramica dettagliata delle principali librerie utilizzate in Health App:

Librerie React e Componenti UI:

1. React (v. 18.2.0): framework open-source JavaScript utilizzato per implementare component grafici.

2. React-DOM (v. 18.2.0): pacchetto che facilita le interazioni con il Document Object Model (DOM).

3. React-Router-DOM (v. 6.18.0): libreria per la gestione del routing e quindi della navigazione tra le pagine.

4. React-Scripts (v. 5.0.1): pacchetto preconfigurato che semplifica l'avvio e la configurazione rapida dei progetti React.

5. Redux (v. 4.2.1): libreria che offre uno store centralizzato per mantenere lo stato dell'applicazione, semplificando la gestione di dati complessi e consentendo un flusso prevedibile delle informazioni nell'intera app.

6. React-Redux (v. 8.1.3): libreria che permette di integrare React e Redux, offrendo componenti specializzati e collegamenti che facilitano la gestione dello stato globale in modo efficiente e dichiarativo.

Librerie Material-UI:

7. Mui/Material (v. 5.14.17): libreria di componenti predefiniti sviluppati secondo le linee guida del Material Design. Comprende anche Mui/Icons-Material, che offre un'ampia gamma di icone predefinite sempre nel formato Material Design.

8. Mui/X-Data-Grid (v. 6.18.1): libreria basata sul Material Design che permette di mostrare dati in forma tabellare, offrendo funzionalità sofisticate e flessibilità nella visualizzazione e manipolazione dei dati.

9. Mui/X-Date-Pickers (v. 6.18.1): pacchetto che fornisce dei componenti ready-made utili per gestire le date.

Librerie per la Visualizzazione di Grafici e Notifiche:

11. Nivo/Bar (v. 0.83.0): libreria che fornisce un componente flessibile che consente di creare grafici a barre adatti alle specifiche esigenze del progetto. Nivo/Bar non solo fornisce una vasta gamma di opzioni di personalizzazione, come l'orientamento delle barre, la gestione degli eventi interattivi e l'aspetto visivo, ma consente anche di configurare in dettaglio le proprietà dei dati, come valori, etichette e colori. Questa libreria è particolarmente utile per rappresentare in modo chiaro e accattivante dati basati su barre in contesti di visualizzazione e analisi.

12. Notistack (v. 3.0.1): libreria utilizzata per implementare il feedback da dare all’utente in seguito alle sue interazioni con la UI.

Librerie di Gestione Data e Orari:

13. Dayjs (v. 1.11.10): libreria Javascript per gestire date e orari.

Librerie di Configurazione e Formattazione:

14. Lint-Staged (v. 15.1.0): strumento che consente di eseguire linting solo sui file che sono stati modificati. Questo è particolarmente utile durante il processo di commit, ottimizzando il flusso di lavoro.

15. Prettier (v. 3.1.0): strumento di formattazione del codice che aiuta a mantenere uno stile di codifica coerente all'interno del progetto. Si integra con diversi linguaggi e offre una configurazione flessibile per adattarsi alle preferenze del team di sviluppo.

Librerie per le Richieste HTTP e il Proxying:

16. Axios (v. 1.6.1): libreria JavaScript per la gestione delle richieste HTTP, utilizzata sia lato client che lato server. Ampiamente adottata, Axios semplifica il processo di interazione con servizi esterni e permette di chiamare delle API in modo efficiente.

17. Http-Proxy-Middleware (v. 2.0.6): middleware che semplifica la configurazione di un server proxy durante lo sviluppo, consentendo di inoltrare richieste HTTP e HTTPS a un server diverso rispetto a quello su cui è in esecuzione l'applicazione. È utile per risolvere problemi di Cross-Origin Resource Sharing (CORS).

Queste librerie svolgono ruoli cruciali, dalla gestione dello stato e dell'interfaccia utente alla visualizzazione dei dati e alle chiamate API. Ogni versione è attentamente selezionata per garantire compatibilità e stabilità nell'ambiente di sviluppo di Health App.

4.1.3 ROUTING

Il routing assume un ruolo fondamentale nelle Single Page Applications, dove la struttura di base della pagina HTML rimane invariata. Di conseguenza, introdurre un sistema che emuli la navigazione tra pagine, simulando un cambio di URL ad ogni transizione, diventa imperativo. Questo approccio è essenziale per evitare che gli utenti si sentano smarriti durante la navigazione.

Per implementare questa funzionalità, è stata adottata la libreria React-Router. Questo strumento sfrutta l'API della cronologia offerta dai browser per sincronizzare l'URL visualizzato con lo stato dell'applicazione React. Ciò permette di aggiornare l'URL senza provocare un ricaricamento della pagina. In questo modo, si garantisce un'esperienza utente coerente e intuitiva, nonostante la struttura a pagina singola dell'applicazione.

```
function Routes() {
  const routes = useRoutes([
    { path: "/", element: <Home2 /> },
    { path: "/form-login", element: <FormLogin /> },
    { path: "/success-fitbit", element: <SuccessFitbit /> },
    { path: "/error-fitbit", element: <ErrorFitbit /> },
    { path: "/reset-password", element: <ResetPasswordPage /> },
    { path: "/change-password-form/:token", element: <ChangePasswordForm /> },
    { path: "/change-password-error", element: <ChangePasswordError /> },
    {
      path: "/patient",
      children: [
        { index: true, element: <HomePatient /> },
        { path: "fitbit-login", element: <FitbitLogin /> },
      ],
    },
    {
      path: "/admin",
      children: [
        {
          path: "doctors",
          children: [
            { index: true, element: <HomeAdmin /> },
            { path: "new", element: <CreateDoctor /> },
          ],
        },
```

