Politecnico di Torino

POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Anno Accademico 2022/2023

Sessione di Laurea Dicembre 2023

HealthApp: Sviluppo e Manutenzione del Backend per l'Applicazione di Monitoraggio degli Assistiti Medici

Relatore:
Prof. Maurizio MORISIO

Candidato:
Marco MELE

Ad Azzurra, la mia forza

Indice

Indice delle Figure

Elenco delle tabelle

Elenco delle formule

Elenco degli allegati

1. Presentazione

1.1 Obiettivi

Health App è un'idea che nasce dalla collaborazione del Dipartimento di Automatica e Informatica del Politecnico di Torino e alcuni medici dell'Azienda Ospedaliera Universitaria Integrata di Verona, cercando di applicare al mondo medico la tecnologia informatica, che può essere utile in diversi ambiti e per diversi scopi.

L'obiettivo dell'applicazione è assistere i medici nel monitoraggio dei parametri vitali dei pazienti, tra cui **sonno, attività fisica, salute e alimentazione**, e capire se un'applicazione del genere possa influenzare positivamente lo stile di vita dei pazienti coinvolti nell'esperimento che nel caso specifico saranno pensionati.

Attraverso una fase di studio sono stati individuati i casi d'uso principali e questo è stato fondamentale per adottare le giuste scelte implementative. Il progetto riguarda lo sviluppo e il mantenimento di un'applicazione completa, con una parte backend, una frontend web e un'applicazione mobile (che per il momento è stata messa in standby), dedicata al monitoraggio dei pazienti.

I medici avranno la possibilità di creare gli account dei pazienti, i quali potranno collegare il proprio account al loro account Fitbit personale. Durante l'esperimento, che durerà circa sei mesi, i pazienti utilizzeranno il dispositivo indossabile Fitbit per l'intera giornata.

Tutti i dati relativi all'attività fisica, al sonno e alla salute saranno inviati all'applicazione e archiviati nel nostro database. I medici potranno aggiungere analisi mediche e valori corporei dei pazienti, come il peso, e potranno accedere alle informazioni in modo semplice per monitorare l'evoluzione dello stato di salute e dello stile di vita dei pazienti.

Oltre a queste funzionalità, un recommender system è integrato nell'applicazione, questo, analizzando i dati dei pazienti in base a delle soglie prestabilite o inserite dai medici, produce dei risultati che aiutano i pazienti a monitorare e a curare il proprio stile di vita e i medici a seguirli attentamente. È importante sottolineare che le raccomandazioni non sostituiscono in nessun caso il parere di un medico.

Analizzando attentamente le diverse aree (sonno, attività fisica, salute e alimentazione), possiamo dire che:

- **Sonno:** È possibile monitorare diversi parametri relativi al sonno del paziente, tra cui le zone di sonno (profondo, leggero, REM, sveglio), l'andamento temporale e la media settimanale.

- **Attività fisica:** È possibile monitorare i dati relativi all'attività fisica, come i passi, al fine di valutare i risultati ottenuti, osservando l'andamento temporale e la media settimanale dei passi.

- **Salute:** È possibile tenere sotto controllo diversi parametri, tra cui il battito cardiaco medio giornaliero a riposo, le zone di frequenza cardiaca, l'indice di massa corporea (BMI) e i valori delle analisi mediche. (Il peso, da cui si ricava il BMI, e i valori delle analisi mediche, devono essere caricati manualmente dal medico).

- **Alimentazione**: Attraverso l'utilizzo dell'applicazione mobile il paziente ha la possibilità di registrare e quindi monitorare i cibi consumati durante il giorno ed essere informato per quanto riguarda i valori nutrizionali. (L'applicazione mobile è temporaneamente in stand-by). I medici possono così tenere sotto controllo i parametri riguardanti i valori nutrizionali assunti dai pazienti.

Come affermato dal prof. Luigi Fontana, una particolare attenzione per la dieta, l'attività fisica e altre piccole accortezze, possono migliorare lo stile di vita dei pazienti e aumentare l'aspettativa di vita (1).

1.2 Architettura dell'Applicazione

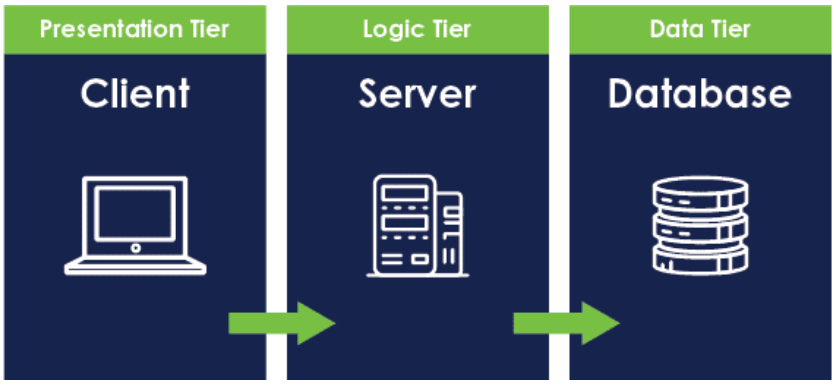

Figura 1 – Three tier architecture

L'architettura a tre livelli (2), nota anche come *three tier architecture* [1], è stata la scelta strutturale per la nostra applicazione. Questo approccio prende il nome dalla sua suddivisione appunto in tre livelli [2]:

- **Presentation tier:** Questo strato rappresenta l'interfaccia utente e il livello di comunicazione dell'applicazione, consentendo all'utente finale di interagire con il sistema. Il suo obiettivo principale è visualizzare le informazioni e raccogliere dati degli utenti. Ci possono essere diverse tipologie di client, browser (pc), mobile (smartphone).

- **Application tier:** anche conosciuto come tier logico o tier intermedio, costituisce il nucleo dell'applicazione. Qui le informazioni vengono raccolte nel tier presentazione ed elaborate, talvolta in relazione ad altre informazioni contenute nel tier dati, applicando delle business rules. Il livello applicazione può aggiungere, eliminare o modificare i dati nel livello dati.

- **Data tier:** Il tier dati, anche denominato tier del database o tier di accesso ai dati, costituisce il punto in cui vengono archiviate e gestite le informazioni elaborate dall'applicazione. Questo strato può utilizzare un sistema di gestione del database relazionale come PostgreSQL, oppure NoSQL come MongoDB.

In un'architettura a tre livelli, tutte le comunicazioni avvengono attraverso il tier applicazione. Il tier presentazione e il tier dati non possono stabilire una comunicazione diretta tra di loro.

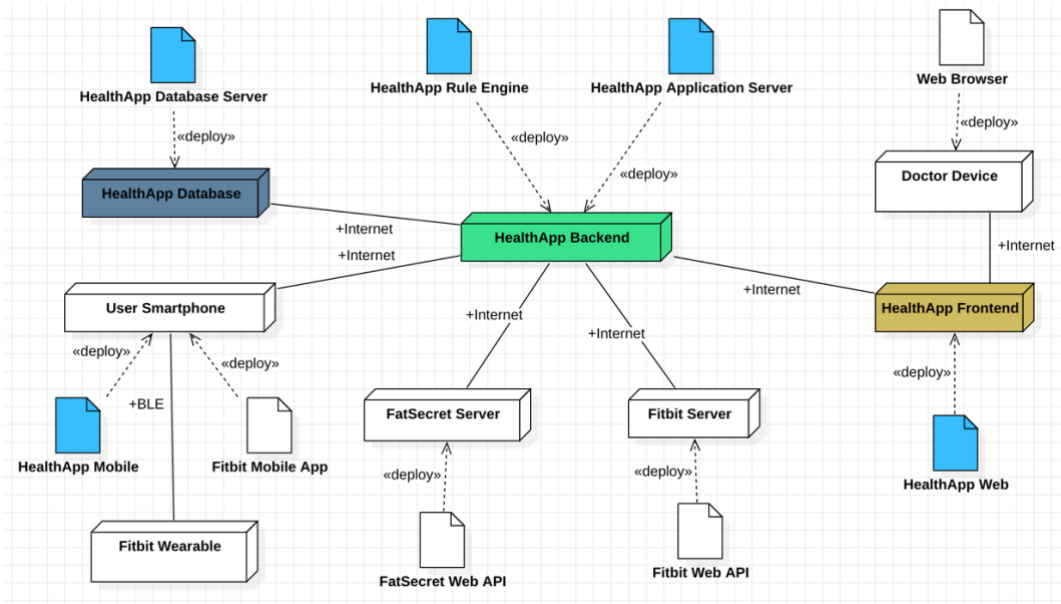

Figura 2 – Diagramma Architetturale HealthApp

HealthApp Frontend, HealthApp Backend ed HealthApp Database sono i tre nodi principali della nostra applicazione, ovvero le componenti hardware, all'interno dei quali troviamo gli artifacts, i componenti software. Alcuni componenti sono stati sviluppati all'interno del nostro progetto, altri (quelli in bianco) sono componenti esterni che interagiscono con la nostra applicazione e forniscono dei servizi, come ad esempio i servizi Fitbit e FatSecret. Nel diagramma sono presenti anche il server di FatSecret e l'applicazione mobile che sono al momento è in stand-by poiché il focus è stato spostato maggiormente sull'utilizzo dell'applicazione da parte dei medici.

Analizzando nel dettaglio il diagramma si possono notare lato hardware:

- **HealthApp Backend:** Rappresenta l'infrastruttura fisica che supporta l'applicazione, consentendo il suo funzionamento, si interfaccia con il frontend e permette la gestione dei dati e l'interazione con gli altri server di terze parti, che nel nostro caso sono Fitbit e FatSecret.

- **HealthApp Frontend:** Hardware sul quale si trova deployata la web application, si connette all'HealthApp Backend tramite internet.
- **HealthApp Database:** Macchina su cui è deployato l'HealthApp Database Server.
- **Doctor Device:** Dispositivo che il dottore utilizza e che tramite un web browser accede al frontend di HealthApp per sfruttarne le funzionalità.
- **Fitbit Server:** Accessibile mediante API, rende disponibili alla nostra applicazione i dati relativi a sonno, attività fisica e frequenza cardiaca dei pazienti.
- **Fitbit Wearable:** Dispositivo indossabile che serve a monitorare e registrare diverse metriche, questo verrà usato dai pazienti per registrare i dati di sonno, attività fisica e salute.
- **FatSecret Server:** Accessibile mediante API, rende disponibile un database contenente cibi e ricette utili per la registrazione dei cibi da parte dei pazienti. (Attualmente non in uso).
- **User Smartphone:** Dispositivo personale del paziente attraverso il quale può utilizzare HealthApp Mobile (attualmente non in uso) oltre che connettersi tramite Bluetooth al Fitbit Wearable.