Figura 53 - estratto della funzione Routes() che associa ad ogni path il componente corrispondente

4.1.5 COMPONENTI

Un componente rappresenta un elemento modulare e isolato all'interno dell'interfaccia di un'applicazione. La sua natura promuove la modularità e il riutilizzo del codice, consentendo la combinazione di diversi componenti per

formare strutture più complesse. L'essenza dei componenti non solo agevola il processo di sviluppo, evitando interferenze tra di essi durante le modifiche, ma promuove anche la facilità di estensione delle funzionalità dell'applicazione. Questo approccio facilita la creazione di interfacce scalabili e manutenibili, migliorando la gestione del processo di sviluppo del software.

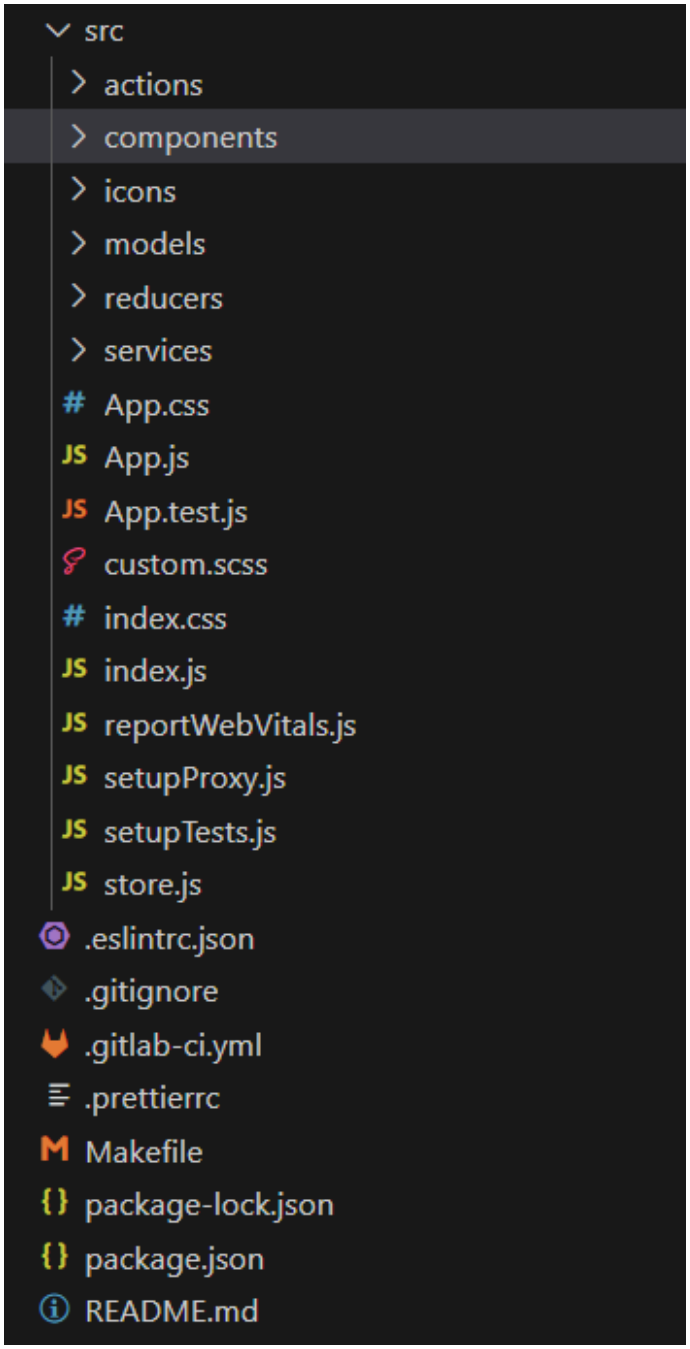

Figura 54 - Albero dei componenti in Health App

Nel progetto Health App, i diversi componenti sono distribuiti in cartelle e sottocartelle, seguendo le linee guida del pattern Redux.

Come mostrato in figura 54, il file ‘store.js’ rappresenta il file in cui si definisce lo stato dell’applicazione.

In aggiunta, sono stati definiti due packages, denominati 'actions' e 'reducers'. Mentre il primo contiene le azioni che rappresentano le varie interazioni dell'utente o eventi nell'app, il secondo si occupa della gestione delle modifiche allo stato dell'applicazione in risposta a tali azioni. Questa organizzazione offre una struttura chiara e modulare, facilitando la comprensione e la manutenzione del codice.

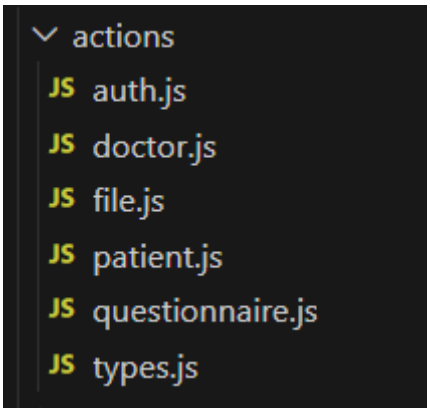

Figura 55 - azioni del Progetto

All'interno della cartella ‘actions’ si trovano diversi file, tra cui auth.js, doctor.js, file.js, patient.js e questionnaire.js. Ciascun file è dedicato a una specifica area di interesse e contiene le azioni associate. Il file types, invece, contiene l’elenco dei nomi di tutte le azioni.