Lato software:

- **HealthApp Application Server:** Fulcro dell'applicazione, è responsabile di manipolare i dati dell'HealthApp Database, di fornire e ricevere dati dai client e di interfacciarsi ai servizi di terze parti. Le connessioni al server avvengono tramite API.
- **HealthApp Rule Engine:** È un sistema che applica una serie di regole ad un insieme di dati e fornisce dei risultati che vengono utilizzati per produrre le raccomandazioni per i pazienti.

- **HealthApp Database:** Modulo dedito alla gestione, all'organizzazione e all'archiviazione dei dati.
- **HealthApp Web:** Modulo software deployato sull'HealthApp Frontend, utilizzato dai medici.
- **Web Browser:** Modulo attraverso cui i medici si connettono all'HealthApp Frontend.
- **Fitbit Web API:** Artifact utilizzato per interfacciarsi al Fitbit Server, questo permetterà tramite le API di recuperare i dati dei pazienti raccolti tramite il Fitbit Wearable.
- **FatSecret Web API:** Artifact utilizzato per interfacciarsi con il FatSecret Server, questo permetterà tramite le API di recuperare i dati relativi a cibi e ricette. (Attualmente non in uso).
- **HealthApp Mobile:** Applicazione mobile in esecuzione sullo smartphone dei pazienti. (Attualmente non in uso).
- **Fitbit Mobile App:** Applicazione mobile Fitbit, utilizzata dai pazienti per collegare il Fitbit Wearable, questa raccoglierà i dati che verranno poi trasmessi al Fitbit Server.

1.2.1 Backend

Molte delle principali operazioni necessarie affinché un'applicazione possa essere eseguita vengono svolte dal suo backend. In particolare, il backend di un'applicazione, si occupa di gestire le richieste degli utenti, della sicurezza, dell'elaborazione logica, della comunicazione con servizi esterni e dell'interazione con il database. Nel caso di HealthApp, il backend ha il compito di rispondere alle richieste che i vari client effettuano tramite le API REST e di comunicare con i servizi di terze parti, utili per recuperare dati necessari per il corretto funzionamento dell'applicazione, come Fitbit Server e FatSecret

Server. Il backend si può definire il “motore” dell’applicazione, e come tale deve avere determinate caratteristiche:

- **Affidabilità e disponibilità**
- **Scalabilità**
- **Sicurezza**
- **Manutenibilità**

Per i motivi sopra elencati nel progetto HealthApp si è deciso di adottare *Spring*, un framework *Java* associato al linguaggio di programmazione *Kotlin*. *Spring* è un framework che offre sicuramente scalabilità, sicurezza ma anche modularità, in quanto è possibile utilizzare solo le parti del framework necessarie ai nostri scopi. Sarebbe stato possibile utilizzare altri framework, anche più semplici, come *Express* per *Node.js,* questo però avrebbe sicuramente rallentato i tempi di sviluppo poiché non offre sicurezza come *Spring*, quindi, sarebbe stato necessario implementare tutto manualmente.

Un altro framework utilizzato per il nostro backend è *Drools*, esso rappresenta il *Rule Engine*, svolge un’analisi sui dati dei pazienti applicando delle regole definite da noi e restituisce dei risultati che saranno utilizzati per le *recommendations*.

1.2.2 Database

Il database di un’applicazione si occupa di memorizzare i dati e gestirli. La scelta per la nostra applicazione è ricaduta su un Database Relazionale, più precisamente il DBMS PostgreSQL, questo offre affidabilità, robustezza, flessibilità ed è Open Source. È possibile accedere e manipolare i dati tramite query SQL ed è interconnesso al backend tramite internet.

1.2.3 Frontend

Il Presentation tier nella nostra applicazione è diviso in due: web e mobile.

L'applicazione Web è stata sviluppata adottando il framework *React* per *Javascript*, questo offre semplicità nello sviluppo in quanto utilizza una struttura che si basa su componenti *(Component-Based Architecture),* questo consente di creare dei componenti riutilizzabili, rendendo più semplice la manutenibilità del codice.

La Web Application sarà utilizzata principalmente dai medici per il monitoraggio dei pazienti, e dai pazienti solo per l'associazione dell'account Fitbit personale all'account HealthApp.

L'applicazione Mobile è stata sviluppata adottando il framework *React Native* per *Javascript.* Il perché di questa scelta è facilmente intuibile, *React* Native oltre ai pregi di *React* sopra elencati, offre la possibilità di sviluppare una singola applicazione che vada bene sia per *iOS* che per *Android*, ovviamente non senza compromessi. Non tutte le funzionalità sono implementabili, ma quelle supportate sono sufficienti per gli scopi della nostra applicazione.

Non è previsto un utilizzo della Mobile Application da parte dei medici, ma solo da parte dei pazienti, che tramite quest'ultima potranno accedere all'account HealthApp e visualizzare i propri dati. Come già detto in precedenza l'applicazione mobile è momentaneamente in stand-by.

1.3 Organizzazione e pianificazione del progetto

1.3.1 Metodologia

In ambito informatico possiamo fare ricorso a diverse metodologie di sviluppo; tra le più comuni troviamo la metodologia Agile e quella Watefall:

- **Waterfall:** Questa metodologia è caratterizzata da un approccio sequenziale allo sviluppo, le fasi sono distinte e pianificate, la documentazione è approfondita e i requisiti dell'applicazione non possono cambiare nel tempo.

- **Agile** (3)**:** Metodologia più flessibile, lavoro suddiviso in *sprint* con il rilascio di parti di software funzionante in ogni sprint, la documentazione redatta è quella essenziale, c'è una partecipazione attiva del cliente poiché tramite i diversi colloqui si stabiliscono i requisiti, e sono accettabili cambiamenti dei requisiti anche in fasi avanzate del progetto.

Nell'ambito del progetto HealthApp abbiamo deciso di utilizzare la metodologia Agile al fine di avere una maggiore flessibilità, anche a causa dei requisiti necessari che non erano tutti definitivi sin dall'inizio, ma si sono sviluppati gradualmente nel corso dei colloqui con i medici. Si è scelto di utilizzare lo sprint della durata di una settimana, con diversi task assegnati a ciascuno per ogni sprint. Settimanalmente sono stati organizzati degli incontri con il professore (una sorta di Scrum Master) e gli altri membri del gruppo, e giornalmente noi tesisti ci aggiornavamo sul lavoro svolto. Questa metodologia inoltre non richiede una documentazione dettagliata, quindi, è stato possibile concentrarsi sullo sviluppo dell'applicazione, producendo la documentazione essenziale.

1.3.2 Divisione del Lavoro

Il lavoro per questo progetto è stato suddiviso in più parti, poiché sarebbe stato molto complicato per una singola persona gestire tutti gli aspetti dell'applicazione. Più precisamente un tesista per il backend (Io), un tesista per il frontend web e uno per il deployment e la sicurezza (per il momento l'applicazione mobile è in stand-by). Questa suddivisione non è fissa poiché ognuno di noi ha aiutato gli altri quando ne avevano bisogno e svolto task diversi da quelli strettamente competenti al suo lavoro.

2. Manutenzione e Descrizione del Backend

Assicurare la manutenzione adeguata di un'applicazione server già esistente rappresenta un aspetto cruciale per garantirne il corretto funzionamento nel tempo, salvaguardando l'alto livello della sua sicurezza ed aprendo le porte a possibili evoluzioni.

Prima di esplorare in profondità la parte più tecnica e illustrarne tutte le caratteristiche, è fondamentale effettuare un'attenta analisi sulla struttura dell'applicazione.

Spring, il framework scelto per lo sviluppo del backend di HealthApp, semplifica il processo di sviluppo, mettendo a disposizione delle annotazioni, che sono delle parole (diverse in base al tipo di componente) precedute da una @. In particolare, adesso andrò ad analizzare le *Stereotype Annotations* [3], come *@Component* che è l'annotazione base, oppure *@Controller*, *@Service* e *@Repository* [4], che sono delle specializzazioni della classe base. Le *Stereotype Annotations* sono utilizzate per assegnare un particolare ruolo ad un determinato componente.

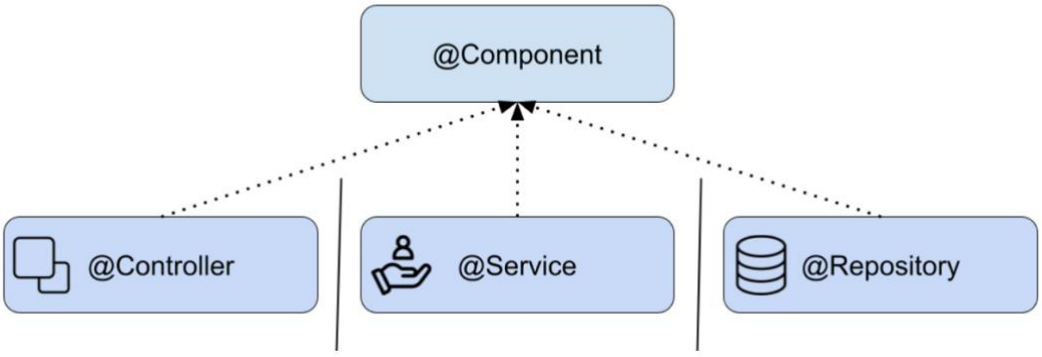

Figura 3 – Annotazione @Componente e specializzazioni

La struttura gerarchica e l'utilità dai tre componenti è la seguente [5]:

- **Controller:** L'annotazione *@Controller* indica che una particolare classe svolge il ruolo di controller, può essere applicato solo alle classi e viene generalmente utilizzata in combinazione con metodi basati

sull'annotazione @*RequestMapping*. Viene utilizzato per contrassegnare una classe come gestore di richieste web, ed è utilizzato principalmente con le applicazioni Spring MVC. Questa annotazione funge da stereotipo per la classe annotata, indicandone il ruolo. Il *dispatcher* esegue la scansione di tali classi annotate per i metodi mappati e rileva le annotazioni @*RequestMapping*. Un controller può anche essere indicato come @*RestController*, questa annotazione permette di lavorare con i JSON invece che con gli HTML [6].

- **Service:** In un'applicazione, la *business logic* si trova all'interno del *service layer*; quindi, utilizziamo l'annotazione @*Service* per indicare che una classe appartiene a quel livello. L'annotazione @*Service* può essere applicata solo alle classi e viene utilizzata per contrassegnare la classe come "fornitore di servizi". Il contesto Spring rileverà automaticamente queste classi quando viene utilizzata la configurazione basata su annotazioni e la scansione del classpath.

- **Repository:** L'annotazione @*Repository* viene utilizzata per indicare che la classe fornisce il meccanismo per le operazioni di archiviazione, recupero, aggiornamento, eliminazione e ricerca sugli oggetti. Le classi *repository* di Spring vengono rilevate automaticamente dal framework Spring tramite la scansione del classpath. Questa annotazione è un'annotazione stereotipata di uso generale molto vicina al modello DAO in cui le classi DAO sono responsabili di fornire operazioni CRUD sulle tabelle del database.

Di seguito possiamo vedere un esempio di implementazione concreta di un @RestController, un @Service e un @Repository. Questi riguardano la gestione delle categorie di cibo nel nostro applicativo. Da qui è possibile visualizzare meglio la gerarchia di questi tre componenti, partendo dal Controller e arrivando al Repository.

```kotlin
@RestController
@RequestMapping("/api/foods/categories")
internal class FoodCategoryController(
    private val foodCategoryService: FoodCategoryService,
) {

    @GetMapping
    fun getAllFoodCategories(): List<FoodCategory> = foodCategoryService.getAllFoodCategories()
}
```

Figura 4 - @RestController

```kotlin
@Service
internal class FoodCategoryServiceImpl(private val foodCategoryRepository: FoodCategoryRepository) :
    FoodCategoryService {

    override fun getAllFoodCategories(): List<FoodCategory> {
        return foodCategoryRepository.findAll()
    }
}
```

Figura 5 - @Service

```kotlin
@Repository
interface FoodCategoryRepository : JpaRepository<FoodCategory, Long> {

    fun findByName(name: String): FoodCategory
}
```

Figura 6 - @Repository

Possiamo notare che la funzione *findAll()* non è presente nel Repository nonostante nel Service venga utilizzata, questo perché la *JpaRepository* ha già all'interno alcuni metodi base per persistere, eliminare e recuperare i dati [7].

2.1 Aggiornamenti e sicurezza

Nel contesto dello sviluppo software, garantire la stabilità e l'efficienza di un'applicazione è molto importante per il mantenimento della stessa a lungo termine; in particolare, con Spring, è fondamentale l'aggiornamento delle

librerie utilizzate, poiché questi aggiornamenti portano con loro correzioni di bug o patch di sicurezza di cui è sempre bene tener conto, soprattutto al giorno d'oggi, in cui la sicurezza informatica è fondamentale.

Un'altra motivazione per cui gli aggiornamenti delle librerie sono molto importanti è che questi agevolano la manutenzione del codice, permettendo a noi sviluppatori di rimanere al passo con le best practice ed evitando l'accumulo di debito tecnico nel codice.

L'aggiornamento principale è stato quello della libreria Spring Boot alla versione 3.0.12, questa ha corretto alcuni bug presenti nella versione precedentemente usata e ha introdotto nuovi metodi rimpiazzandone alcuni di quelli esistenti; quindi, è stato possibile rimuovere i metodi deprecati precedentemente utilizzati, in favore di quelli aggiornati.

Dopo l'aggiornamento di Spring Boot è stato necessario aggiornare anche la configurazione di Spring Security. La versione precedente faceva uso di metodi deprecati, quindi è stata necessaria una revisione completa, per garantire che il sistema rimanesse allineato con le best practice e le raccomandazioni di sicurezza. In particolare, la funzione *securityFilterChain(…)* è stata utilizzata per definire la configurazione di sicurezza.

Altri importanti aggiornamenti sono stati effettuati per le versioni di Java, passando dalla 11 alla 17 e Kotlin, passando alla versione 1.9.20.

Questo processo di aggiornamento non solo ha permesso di eliminare i metodi obsoleti, ma ha consentito di implementare nuove caratteristiche di sicurezza e miglioramenti, garantendo al nostro sistema robustezza e protezione e facendo si che i dati sensibili degli utenti siano protetti da eventuali minacce esterne.

Gli schemi e la documentazione precedentemente presenti sono stati sottoposti a un aggiornamento accurato. Coerentemente con quanto richiede la metodologia Agile, solo la documentazione strettamente necessaria è stata

redatta; nei prossimi paragrafi vedremo le versioni attuali dei diagrammi, che riflettono le modifiche apportate durante gli ultimi mesi e offrono una visione aggiornata e completa del sistema; queste non solo facilita il processo di manutenzione per il gruppo attuale, ma saranno una guida importante per i futuri sviluppatori. Il diagramma architetturale è già stato presentato nel capitolo precedente; pertanto, non sarà riproposto.

2.1.1 Diagramma di Controllo degli Accessi di HealthApp

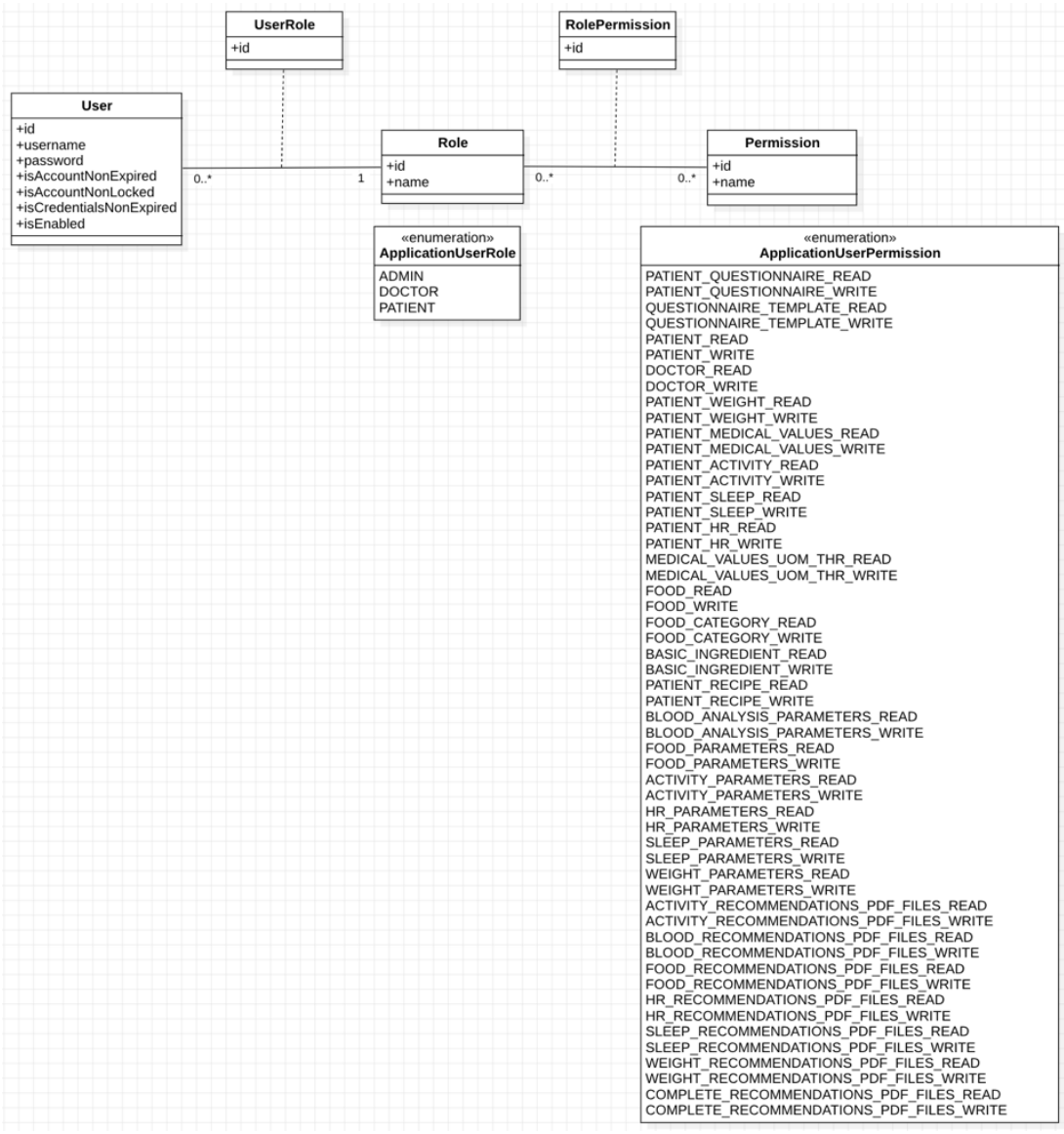

Figura 7 – Diagramma di controllo degli accessi di HealthApp

Il diagramma di controllo degli accessi di HealthApp definisce i ruoli e i permessi nel contesto dell'applicazione. Ogni utente è identificato da un id, un nome utente (username), da una password, e da altri campi relativi alla validità dell'account, ed è associato ad un singolo ruolo; ogni ruolo conferisce specifici permessi di accesso (lettura o scrittura) da parte dell'utente a cui è assegnato alle entità presenti nel database dell'applicazione. I permessi consentono l'accesso in lettura o scrittura a determinate entità presenti nel database, definendo così i limiti di autorizzazione per ciascun utente in base al ruolo assegnato. Questo sistema di definizione dei ruoli assicura che gli utenti possano accedere solo alle informazioni e alle funzionalità pertinenti al loro ruolo all'interno di HealthApp, garantendo la sicurezza e la privacy delle informazioni sensibili.

Analizzando il diagramma rappresentato sopra possiamo notare che i ruoli contemplati all'interno della nostra applicazione sono:

- **ADMIN:** Amministratore della nostra applicazione, ha il massimo controllo del sistema, ha la possibilità di accedere a tutte le informazioni sia in scrittura che in lettura.