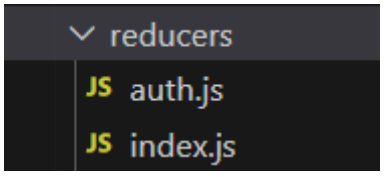

Figura 56 - reducers del Progetto

Nella cartella ‘reducers’ si trovano due file: index.js e auth.js. Mentre nel primo viene dichiarato il reducer mediante l'invocazione della funzione 'combineReducers', il secondo contiene tutti gli stati globali dell'applicazione e si occupa anche di gestire le modifiche a questi stati in risposta alle azioni. La

presenza di questo file consolida la gestione dello stato globale, fornendo un punto centrale per la definizione e l'aggiornamento degli stati in diversi contesti di Health App.

Figura 57 - file dedicati all'interazione con le API esposte lato backend

Il package ‘services’ contiene diversi file che, tramite l’utilizzo della libreria Axios, comunicano con le API esposte dal backend scambiando informazioni.

È rilevante notare che, analogamente alle actions, ogni area di interesse è associata a un file dedicato. Inoltre, vi è una stretta correlazione tra le actions e i servizi, poiché ciascun file di actions richiama un metodo specifico definito nel corrispondente servizio per eseguire l'azione desiderata.

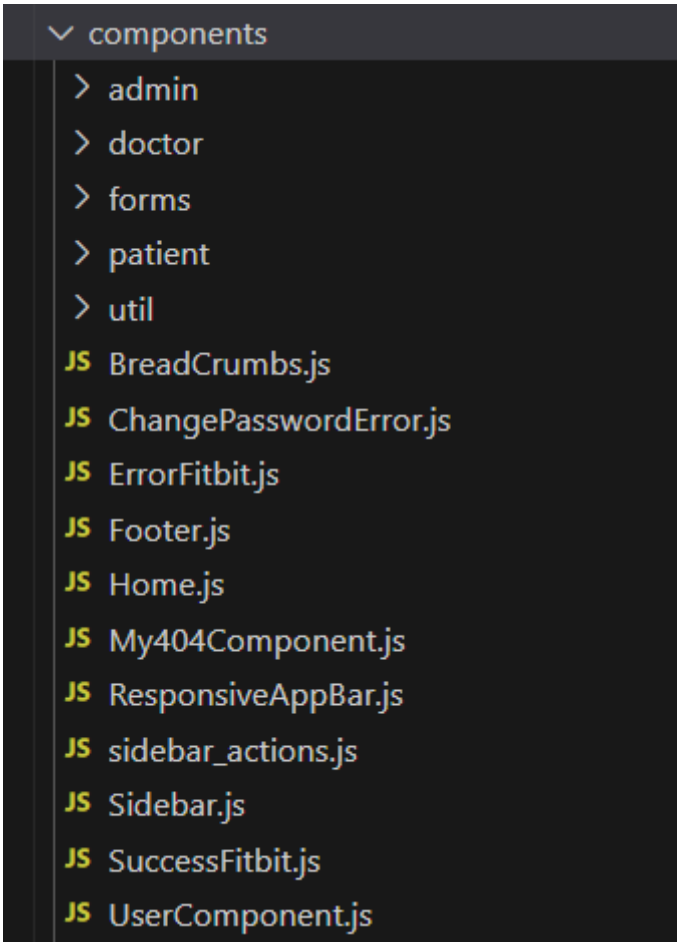

Figura 58 - sezione dedicata ai component

I componenti sono strutturati in cartelle con l'obiettivo di identificare chiaramente le funzionalità e gli utenti destinatari. Questa organizzazione comprende divisioni concettuali, come evidenziato nel package 'forms', che raccoglie tutti i form di inserimento e aggiornamento. Inoltre, ulteriori raggruppamenti sono effettuati in base all'utilizzatore finale, distinguendo chiaramente le sezioni della web application dedicate ad amministratori, medici e pazienti.

I componenti generici, condivisi da tutte le tipologie di utenti, sono posizionati in modo generico nella directory 'components'. Questa organizzazione facilita la navigazione e la comprensione della gerarchia dei componenti, assicurando una disposizione intuitiva e ordinata.

Tra i componenti condivisi abbiamo:

ResponsiveAppBar.js: costituisce la barra di navigazione per tutte le pagine dell' applicazione Health App. E' caratterizzata dalla presenza di un logo, dal nome dell'app e da un pulsante posizionato strategicamente sulla parte destra. Tale

pulsante, contrassegnato dalle iniziali dell'utente, non solo facilita il logout, ma si presta anche a ulteriori espansioni. In prospettiva, infatti, potremmo arricchire questo spazio con funzionalità aggiuntive, come la gestione del profilo e altre opzioni utili per l'utente.

Figura 59 - barra di navigazione dell'applicazione Health App

Sidebar.js: oltre alla barra di navigazione, l'interfaccia è completata da una sidebar che ospita le diverse sezioni navigabili. Inoltre, in basso, un footer offre ulteriori informazioni chiave. Quest'ultimo contiene infatti il nome, la versione attuale dell'applicazione e la dicitura "All Rights Reserved" a titolo di copyright. In futuro, prevediamo di arricchire ulteriormente questa sezione, aggiungendo un link all'informativa sulla privacy. Comunque, la presenza del copyright enfatizza il nostro impegno nella protezione dei diritti riservati e nella trasparenza delle informazioni fornite.