- **DOCTOR:** I medici hanno la possibilità di creare, modificare o eliminare gli account dei pazienti, possono aggiungere le misurazioni del peso e i valori delle analisi mediche dei pazienti, possono accedere ai dati dei pazienti per monitorarli e possono definire le soglie che saranno poi utilizzate per applicare le regole che serviranno a creare le recommendations.

- **PATIENT:** Gli account di tipo paziente hanno accesso in lettura e scrittura ai propri dati, hanno la possibilità di inserire ricette, cibi consumati, peso e analisi mediche nonché di accedere ai dati relativi a sonno, attività fisica e salute, oltre alla possibilità di accedere in sola lettura ai file pdf delle recommendations.

2.1.2 HealthApp Database

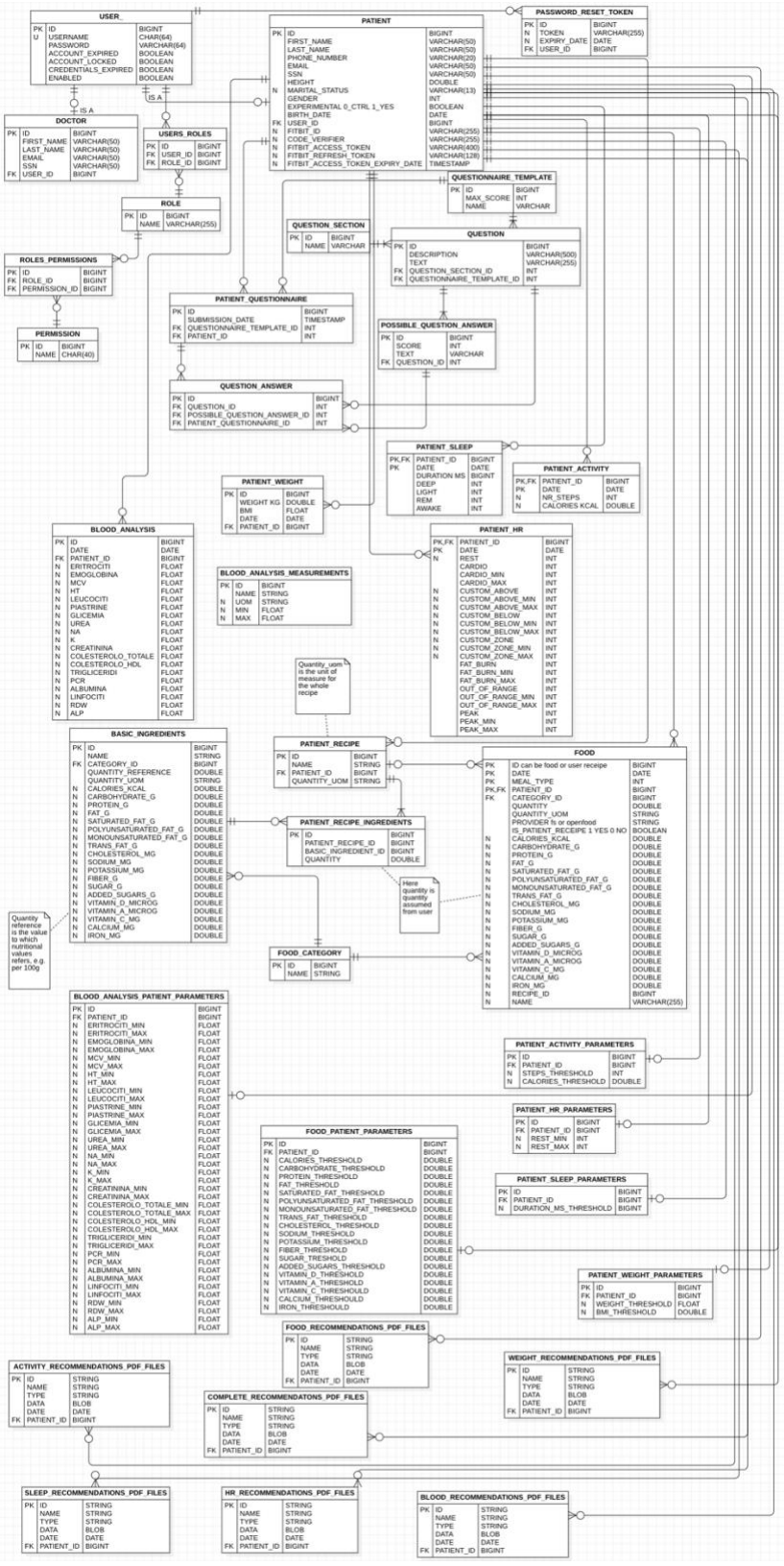

Figura 8 – Schema HealthApp Database

Il database di HealthApp è rappresentato nello schema qui sopra; si può notare come la nostra base dati sia abbastanza complessa, con tante entità in gioco e tante relazioni di cui preoccuparsi.

In particolare, possiamo notare che nel nostro database sono presenti 38 tabelle, ognuna corrispondente ad un'entità del nostro sistema.

Di seguito una descrizione di ogni tabella:

- **USER_:** Rappresenta l'utente, contiene le info relative allo username e alla password oltre ad altri campi relativi allo stato dell'account.
- **PATIENT:** Contiene le informazioni relative al paziente, come nome, cognome, e tutte le altre info inserite dal medico in fase di creazione dell'account, nonché i dati relativi all'account Fitbit collegato se presenti.
- **DOCTOR:** Contiene le informazioni relative al dottore come nome, cognome e tutte le altre info che l'admin ha inserito in fase di creazione dell'account.
- **ROLE:** Contiene le informazioni relative ai ruoli.
- **PERMISSION:** Contiene le informazioni relative ai permessi.
- **ROLES_PERMISSIONS:** Contiene le informazioni relative ai permessi associati a ciascun ruolo.
- **USER_ROLES:** Contiene le informazioni relative al ruolo di ciascuno user.
- **PASSWORD_RESET_TOKEN:** Contiene le informazioni relative ai token per il reset della password.
- **QUESTIONNAIRE_TEMPLATE:** Contiene le informazioni relative ai questionari quali nome e punteggio massimo.

- **QUESTION_SECTION:** Contiene le informazioni relative ai nomi delle varie sezioni dei questionari.
- **QUESTION:** Contiene le informazioni relative alle domande quali la domanda, la descrizione, e gli id di questionario e sezione.
- **POSSIBLE_QUESTION_ANSWER:** Contiene le informazioni relative alle possibili risposte alle domande, quali la riposta, il punteggio e l'id della domanda.
- **PATIENT_QUESTIONNAIRE:** Contiene le informazioni relative ai questionari compilati dai pazienti quali punteggio, data, id del paziente e id del questionario.
- **QUESTION_ANSWER:** Contiene le informazioni relative alle risposte date dai pazienti a cui il questionario è stato sottoposto, in particolare id del questionario, id della domanda e id della possibile risposta.
- **FOOD:** Contiene le informazioni relative ai cibi consumati dai pazienti.
- **FOOD_CATEGORY:** Contiene le informazioni relative alle categorie di cibo.
- **PATIENT_RECIPE:** Contiene le informazioni relative alle ricette inserite dai pazienti.
- **PATIENT_RECIPE_INGREDIENTS:** Contiene le informazioni relative agli ingredienti delle ricette inserite dai pazienti.
- **FOOD_PATIENT_PARAMETERS:** Contiene le informazioni relative alle soglie dei valori nutrizionali definite per ogni paziente.
- **BASIC_INGREDIENTS:** Contiene le informazioni relative ai valori nutrizionali di diversi ingredienti base.

- **FOOD_RECOMMENDATIONS_PDF_FILES:** Contiene i file pdf con le recommendations relative al cibo.
- **PATIENT_ACTIVITY:** Contiene le informazioni relative al movimento dei pazienti quali data, numero di passi e calorie bruciate.
- **PATIENT_ACTIVITY_PARAMETERS:** Contiene le informazioni relative alle soglie di attività definite per ogni paziente.
- **ACTIVITY_RECOMMENDATIONS_PDF_FILES:** Contiene i file pdf con le recommendations relative all'attività fisica.
- **PATIENT_HR:** Contiene le informazioni relative alla frequenza cardiaca dei pazienti.
- **PATIENT_HR_PARAMETERS:** Contiene le informazioni relative alle soglie per la frequenza cardiaca definite per ogni paziente.
- **HR_RECOMMENDATIONS_PDF_FILES:** Contiene i file pdf con le recommendations relative alla frequenza cardiaca.
- **PATIENT_SLEEP:** Contiene le informazioni relative al sonno dei pazienti, durata e fasi.
- **PATIENT_SEEP_PARAMETERS:** Contiene le informazioni relative alle soglie di sonno definite per ogni paziente.
- **SLEEP_RECOMMENDATIONS_PDF_FILES:** Contiene i file pdf con le recommendations relative al sonno.
- **PATIENT_WEIGHT:** Contiene le informazioni relative al peso dei pazienti, in particolare: data, peso e bmi.
- **PATIENT_WEIGHT_PARAMETERS:** Contiene le informazioni relative alle soglie per il peso e per il bmi definite per ogni paziente.

- **WEIGHT_RECOMMENDATIONS_PDF_FILES:** Contiene i file pdf con le recommendations relative al peso e al bmi.
- **BLOOD_ANALYSIS_MESUREMENTS:** Contiene le informazioni relative alle soglie standard delle analisi del sangue.
- **BLOOD_ANALYSIS:** Contiene le informazioni relative alle analisi del sangue dei pazienti.
- **BLOOD_ANALYSIS_PATIENT_PARAMETERS:** Contiene le informazioni relative alle soglie per i valori delle analisi del sangue definite per ogni paziente.
- **BLOOD_RECOMMENDATIONS_PDF_FILES:** Contiene i file pdf con le recommendations relative alle analisi del sangue.
- **COMPLETE_RECOMMENDATIONS_PDF_FILES:** Contiene i file pdf con le recommendations complete.

2.1.3 Documentazione delle API

Il backend di HealthApp conta attualmente 101 API, che permettono di accedere a tutte le funzionalità presenti nel sistema. La documentazione delle API è stata recentemente arricchita e migliorata per garantire una comprensione ancora più approfondita e chiara delle funzionalità offerte dal sistema. In particolare, ogni API ora include un *summary* e una descrizione del codice di risposta, fornendo informazioni specifiche sulle finalità e sul funzionamento. Questo permette agli sviluppatori di acquisire in modo rapido una comprensione chiara di quali operazioni sono supportate, quali sono i parametri necessari e quale tipo di risposta aspettarsi per ogni API. Realizzare questa integrazione delle descrizioni nella documentazione utilizzando Swagger rappresenta un notevole progresso nel miglioramento dell'accessibilità e della facilità d'uso del sistema (4).

In particolare, il *summary* è stato aggiunto attraverso l'utilizzo dell'annotazione *@Operation* su ciascuna API, mentre la descrizione del codice di risposta attraverso l'annotazione *@ApiResponse*.

È possibile, per noi sviluppatori, visualizzare una versione aggiornata della documentazione delle API presenti nel nostro sistema all'indirizzo *https://dev.verona-experiment.org/swagger-ui/index.html,* dopo aver effettuato l'accesso come admin. Dalla dashboard di Swagger è inoltre possibile testare le API per verificarne il corretto funzionamento.

Di seguito è riportato un esempio di API visualizzabile sulla dashboard di Swagger.

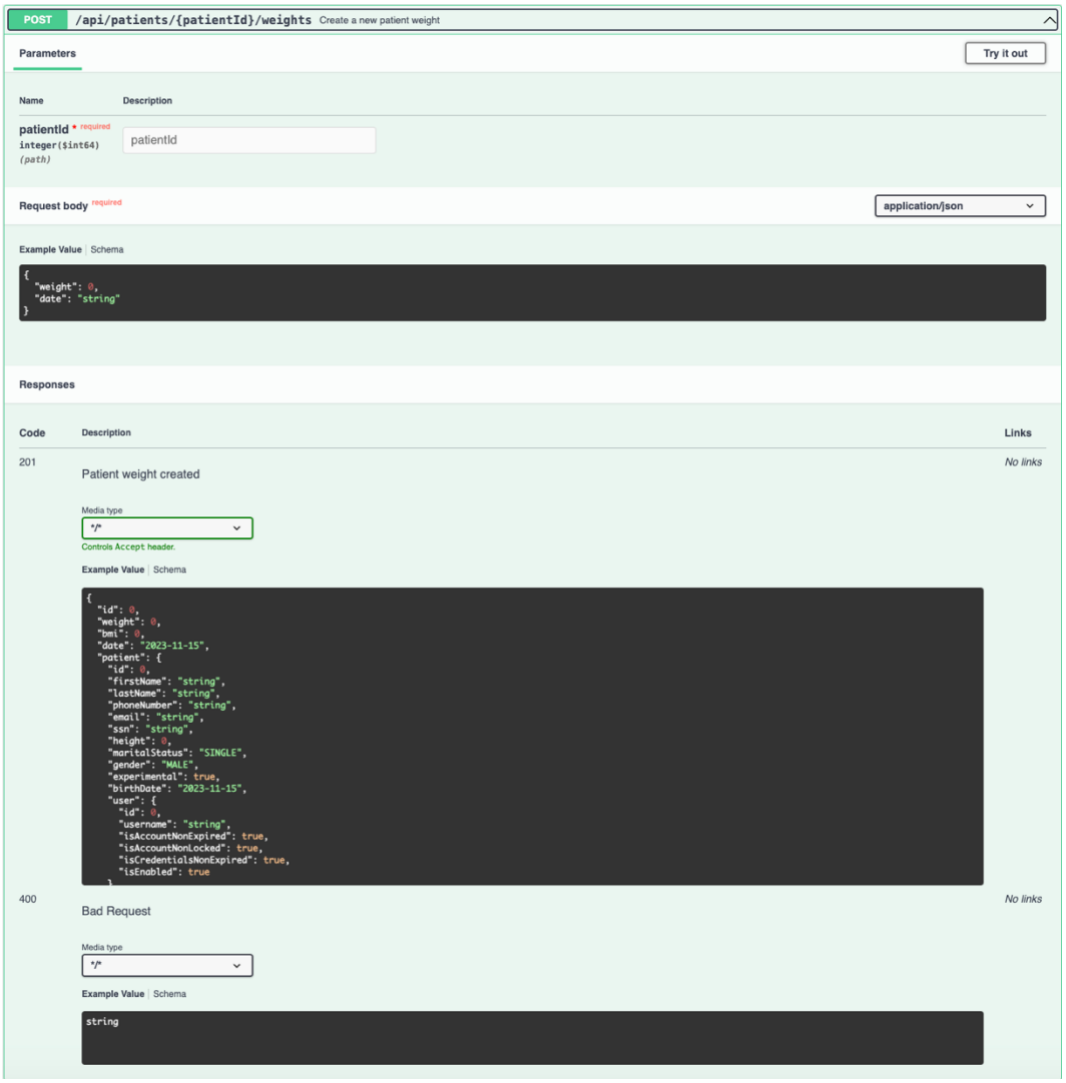

Figura 9 – Esempio Dashboard Swagger

2.2 Funzionalità Principali

Tra le funzionalità principali della nostra applicazione abbiamo la creazione di account di tipo DOCTOR e la creazione di questionari da parte dell'ADMIN, la creazione di account di tipo PATIENT da parte dei DOCTOR e l'accesso ai dati del paziente sia da parte del paziente stesso che da parte dei dottori. Vedremo nello specifico le funzionalità più significative nei prossimi paragrafi.

Una funzionalità molto utile di HealthApp è la possibilità da parte dei medici di somministrare dei questionari ai pazienti, senza dover ricorrere all'utilizzo di siti esterni o di materiale cartaceo. I questionari attualmente presenti nella nostra base dati sono i seguenti:

- **MEDAS**
- **CEREALI**
- **INSOMNIA SEVERITY INDEX**
- **INDICE PROGNOSTICO MULTIDIMENSIONALE (MPI)**

Questi questionari sono utili ai medici per avere un quadro completo sullo stato di salute e sullo stile di vita dei pazienti all'inizio dell'esperimento ma anche ad esperimento in corso e alla fine, in quanto i questionari possono essere ripetuti più volte.

Come accennato in precedenza, HealthApp è concettualmente divisa in quattro macroaree:

- **Sonno**
- **Attività Fisica**
- **Salute**
- **Alimentazione**

Di seguito vedremo un'overview delle principali funzionalità relative alle diverse macroaree.

2.2.1 Sonno

I dati relativi al sonno dei pazienti vengono acquisiti dal dispositivo indossabile Fitbit e, come vedremo nel capitolo 5, vengono successivamente recuperati dal backend di HealthApp ed archiviati nell'HealthApp database. Queste informazioni sul sonno sono destinate all'analisi e al monitoraggio da parte del personale medico; infatti, gli specialisti, utilizzando tali dati, possono valutare il tempo e la qualità del sonno dei pazienti, nonché l'andamento temporale dello stesso. Inoltre, il sistema di raccomandazione (recommender system) integrato, come vedremo nello specifico nel capitolo 4, analizza i dati relativi al sonno, li correla con le soglie specifiche impostate per ciascun paziente e genera raccomandazioni specifiche inerenti a questa macroarea, le quali sono rese accessibili ai medici e ai pazienti. È possibile accedere ai dati relativi al sonno attraverso le relative API messe a disposizione dal backend di HealthApp, in particolare si possono ottenere i dati relativi ad un intervallo di date selezionato.

2.2.2 Attività Fisica

I dati relativi all'attività fisica dei pazienti sono sempre raccolti tramite il dispositivo Fitbit e memorizzati nell'HealthApp database. Per questa macroarea abbiamo due tipi di dati:

- **Numero di passi**
- **Calorie bruciate**

Questi parametri forniscono una misura quantitativa delle attività quotidiane dei pazienti, consentendo al personale medico di valutare in modo efficace

l'evoluzione delle abitudini e dei livelli di attività dei pazienti nel corso del tempo. Anche per questa macroarea c'è la possibilità di definire delle soglie personalizzate in base alle esigenze di ciascun paziente, che vengono utilizzate dal sistema di recommendations per analizzare i dati e produrre le relative raccomandazioni. È possibile accedere ai dati relativi all'attività fisica attraverso le relative API messe a disposizione dal backend di HealthApp, in particolare si può scegliere di ottenere i dati relativi al numero di passi o le calorie bruciate in un intervallo di date selezionato.

2.2.3 Salute

La macroarea salute è più ampia rispetto alle due analizzate in precedenza, poiché vengono presi in considerazione sia dati provenienti da Fitbit, sia dati manualmente inseriti dai medici o dai pazienti; in particolare abbiamo i seguenti dati:

- **Frequenza cardiaca:** I dati relativi alla frequenza cardiaca vengono raccolti tramite Fitbit e memorizzati nell'HealthApp database, questi permettono ai dottori di monitorare due aspetti fondamentali:
    - **Frequenza cardiaca a riposo:** Il valore recuperato Fitbit è la media giornaliera dei battiti a riposo.
    - **Zone di frequenza cardiaca:** Caratterizzate da valori minimi e massimi di frequenza cardiaca. Per ciascuna zona, sono registrati i minuti durante i quali il paziente ha mantenuto una frequenza cardiaca che ricade in quella soglia.

Grazie ai dati raccolti è possibile monitorare l'andamento temporale dei pazienti, nonché ottenere recommendations personalizzate per ogni paziente sulla base delle soglie inserite per la frequenza cardiaca a riposo. È possibile accedere ai dati relativi alla frequenza cardiaca attraverso le relative API

messe a disposizione dal backend di HealthApp, in particolare è possibile ottenere i dati relativi ad un intervallo di date selezionato.

- **Analisi del sangue:** I dati relativi alle analisi del sangue possono essere inseriti manualmente nel sistema sia dal medico che dal paziente. Questo comporta una grande flessibilità e permette al medico di avere sempre accesso alle analisi dei pazienti, comprese quelle effettuate in passato. Tale approccio elimina la necessità di portare documenti cartacei, offrendo una soluzione più moderna ed efficiente. Inoltre, avendo accesso a tutte le analisi del paziente, il medico ha la possibilità di monitorarne i progressi. Anche per questa macroarea è possibile definire delle soglie che verranno utilizzate dal recommender system per analizzare i dati e produrre delle recommendations per ogni paziente. È possibile inserire, modificare, eliminare e recuperare le analisi attraverso le relative API messe a disposizione dal backend.