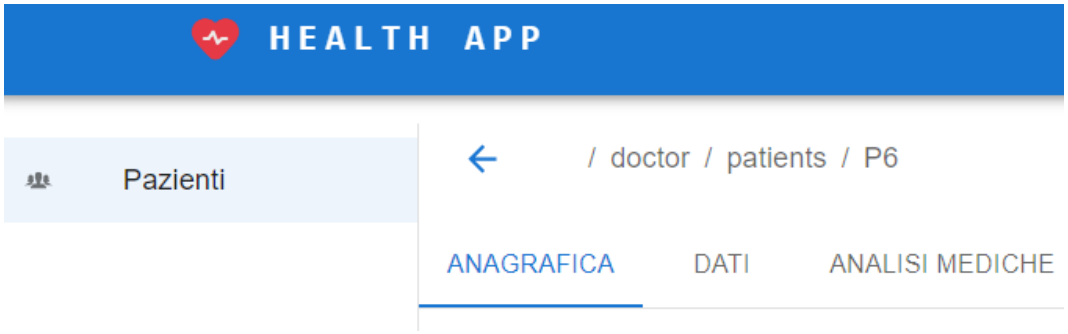

Figura 60 - sezioni navigabili mostrate nella sidebar

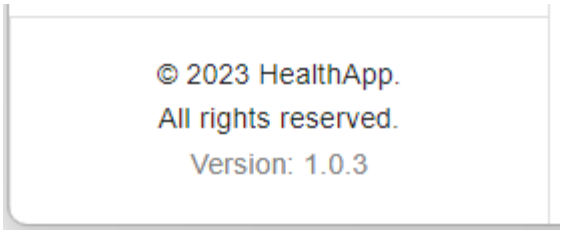

Figura 61 - Esempio di footer

BreadCrumbs.js: componente progettato per visualizzare in modo chiaro e dettagliato il percorso compiuto dall'utente all'interno dell'applicazione.

Esso è posizionato nella parte alta della pagina, subito sotto il componente ResponsiveAppBar.

Figura 62 - Esempio di breadcrumbs: anagrafica del paziente

BarChart.js: sviluppato a partire dal componente ResponsiveBar della libreria nivo/bar, questo componente è stato adattato alle nostre esigenze personalizzandolo tramite l'utilizzo di props specifiche. Ad esempio, è stata configurata la disposizione delle barre in maniera flessibile, consentendo di raggrupparle o sovrapporle a seconda delle necessità.

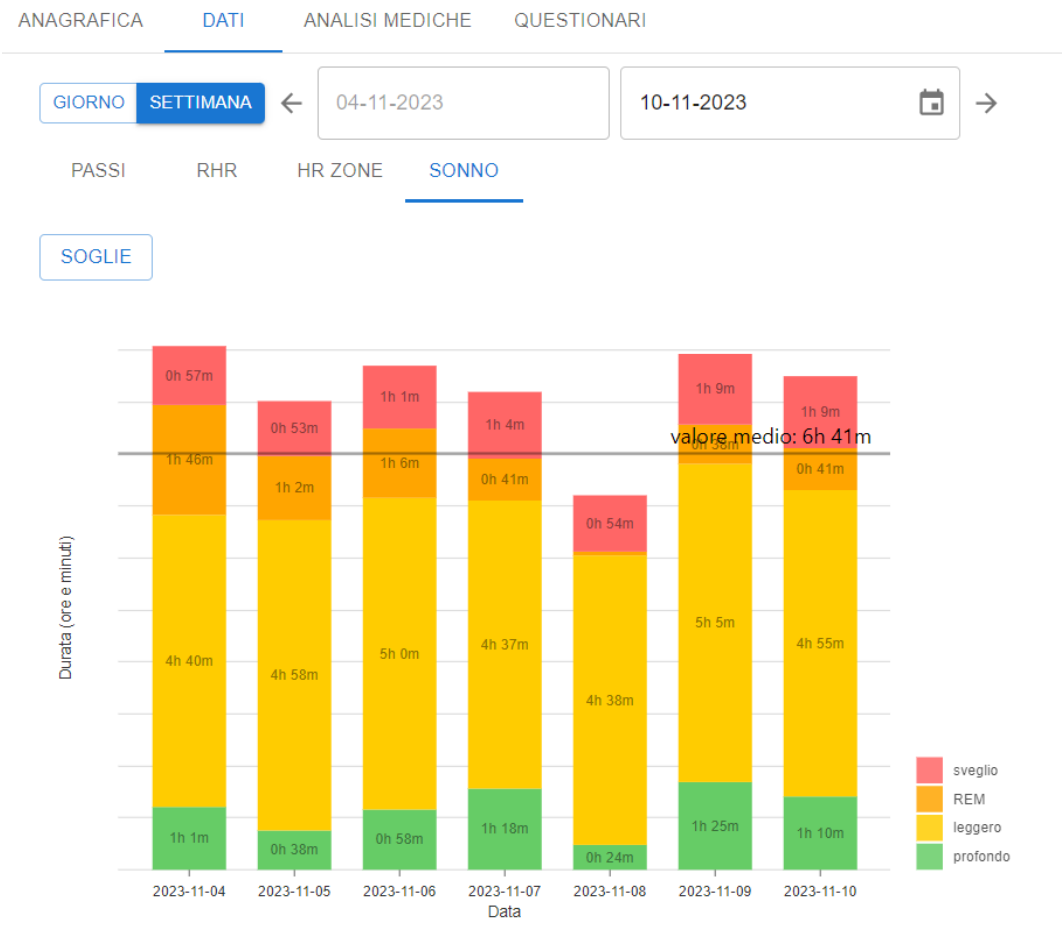

Figura 63 - esempio di barChart con groupMode="stacked"

My404Component.js: componente che viene visualizzato nel caso in cui venga inserito un percorso non definito tra i "path" dei componenti Route (come indicato nella parte inferiore del componente App nella Figura 64). Il componente My404Component.js sarà renderizzato se non viene trovata una corrispondenza tra il percorso inserito e i componenti specificati in precedenza nel sistema.

```
            },
          ],
        },
        { path: "*", element: <My404Component /> },
      ]);
      return routes;
    }
```

Figura 64 - path del componente 'My404Component'

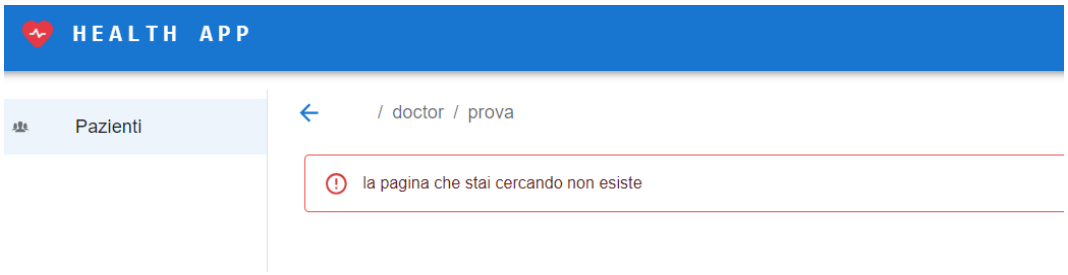

Figura 65 - componente mostrato in presenza di path non validi

```
const { createProxyMiddleware } = require("http-proxy-middleware");

module.exports = function (app) {
  app.use(
    "/api",
    createProxyMiddleware({
      target: "http://localhost:8080",
      changeOrigin: true,
    }),
  );
```

Figura 66 - esempio di creazione di un proxy middleware

setupProxy.js: questo file configura un middleware proxy utilizzando la libreria ‘http-proxy-middleware’ per gestire le richieste HTTP in un'applicazione React durante lo sviluppo. Il middleware proxy è utile quando l'applicazione frontend deve comunicare con un backend, ma il server backend ha un'origine diversa o un diverso dominio. In questo caso, il file di configurazione del proxy specifica che tutte le richieste iniziate con "/api" devono essere reindirizzate al server locale in esecuzione su "http://localhost:8080". L'opzione "changeOrigin: true" indica al proxy di cambiare l'origine del richiedente durante l'inoltro delle richieste, assicurando che il server di destinazione riconosca correttamente l'origine della richiesta proveniente dall'applicazione frontend.

MakeFile: un Makefile è un file di configurazione utilizzato in fase di sviluppo del software per definire le regole e le operazioni necessarie per la compilazione, la gestione delle dipendenze e l'esecuzione di un progetto. Nella cartella del progetto Health App, eseguendo il comando ‘make’, sarà il Makefile ad occuparsi di scaricare le dipendenze necessarie ed eseguire l'applicazione, rendendo il processo di sviluppo più efficiente, evitando di eseguire manualmente una serie di comandi. In breve, l'utilizzo del Makefile semplifica la gestione delle attività di sviluppo, migliorando la produttività complessiva del team di sviluppo.

4.1.6 AUTENTICAZIONE

Per accedere all’applicazione, è indispensabile autenticarsi con successo.

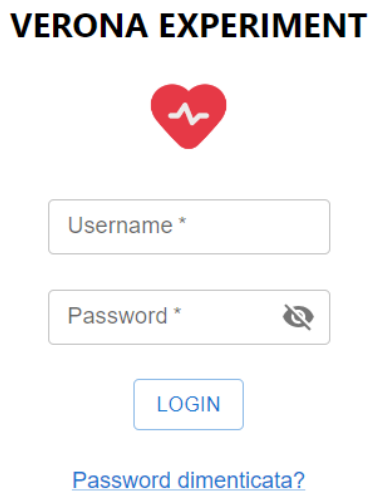

Figura 67 - pagina di login

La schermata di accesso, illustrata nella figura 67, consente di inserire le credenziali compilando i campi username e password. Utilizzando l'attributo type="password", la password viene cifrata e rimane invisibile mentre la si digita. Cliccando sull'icona a forma di occhio è invece possibile visualizzarla in chiaro, in maniera tale da notare e correggere eventuali errori.

Le credenziali inserite sono prima sottoposte ad un processo di validazione e, in presenza di eventuali errori, vengono mostrati dei feedback all'utente. La gestione dell'autorizzazione segue lo standard OAuth, consentendo chiamate sicure e autorizzate alle API mediante l'utilizzo di un token di accesso.

La procedura di autenticazione avviene tramite una chiamata al backend in cui username e password vengono inviati come parametri. Se validi, il client riceve un token di autenticazione JWT. Questo token viene memorizzato all'interno di un cookie e successivamente utilizzato nell'header delle richieste successive per accedere alle risorse. Il cookie ha una validità temporale, dopo la quale è necessario eseguire nuovamente l'accesso.