- **Peso:** I dati relativi al peso, similmente alle analisi del sangue, possono essere inseriti manualmente sia dal medico che dal paziente. Anche in questo caso il medico ha la possibilità di monitorare l'andamento temporale del peso del paziente. Inoltre, il sistema è progettato per calcolare automaticamente il BMI (Body Mass Index) basato sul peso fornito e sull'altezza del paziente (fornita in fare di creazione dell'account) tramite la seguente formula [8]:

$$BMI = \frac{Massa\ (kg)}{[Altezza(m)]^2}$$

Formula 1 – Calcolo BMI

Il valore calcolato, insieme al valore del peso inserito vengono memorizzati nell'HealthApp database. Anche per il peso e per il BMI c'è la possibilità di definire delle soglie personalizzate per ogni paziente,

che verranno utilizzate dal recommender system per produrre delle raccomandazioni per ogni paziente. È possibile aggiungere, modificare, eliminare e recuperare i dati relativi a peso e BMI dei pazienti attraverso le relative API messe a disposizione dal backend.

2.2.4 Alimentazione

I dati relativi all'alimentazione sono inseriti manualmente nel sistema dal paziente, garantendo una notevole flessibilità e permettendo a quest'ultimo e ai medici di tenere traccia dettagliata dei consumi. Questo offre un metodo moderno ed efficiente per registrare e monitorare l'alimentazione quotidiana.

Attraverso l'integrazione con FatSecret, come vedremo nel capitolo 5, i valori nutrizionali associati a ciascun alimento inserito dal paziente vengono automaticamente salvati nel database. Inoltre, il paziente ha la possibilità di creare ricette personalizzate selezionando dai 568 record di alimenti di base presenti nel nostro database. L'interoperabilità tra i piatti di FatSecret e le ricette create dall'utente è assicurata dal fatto che vengono memorizzati gli stessi valori nutrizionali, con un valore salvato nel database che distingue se si tratta di una ricetta personale o di dati provenienti da FatSecret.

Questa funzionalità consente ai medici di monitorare in modo efficace l’alimentazione dei pazienti e, anche in questo caso, possono essere definite delle soglie personalizzate utilizzate dal recommender system per produrre le raccomandazioni per ciascun paziente. È possibile inserire, aggiornare, eliminare e recuperare i dati relativi all’alimentazione utilizzando le API dedicate.

Il modulo relativo all'alimentazione attualmente non è in uso, poiché era originariamente integrato con l'applicazione mobile dedicata esclusivamente ai pazienti. Tuttavia, quest'applicazione è stata dismessa poiché l’esperimento attuale non era idoneo al suo utilizzo. Nonostante ciò, le funzionalità legate all'alimentazione sono state conservate e aggiornate nel backend. Questa

strada è stata seguita per garantire che, se in futuro dovesse esserci la necessità di utilizzare nuovamente l'applicazione mobile per un altro esperimento, eventuali modifiche necessarie sarebbero limitate esclusivamente a quest'ultima.

Per una descrizione più dettagliata e aggiornata di tutte le API disponibili nel nostro sistema si rimanda alla dashboard di Swagger, consultabile all’indirizzo:

https://dev.verona-experiment.org/swagger-ui/index.html

3. Evoluzione del Backend

Durante questo periodo in cui ho lavorato su questo progetto sono stati necessari dei cambiamenti e delle aggiunte al backend esistente. Oltre a delle aggiunte e delle modifiche alla struttura del database, c'è stata la necessità di aggiungere alcune funzionalità che in precedenza non esistevano e soprattutto si è deciso di effettuare una migrazione del database, passando da un semplice file ad un sistema di gestione di database quale PostgreSQL.

3.1 Sviluppi e Aggiunte al Backend

Nel Corso dello sviluppo del nostro sistema, abbiamo continuamente lavorato per migliorarne l'usabilità e l'efficienza, soprattutto a seguito dei colloqui con i medici. Nei prossimi paragrafi esploreremo le nuove funzionalità che sono state introdotte nel backend, tra cui l'eliminazione dei pazienti e dei medici, l'espansione delle informazioni nell'entità delle analisi del sangue e le modifiche apportate alle fasi del sonno; discuteremo dei cambiamenti apportati alla gestione dei questionari compilati dai pazienti e parleremo brevemente delle nuove regole implementate per il sistema di recommendations che poi verranno trattate più nel dettaglio nel prossimo capitolo.

3.1.1 Eliminazione dei Pazienti e dei Medici

Abbiamo introdotto la possibilità di eliminare i pazienti e i medici dal sistema. Questa funzionalità è stata implementata oltre che per garantire la pulizia e l'organizzazione del nostro database anche per questioni di GDPR e rispetto della privacy. Per offrire queste funzionalità sono state implementate due nuove API che hanno una funzione complessa, poiché oltre all'eliminazione degli utenti, devono garantire anche l'eliminazione di tutti i dati ad essi associati, come ad esempio i dati ricevuti da Fitbit nel caso dei pazienti. Se

durante il processo di eliminazione qualcosa dovesse andare storto è garantita la transazionalità dell'operazione, il che significa che o tutti i dati vengono eliminati oppure nessun dato viene eliminato.

In particolare, le due nuove API sono le seguenti:

- **DELETE /api/patients/{patientID}:** Riceve l'id del paziente ed elimina il paziente e tutti i dati ad esso associati.
- **DELETE /api/doctor/{doctorID}:** Riceve l'id del dottore ed elimina il dottore e tutti i dati ad esso associati.

Ognuna di queste API ha un Service associato che si occupa di eliminare tutti i dati associati a quell'utente e successivamente l'utente stesso.

3.1.2 Espansione Informazioni Analisi del Sangue

Abbiamo ampliato le informazioni nell'entità delle analisi del sangue per fornire una visione più dettagliata dello stato di salute del paziente. In particolare, i nuovi campi aggiunti includono:

- **Albumina** [9]: È la proteina più abbondante presente nel plasma. Viene prodotta dal fegato e svolge diverse funzionalità. La presenza di albumina nel sangue è utilizzata come indicatore dello stato nutrizionale dei pazienti ospedalizzati, così come della funzionalità renale ed epatica.
- **Linfociti** [10]**:** I linfociti costituiscono circa il 20% - 40% dei globuli bianchi e sono presenti sia nel sangue che nel sistema linfatico. Livelli anomali possono indicare infezioni o malattie autoimmuni, quindi è consigliabile tenerli sotto controllo.
- **RDW (Red Cell Distribution Width)** [11]**:** Questo valore misura la variazione delle dimensioni dei globuli rossi nel sangue. Un RDW

elevato può indicare diverse condizioni, comprese anemie e carenze vitaminiche.

- **ALP (Alkaline Phosphatase)** [12]**:** Questo enzima è diffuso in vari tessuti corporei, con abbondante presenza sia nelle ossa che nel fegato. L'analisi della concentrazione della fotofasi alcalina nel sangue viene utilizzata per diagnosticare disturbi del fegato, in particolare delle vie biliari e delle ossa.

3.1.3 Modifiche alle Fasi del Sonno

Per rendere l'analisi del sonno più dettagliata abbiamo ampliato le informazioni relative a questo parametro del paziente includendo la durata delle varie fasi del sonno [13], tra cui:

- **Deep Sleep:** Questa è la fase più profonda del sonno, quella in cui il riposo è maggiormente efficace, essenziale per il riposo fisico e mentale.
- **Light Sleep:** Questa fase rappresenta una transizione tra la veglia e il sonno profondo, il sonno inizia a diventare effettivo.
- **REM (Rapid Eye Movement) Sleep:** In questa fase c'è un'attività cerebrale piuttosto intensa, si svolgono i sogni più vividi e la memoria a breve termine viene consolidata.
- **Awake:** Questo rappresenta il periodo in cui il paziente è sveglio durante la notte o è in fase di addormentamento.

3.1.4 Modifiche ai Questionari dei Pazienti

Abbiamo deciso di eliminare il campo "Descrizione" dai questionari compilati dai pazienti, poiché abbiamo ritenuto che questo campo non contribuisse significativamente alla raccolta di informazioni utili per la diagnosi e il monitoraggio. Abbiamo riflettuto su come semplificare i questionari potesse migliorare l'esperienza utente, quindi eliminando i campi superflui, è stato possibile concentrarci su dati più rilevanti.

3.1.5 Modifiche nella Gestione dei Medici e dei Pazienti

Abbiamo adottato un approccio più flessibile nella gestione dei pazienti e dei medici nel nostro sistema. Invece di associare un paziente a un singolo medico, tutti i medici possono ora monitorare tutti i pazienti. Questa decisione è stata presa in virtù del fatto che l'esperimento conterà circa 100 pazienti e 3 medici, quindi, non essendo numeri troppo elevati si è optato per una gestione flessibile. Il cambiamento, oltre che semplificare leggermente lo schema del database, permette anche di migliorare la collaborazione tra i medici e garantire un'assistenza migliore per i pazienti.

3.1.6 Nuove Regole per il Sistema di Recommendations

Abbiamo implementato nuove regole nel sistema di raccomandazioni per tener conto dei nuovi dati raccolti per le analisi del sangue analizzati in precedenza.

In particolare:

- **Regole per Albumina**
- **Regole per Linfociti**
- **Regole per RDW**

- **Regole per ALP**

Queste regole consentono all'applicativo di fornire raccomandazioni più accurate e personalizzate basate sui dati più recenti e completi disponibili nel sistema. Il Business Rule Engine e tutte le regole ad esso associate saranno analizzati nel dettaglio nel prossimo capitolo.

In questi paragrafi, abbiamo esplorato le nuove funzionalità introdotte nel nostro backend, che includono l'eliminazione dei pazienti e dei medici, l'espansione delle informazioni nelle analisi del sangue, le modifiche alle fasi del sonno e le nuove regole per il sistema di raccomandazioni. Questi miglioramenti sono stati progettati per garantire che il nostro sistema fornisca un servizio sempre più completo, accurato ed efficiente agli utenti.