Nell'ambito della gestione dell'autenticazione, il componente 'UserComponent.js' riveste un ruolo significativo. Gestisce infatti situazioni come l'aggiornamento della pagina o periodi prolungati di inattività: se la sessione è scaduta, il componente reindirizza l'utente alla pagina di login; in caso contrario, ricarica la pagina visualizzata precedentemente.

4.1.7 FEEDBACK UTENTE

La gestione del feedback utente riveste un ruolo centrale nell'esperienza complessiva offerta da un'applicazione. La sua efficacia non si limita solamente alla gestione degli errori, ma si estende anche al riconoscimento delle operazioni eseguite con successo. La capacità di comunicare in modo chiaro e tempestivo con gli utenti è fondamentale per creare un ambiente interattivo e intuitivo.

In Health App è stato implementato un sistema robusto che fornisce informazioni chiare sia in caso di errori che di operazioni eseguite correttamente. Ciò si traduce in una comunicazione proattiva con l'utente, garantendo che ogni azione intrapresa sia seguita da una risposta visiva adeguata.

Un elemento chiave in questo processo è lo SnackbarProvider, che sebbene non sia un componente core sviluppato internamente, merita menzione in quanto riveste un ruolo cruciale nella gestione delle notifiche all'interno di un'app React ed è ampiamente utilizzato in Health App. La sua importanza sta nel fornire agli utenti un dettaglio completo delle operazioni eseguite, assicurando una

comunicazione trasparente in merito agli esiti delle interazioni con il sistema backend.

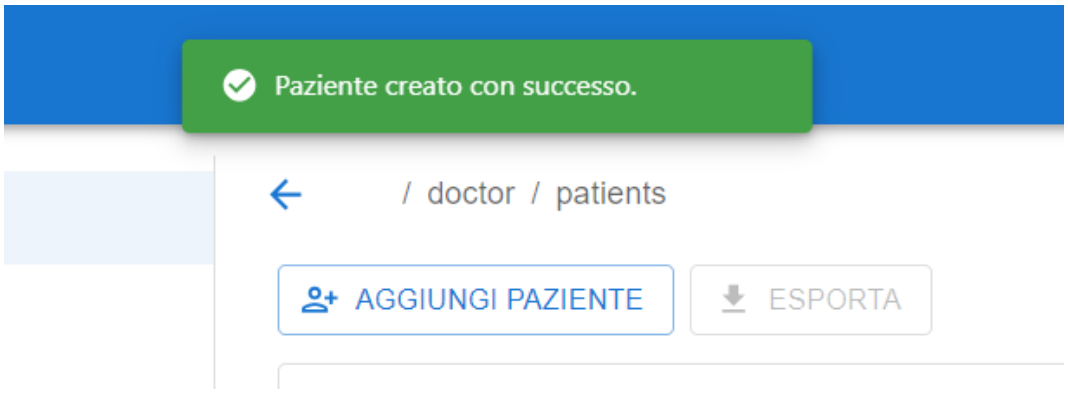

Figura 68 - esempio di feedback positivo dato in seguito alla creazione di un paziente

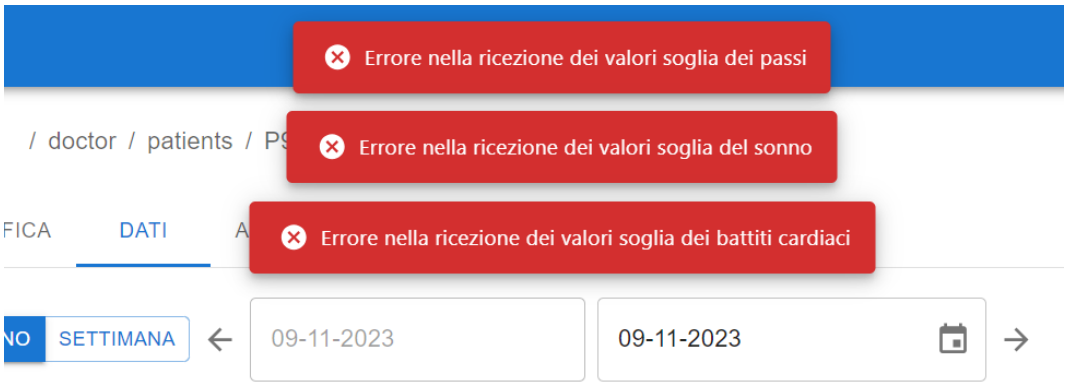

Figura 69 - messaggi mostrati in seguito agli errori ricevuti in risposta dalle chiamate http

5. USABILITY TEST

Nel processo di sviluppo delle applicazioni web, garantire un'esperienza utente ottimale è di cruciale importanza. L'usabilità [20], che riflette la facilità con cui gli utenti possono interagire con un sistema, gioca un ruolo fondamentale nel determinare il successo di un'applicazione. Questo capitolo si concentra sull'esecuzione di uno usability test mirato alla valutazione approfondita della nostra web app, con l'obiettivo di individuare e risolvere potenziali problemi che potrebbero influenzare negativamente l'esperienza degli utenti finali.