3.2 Migrazione del Database

Originariamente il database era un semplice file integrato nel backend dell'applicazione. Tuttavia, durante l'evoluzione del progetto, è stato deciso di separare il database dal backend principale. Attualmente il sistema di gestione dei dati è stato migrato a PostgreSQL e deployato come un'istanza separata, migliorando così la scalabilità, l'affidabilità e le prestazioni complessive del sistema.

3.2.1 Scelta del Database

Per la scelta del database la domanda a cui bisogna rispondere inizialmente è: Database SQL o NoSQL? [14]. Per rispondere a questa domanda bisogna analizzare i due tipi di database e vedere quale si addice di più agli scopi della nostra applicazione [15].

Feature	SQL Databases	NoSQL Databases
Key Focus	Reducing data duplication	Scaling and rapid application change
Data Storage Model	Tables with fixed rows and colums	Document: JSON documents; Key-value: key-value pairs; Wide-column: tables with rows and dynamic columns; Graph: nodes and edges
Schemas	Rigid	Flexible
Data to Object Mapping	Requires ORM (object-relational mapping)	Typically doesn't require ORMs. E.g. MongoDB documents map directly to data structures in popular programming languages.
Scaling	Vertical (scale-up with a larger server)	Horizontal (scale-out across commodity servers)

Figura 10 – SQL vs NoSQL

I database SQL (Structured query language) sono i più diffusi e sono costituiti da tabelle, righe, chiavi e relazioni tra tabelle. Eseguire operazioni su questo tipo di database è semplice, ma presenta dei limiti quando si tratta di gestire informazioni non strutturate. Se non conosciamo in anticipo i dati da memorizzare, è necessario prima trasformarli in un formato “standard” e poi inserirli nel database SQL. In un database NoSQL, al contrario, possiamo memorizzare i dati senza la necessità di definire un modello specifico in anticipo. Un altro vantaggio dei database NoSQL è che sono particolarmente adatti per gestire grandi quantità di dati. Tuttavia, interrogare i database NoSQL è più complesso rispetto ai database SQL.

Considerando attentamente queste caratteristiche e valutando le dimensioni moderate del nostro database insieme alla natura dei dati che trattiamo, abbiamo deciso di optare per un database SQL. Questa scelta ci consente di gestire efficacemente i dati nella nostra applicazione senza affrontare la complessità aggiuntiva associata all’uso di un database NoSQL.

Inoltre, va considerato il contesto preesistente: nel nostro caso, era già presente un file di database SQL. Questo ha pesato nella nostra decisione poiché la migrazione da un sistema SQL esistente a un database NoSQL sarebbe stata più complicata e avrebbe richiesto notevole sforzo di trasferimento dei dati e adattamento delle query.

Optare per un database SQL, in questo contesto, si è rivelato non solo pratico, ma anche efficiente, poiché ci ha permesso di continuare a lavorare con la struttura esistente, riducendo al minimo le interruzioni e semplificando il processo complessivo di gestione dei dati.

Dopo aver deciso che tipo di database utilizzare, ci siamo trovati di fronte a diverse opzioni trai vari sistemi di gestione di database relazionali (RDBMS) SQL disponibili. Fin dall'inizio abbiamo optato per l'uso di un sistema open source gratuito.

Tra i più diffusi abbiamo [16]:

- **MySQL:** Database più popolare e utilizzato nel mondo dello sviluppo web, maturo, robusto e stabile.
- **MariaDB:** Sistema di gestione di database relazionali, molto simile a MySQL.
- **PostgreSQL:** Database relazionale a oggetti. È il DBMS open source più avanzato che esista, con funzionalità paragonabili a quelli commerciali.

Dopo aver valutato le diverse opzioni disponibili abbiamo scelto di utilizzare PostgreSQL. Questa decisione è stata presa poiché questo DBMS è il più avanzato tra quelli open source e perché essendo largamente impiegato ha una community molto forte.

3.2.2 PostgreSQL

PostgreSQL (5) rappresenta un sistema di gestione di database relazionale a oggetti open source estremamente potente, che sfrutta ed estende il linguaggio SQL, integrandolo con una vasta gamma di funzionalità progettate per gestire con sicurezza i carichi di lavoro più complessi. La sua storia affonda

le radici nel lontano 1986 e nel corso degli oltre 35 anni di sviluppo attivo, ha acquisito una solida reputazione.

Questo database si distingue per la sua architettura consolidata, l'affidabilità, l'integrità dei dati, un ricco set di funzionalità, l'estensibilità e l'impegno costante della fervente comunità open source che lo supporta. Grazie a questa dedizione, PostgreSQL continua a offrire soluzioni innovative e performanti nel campo della gestione dei dati. È compatibile con tutti i principali sistemi operativi, rispetta gli standard ACID fin dal 2001 e dispone di componenti aggiuntivi potentissimi. Non è quindi una sorpresa che PostgreSQL sia diventato la scelta prediletta di molte persone e organizzazioni quando si tratta di database relazionali open source [16].

3.2.3 Transizione

Per effettuare questa transizione è stato necessario apportare alcune modifiche al file *application.properties,* settando alcune proprietà in modo da far comunicare la nostra applicazione con il nuovo database.

```
# Database
# Address of the DB (kubernetes service).
spring.datasource.url=jdbc:postgresql://${DATABASE_ADDRESS}/health-app-db
# Postgres DB username.
spring.datasource.username=${POSTGRES_USER}
# Postgres DB password.
spring.datasource.password=${POSTGRES_PASSWORD}
# Suppress DB logs.
spring.jpa.show-sql=false
# The DB initialization is handled by the DB container itself.
spring.sql.init.mode=never
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.type.preferred_instant_jdbc_type=TIMESTAMP
spring.liquibase.change-log=classpath:/db/schema.yaml
```

Figura 11 – application.properties collegamento Database

3.3 Risoluzione dei bug

Nel contesto delle migliorie apportate, oltre alle modifiche e agli aggiornamenti al backend descritti in precedenza, è importante sottolineare che sono stati risolti diversi bug.

In particolare, alcuni dei problemi che sono stati risolti sono i seguenti:

- Inconsistenza di nomenclatura per alcuni parametri delle analisi del sangue tra frontend e backend che non permetteva di salvare correttamente i dati.
- Inconsistenza di nomenclatura per alcuni parametri del peso tra frontend e backend che non permetteva di salvare correttamente i dati.
- Problema con API di modifica del peso di un paziente per una determinata data, non funzionava correttamente poiché la data non veniva considerata tra i parametri ricevuti.
- Problema di aggiornamento dei dati provenienti da Fitbit.

3.4 Osservazioni

Tutti questi interventi hanno contribuito a migliorare la stabilità, l'affidabilità e le prestazioni complessive del sistema, contribuendo a un'esperienza utente più fluida e più sicura. Grazie a questi interventi, l'applicazione ora è funzionante in ogni sua parte, poiché i bachi conosciuti sono stati risolti. Tuttavia, è fondamentale tenere presente che, nonostante gli sforzi, i bug possono emergere in qualsiasi momento e vengono affrontati di conseguenza per garantire la migliore esperienza possibile agli utenti; infatti, in ambito informatico è solito affermare che non esistono programmi privi di bachi ma che i bachi presenti non sono stati ancora scoperti.

4. Descrizione del Sistema di Recommendations

In questo capitolo analizzeremo il sistema di recommendations integrato nella nostra applicazione.

I recommender system sono sistemi che al giorno d'oggi stanno riscuotendo molto successo, sia in ambito web, dove facilitano la raggiungibilità dei prodotti, sia negli altri ambiti, in particolare in quello sanitario, dove fungono da strumento di supporto per le decisioni dei professionisti medici (6).

Il sistema di recommendations di HealthApp è un modulo molto importante del nostro applicativo, questo svolge un ruolo cruciale nell'analisi dei dati, fornendo risultati significativi che aiutano sia i pazienti che i medici a tenere sotto controllo lo stato di salute e lo stile di vita dei pazienti. Grazie a questo sistema, è possibile ottenere raccomandazioni personalizzate basate sui dati raccolti, offrendo ai medici importanti informazioni per prendere decisioni informate. Allo stesso tempo, i pazienti possono beneficiare di queste informazioni per migliorare e mantenere il proprio stato di salute e il proprio stile di vita. È importante sottolineare che queste recommendations non si possono in nessun caso considerare sostitutive al parere esperto di un medico. Una cosa fondamentale di questo sistema di recommendations è che questo è personalizzabile per ogni paziente, perché il medico decide per ciascuno le soglie che più sono indicate in base ai colloqui condotti con i pazienti e alle condizioni cliniche di questi ultimi.

Per la realizzazione di questo modulo si è deciso di utilizzare un Business Rule Engine, questo ci permette di offrire un servizio di alta qualità, personalizzato e basato su evidenze, sia per i medici che per i pazienti. Di seguito verrà analizzato che cos'è un Business Rule Engine e quale di quelli esistenti è stato adottato per HealthApp.

4.1 Business Rule Engine

4.1.1 Che cos'è un Business Rule Engine?

Un Business Rule Engine rappresenta un sistema software progettato per eseguire una o più business rules su determinati dati ricevuti in input e restituire un risultato come output [17].

Le regole, anche chiamate *rules,* non sono altro che un flusso di esecuzione strutturato secondo il modello ECA (Event-Condition-Action) [18].

- **Event:** Specifica il segnale che attiva l'invocazione della regola.
- **Condition:** È un test logico che se soddisfatto determina l'esecuzione della regola.
- **Action:** La conseguenza di una condition soddisfatta che produce un output.

Ciascuna regola può contenere una o più condizioni relative a uno o più parametri di una specifica macroarea. Queste condizioni, verificate contemporaneamente, determinano la generazione di un consiglio da presentare al paziente. Inoltre, esistono regole più complesse che coinvolgono diverse macroaree simultaneamente, come ad esempio Movimento e Alimentazione. Le condizioni sono basate su soglie che possono rappresentare limiti superiori o inferiori, e l'azione da intraprendere dipende dal superamento o meno di tali soglie.

Quando i dati analizzati rientrano entro i limiti previsti, viene generata una raccomandazione di tipo positivo. Questo tipo di raccomandazione è principalmente incentrata sull'informare il paziente che il suo percorso, relativamente ai parametri analizzati, è corretto e positivo per la sua salute. Al contrario, se i dati superano le soglie stabilite, viene creata una raccomandazione di tipo negativo. Questa raccomandazione fornisce un avviso informativo e guida il paziente verso comportamenti più salutari.