Valutare l'usabilità non è unicamente una questione di estetica, ma piuttosto un approccio strategico volto a massimizzare la soddisfazione dell'utente, la produttività e la fidelizzazione. Attraverso uno studio approfondito dell'usabilità, si è in grado di esplorare le dinamiche di interazione tra gli utenti e l'applicazione, al fine di creare un ambiente che risponda in modo efficiente e intuitivo alle esigenze degli utilizzatori.

Si è quindi deciso di condurre un test di usabilità sulla Web Application di Health App, coinvolgendo 10 candidati rappresentativi di diverse fasce di età. Questa scelta ci ha permesso di ottenere un quadro completo delle esperienze degli utenti, garantendo una valutazione il più generica possibile. I risultati di questo test sono stati poi sottoposti a un'analisi dettagliata per identificare punti di forza e debolezza, fornendo una solida base per il miglioramento del nostro prodotto.

Prima del test, ciascun partecipante ha completato un questionario, utile per comprendere la sua familiarità con la tecnologia e le web applications. Inoltre, è stata fornita un'ampia panoramica del progetto per contestualizzare l'applicazione. Durante questa fase, è stato chiarito lo scopo del test, assicurandosi che gli utenti fossero consapevoli che l'oggetto di valutazione fosse l'applicazione stessa, e non le prestazioni personali degli utenti.

Il test è stato strutturato in 10 task, ciascuno accompagnato da uno scenario per fornire all'utente le informazioni necessarie per completarlo con successo. Una volta completati tutti i task, è stato richiesto all'utente di rispondere al questionario "SUS".

Il System Usability Scale (SUS) è uno strumento ampiamente utilizzato per valutare la percezione degli utenti sull'usabilità di un sistema. Si tratta di un questionario standardizzato composto da 10 affermazioni, ciascuna valutata su una scala Likert a 5 punti che va da "Fortemente in disaccordo" a "Fortemente d'accordo".

Le affermazioni sono poste in modo tale che alcune siano orientate positivamente verso l'usabilità del sistema, mentre altre lo siano negativamente. Questa varietà mira a ottenere una visione completa della percezione dell'utente. Le risposte vengono poi sommate e trasformate secondo una formula specifica per ottenere un punteggio totale che rappresenta la valutazione complessiva della usabilità del sistema.

Il punteggio della System Usability Scale (SUS) può variare da 0 a 100. Il valore medio è 68, mentre punteggi superiori indicano una maggiore percezione di usabilità da parte degli utenti. Il questionario SUS [21] è apprezzato per la sua semplicità e flessibilità, rendendolo uno strumento affidabile per valutare la qualità dell'esperienza utente in una vasta gamma di contesti.

#	**Lista dei Task**
T1	Prime impressioni, dai un'occhiata.
T2	Crea un nuovo paziente inserendo i tuoi dati.
T3	Completa la registrazione del paziente collegando il suo account Fitbit.
T4	Vai nella sezione 'Analisi Mediche' e inserisci il tuo peso odierno.
T5	Compila il questionario 'Medas'.
T6	Consulta i risultati del questionario appena compilato.
T7	Seleziona un altro paziente già presente nella piattaforma e visualizza il numero di passi che ha compiuto nell'ultima settimana.
T8	Fai lo stesso, però per la sezione 'Sonno'.
T9	Riapri il paziente da te precedentemente creato e inserisci i valori delle analisi del sangue.
T10	Cancella l'utente che hai creato al punto 1.

Figura 70 - Lista dei task eseguiti durante il test

RISULTATI

Tutti i partecipanti hanno portato a termine con successo l'intero set di task proposti.

Nonostante ciò, il terzo task ha creato non poche difficoltà ad alcuni. La sequenza di operazioni che include il logout, l'impersonificazione del paziente appena creato, la connessione dell'account Fitbit a Health App e, infine, il successivo login come dottore, è stata infatti percepita da alcuni come un processo potenzialmente complicato e macchinoso.

Tra gli aspetti positivi riscontrati, è importante evidenziare che le informazioni relative ai dati biometrici raccolti dal tracker sono presentate in modo accattivante. Gli utenti possono facilmente visualizzare l'andamento dei dati nel tempo, distinguendo le varie fasi in modo chiaro grazie a un uso intelligente dei colori. Un apprezzamento è stato espresso anche per la semplicità nella compilazione dei questionari, con un plauso per l'efficace automatizzazione di un processo che tradizionalmente richiede la gestione su carta.

Per quanto riguarda i miglioramenti, si potrebbe considerare di arricchire la grafica per renderla più accattivante e coinvolgente. Inoltre, sarebbe utile consentire ai medici di importare direttamente file PDF per le analisi del sangue, semplificando il processo e migliorando l'esperienza complessiva degli utenti.

Il fatto che i partecipanti non abbiano riscontrato problematiche significative nello svolgere i vari task può essere ricondotto all'impiego di un approccio basato sui principi del Material Design. Questa scelta ha contribuito a garantire una familiarità degli elementi dell'applicazione con quelli già presenti in altre app, creando un'esperienza utente fluida e intuitiva.

I risultati ottenuti attraverso il “SUS” hanno registrato una media del 91.5, notevolmente al di sopra del valore medio di riferimento (68). Da sottolineare che solo nel 10% dei casi di test di solito si raggiunge o supera un punteggio pari al 80.3, indicando quindi un risultato particolarmente positivo. Questi dati non solo forniscono spunti costruttivi per possibili miglioramenti, ma confermano che la direzione intrapresa è quella corretta.

La valutazione ci offre una prospettiva incoraggiante, suggerendo che si sta consolidando un approccio all'usabilità che va oltre le aspettative comuni.

6. CONCLUSIONI

In questo lavoro di tesi è stata sviluppata la user interface dell'applicazione Health App, con l'obiettivo di fornire uno strumento che coniugasse intuitività ed efficacia per soddisfare appieno gli obiettivi dell'esperimento. Dopo una fase dedicata all'implementazione, ci siamo focalizzati sui test di usabilità, i quali hanno restituito risultati incoraggianti. Attualmente, l'applicazione è operativa e utilizzata in maniera proficua, ma il nostro team rimane sempre vigile e pronto a implementare miglioramenti qualora necessario.

Questa esperienza è stata per me molto formativa e stimolante poiché mi ha dato l'opportunità di contribuire ad un progetto in cui la tecnologia è applicata in un contesto medico, infatti il nostro impegno ha dato vita a un prodotto che migliora effettivamente la qualità di vita delle persone.

Inoltre, lavorare in team è stato un elemento chiave di questa esperienza e costituisce un aspetto che ritengo altamente formativo e prezioso. La collaborazione tra membri del team ci ha permesso di affrontare sfide complesse, pianificare congiuntamente e trovare soluzioni innovative. La gestione di scadenze e la partecipazione attiva a meeting settimanali e sessioni di brainstorming hanno contribuito invece a consolidare le mie competenze organizzative e decisionali.

Ciò che emerge da questo percorso non è solo la soddisfazione per i risultati raggiunti, ma anche una crescita personale e professionale significativa. La guida preziosa del docente e la collaborazione costruttiva dei colleghi hanno contribuito a garantire non solo il successo del progetto, ma anche il mio sviluppo individuale. Attraverso le sfide affrontate e il lavoro di squadra, ho acquisito nuove competenze e una consapevolezza più profonda del mio percorso accademico. Guardando al futuro, sono grato per aver colto quest'opportunità così preziosa e stimolante, che sicuramente si rifletterà positivamente nel mio percorso professionale.

INDICE DELLE FIGURE

BIBLIOGRAFIA

- Maximilian Schwarzmuller, React Key Concepts: Consolidate your knowledge of React's core features, 26 dicembre 2022, Inglese
- Adam Boduch, React Material-UI Cookbook: Build captivating user experiences using React and Material-UI, I edizione, Inglese
- Greg L.Turnquist, Learning Spring Boot 3.0: Simplify the development of production-grade applications using Java and Spring, III edizione, 30 dicmebre 2022, Inglese

1. IBM Cloud Education, Three-Tier Architecture, https://www.ibm.com/cloud/learn/three-tier-architecture
2. RedHat, Agile Methodology https://www.redhat.com/it/devops/what-is-agile-methodology
3. IBM Cloud Education, Java Spring Boot https://www.ibm.com/it-it/topics/java-spring-boot
4. Spring, Spring Data https://spring.io/projects/spring-data
5. Spring, Spring Security https://spring.io/projects/spring-security
6. Spring, Building we applications with Spring Boot and Kotlin https://spring.io/guides/tutorials/spring-boot-kotlin/
7. Kotlin, Null Safety language https://kotlinlang.org/docs/null-safety.html
8. Kinsta.com, What is Express.js? https://kinsta.com/knowledgebase/what-is-express-js/
9. Node JS Architecture – Single Threaded Event Loop https://www.digitalocean.com/community/tutorials/node-js-architecture-single-threaded-event-loop

10. React
    https://en.wikipedia.org/wiki/React_(software)
11. Componenti e Props
    https://it.legacy.reactjs.org/docs/components-and-props.html
12. React Router
    https://reactrouter.com
13. Angular vs React
    https://kinsta.com/it/blog/angular-vs-react/
14. Axios
    https://axios-http.com/
15. Material Design
    https://m3.material.io
16. Material UI: React components based on Material Design
    https://mui.com/material-ui/
17. React Redux
    https://react-redux.js.org
18. React Redux, fundamentals
    https://redux.js.org/tutorials/fundamentals/part-5-ui-react
19. Promise – JavaScript
    https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
20. Measuring Perceived Usability with the SUS, NASA-TLX, and Single Ease Question After Tasks and Usability Tests
    https://www.nngroup.com/articles/measuring-perceived-usability/
21. The Pros and Cons of the System Usability Scale (SUS)
    https://research-collective.com/sus/
22. Introduzione a JSX
    https://it.legacy.reactjs.org/docs/introducing-jsx.html
23. ReactJS, Virtual DOM
    https://www.geeksforgeeks.org/reactjs-virtual-dom/
24. ReactJS, Undirectional Data Flow
    https://www.geeksforgeeks.org/reactjs-unidirectional-data-flow/

25. React Native, outstanding app performance with a single code base
https://www.milocreative.com/post/react-native-app-development
26. Introduzione agli Hooks
https://legacy.reactjs.org/docs/hooks-intro.html
27. Utilizzare al meglio l'Hook useEffect di React
https://kinsta.com/it/knowledgebase/react-useeffect/
28. Cosa avviene durante le fasi del sonno
https://www.helsana.ch/it/blog/corpo/conoscenza-del-corpo/fasi-del-sonno.html
29. React, tutto sulla libreria Javascript
https://www.codemotion.com/magazine/it/linguaggi-programmazione/react-tutto-sulla-library-javascript/
30. Riconciliazione
https://it.legacy.reactjs.org/docs/reconciliation.html