4.1.2 Scelta del Business Rule Engine

Nel processo di selezione del Business Rule Engine per la nostra applicazione, la nostra scelta è stata indirizzata fin da subito verso una soluzione che fosse non solo gratuita ma anche estremamente integrabile con il framework Spring.

Dopo un'analisi dettagliata delle diverse alternative disponibili, abbiamo individuato Drools come la scelta ideale per soddisfare le nostre esigenze. La sua capacità di adattarsi senza sforzi al nostro ambiente Spring e la sua natura open source sono risultati di fondamentale importanza per la scelta. Inoltre, le robuste funzionalità di Drools e la sua flessibilità ci hanno permesso di implementare le regole in modo efficace ed efficiente.

La scelta di Drools si è rivelata vincente, offrendo una solida base per il nostro Business Rule Engine e contribuendo significativamente al successo del nostro progetto.

4.1.3 Drools

Drools permette di definire regole in modo semplice e permette di separare la logica quindi le regole vere e proprie, dalla configurazione. La funzione di base del motore Drools è quella di confrontare i dati in ingresso, o fatti, con le condizioni delle regole e determinare se ed in che modo eseguire le regole.

Drools utilizza i seguenti componenti di base [19]:

- **Regole**: Definite dal programmatore, tutte le regole devono contenere almeno le condizioni che attivano la regola e le azioni che la regola prescrive. Possono avere parametri variabili quindi adattabili per ogni paziente.

- **Fatti**: Dati che entrano o cambiano e che vengono confrontati con le condizioni delle regole per eseguire le regole applicabili.

- **Memoria di produzione**: Dove le regole sono memorizzate.
- **Memoria operativa**: Dove i fatti sono memorizzati.
- **Agenda**: Luogo in cui le regole attivate vengono registrate e ordinate (se necessario) in preparazione all'esecuzione.

Quando un utente o un sistema automatizzato aggiunge o aggiorna le informazioni correlate alle regole in Drools, tali informazioni vengono inserite nella memoria operativa del motore Drools sotto forma di uno o più fatti. Il motore Drools confronta questi fatti con le condizioni delle regole memorizzate nella memoria di produzione per determinare le esecuzioni di regole idonee. (Questo processo di confronto dei fatti con le regole è spesso chiamato "pattern matching".) Quando le condizioni delle regole sono soddisfatte, Drools attiva e registra le regole nell'agenda, dove vengono ordinate in base alla priorità o alla conflittualità in preparazione all'esecuzione.

Il diagramma seguente illustra questi componenti di base di Drools:

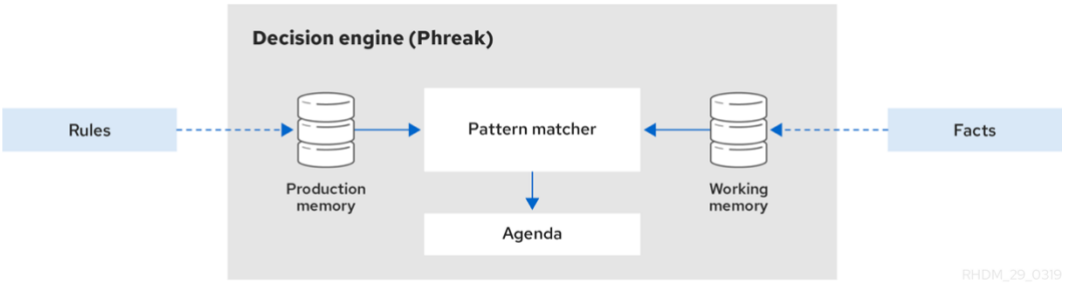

Figura 12 – Diagramma dei Componenti di Drools

Per il funzionamento di Drools è necessario definire una sessione; queste possono essere:

- **Stateless:** Non mantengono uno stato, non ricordano i dati tra le varie sessioni. I fatti vengono inseriti nella Knowledge base session prima di attivare le regole ed eventuali cambiamenti non vengono comunicati.
- **Stateful:** Mantengono uno stato, i fatti vengono inseriti nella Knowledge base session prima dell'attivazione delle regole, bisogna chiamare il

metodo *dispose()* per evitare perdite di memoria ed eventuali cambiamenti dei dati vengono comunicati.

La nostra scelta è ricaduta su sessioni di tipo stateful, poiché se durante una sessione qualche dato cambia, il BRE verrà informato e le regole saranno eseguite sulle informazioni aggiornate.

La struttura di una regola Drools è la seguente [20]:

```
rule "Nome"
        // Attributi
when
        // LHS − Premessa
then
        // RHS − Conclusione
end
```

Gli attributi che possono essere assegnati ad una regola sono molteplici, quelli di principale interesse per i nostri scopi sono 4 [21]:

- **dialect:** Attributo che specifica il linguaggio utilizzato per scrivere le regole e può essere impostato come *MVEL* o *Java*. Nel nostro caso, utilizziamo *MVEL*.

- **agenda-group:** È una stringa che consente di categorizzare le regole in gruppi specifici, specificando a quale gruppo appartiene ciascuna regola. Questo attributo è utile perché offre la possibilità di eseguire solo le regole di un particolare gruppo.

- **activation-group:** Le regole che appartengono allo stesso gruppo di attivazione, identificato dal valore di questo attributo, verranno attivate esclusivamente. La prima regola che si attiva in un gruppo di attivazione annullerà le attivazioni delle altre regole.

- **salience:** Attributo rappresentato da un valore intero che, per

impostazione predefinita, è zero. Questo valore può essere sia positivo che negativo. La *salience* agisce come una forma di priorità, dove le regole con valori di *salience* più alti vengono considerate più importanti. Quando le regole vengono ordinate nella coda di attivazione, quelle con *salience* più elevata sono eseguite prima, poiché hanno una priorità maggiore.

4.2 Implementazione

Nel nostro applicativo, il BRE è configurato in un file di configurazione, qui vengono raccolti tutti i file nei quali sono definite le regole.

```
@Configuration
class DroolsConfig {
    @Bean
    fun kieContainer(): KieContainer {
        val kieFileSystem = kieServices.newKieFileSystem()
        kieFileSystem.write(ResourceFactory.newClassPathResource(BLOOD_ANALYSIS_RULES_DRL))
        kieFileSystem.write(ResourceFactory.newClassPathResource(ACTIVITY_RULES_DRL))
        kieFileSystem.write(ResourceFactory.newClassPathResource(SLEEP_RULES_DRL))
        kieFileSystem.write(ResourceFactory.newClassPathResource(HR_RULES_DRL))
        kieFileSystem.write(ResourceFactory.newClassPathResource(FOOD_RULES_DRL))
        kieFileSystem.write(ResourceFactory.newClassPathResource(WEIGHT_RULES_DRL))
        val kb = kieServices.newKieBuilder(kieFileSystem)
        kb.buildAll()
        val kieModule = kb.kieModule
        return kieServices.newKieContainer(kieModule.releaseId)
    }

    companion object {
        private const val BLOOD_ANALYSIS_RULES_DRL = "rules/BloodAnalysisRules.drl"
        private const val ACTIVITY_RULES_DRL = "rules/ActivityRules.drl"
        private const val SLEEP_RULES_DRL = "rules/SleepRules.drl"
        private const val HR_RULES_DRL = "rules/HrRules.drl"
        private const val FOOD_RULES_DRL = "rules/FoodRules.drl"
        private const val WEIGHT_RULES_DRL = "rules/WeightRules.drl"
        private val kieServices = KieServices.Factory.get()
    }
}
```

Figura 13 – Configurazione di Drools

Ogni file con estensione .drl, contiene le regole associate ad un determinato gruppo, ed ogni gruppo è associato ad una classe Service, incaricata di gestire

il ciclo di vita di una sessione, di inserire i dati e di chiamare l'esecuzione delle regole.

Per implementare il sistema di recommendations di HealthApp, è stato necessario definire le regole e le soglie. Le soglie sono definite in base a valori provenienti da documentazione medica, però questo non lascia molta flessibilità, per questo motivo si è deciso di rendere le soglie personalizzabili per ogni paziente, così che i dottori possano scegliere quelle più indicate per ciascuno.

Nel nostro sistema sono state definiti diversi tipi di regole:

- **A parametro singolo:** Singolo parametro in una singola macroarea.
- **A parametro multiplo**: Regola relativa a più parametri della stessa macroarea messi in relazione tra di loro.
- **A macroarea multipla:** Regola relativa a più parametri di macroaree diverse, messi in relazione tra loro.

Per semplicità, sono stati individuati dei macrogruppi su cui definire le regole e sono i seguenti:

- **Attività**
- **Analisi del sangue**
- **Cibo**
- **Frequenza cardiaca**
- **Sonno**
- **Peso**

4.3 Regole

4.3.1 Attività

In questo gruppo sono contenute le regole riguardanti l'attività fisica dei pazienti, in particolare sui passi e sulle calorie bruciate, tutte le regole sono a parametro singolo.

Per i passi abbiamo quattro regole mutuamente esclusive poiché fanno parte dello stesso activation-group, una sola regola è positiva, le altre sono negative.

La soglia dei passi standard è diecimila [22], ma è personalizzabile per ogni paziente.

Per quanto riguarda le calorie bruciate, abbiamo quattro regole mutuamente esclusive poiché fanno parte dello stesso activation-group, una sola regola è positiva, le altre sono negative.

La soglia delle calorie bruciate standard è 1200, ma è personalizzabile per ogni paziente.

4.3.2 Analisi del sangue

In questo gruppo sono contenute le regole riguardanti le analisi del sangue dei pazienti, sono presenti diversi sottogruppi quali: Eritrociti, Emoglobina, MCV, HT, Leucociti, Piastrine, Glicemia, Urea, Na, K, Creatinina, Colesterolo Totale, Colesterolo HDL, Trigliceridi, PCR, Albumina, Linfociti, RDW, ALP, $\frac{Colesterolo\ Totale}{Colesterolo\ HDL}$, $\frac{Urea}{Creatinina}$ e $\frac{Trigliceridi}{Colesterolo\ HDL}$.

Per tutti i sottogruppi che hanno regole a parametro singolo abbiamo sei regole mutuamente esclusive poiché fanno parte dello stesso activation-group, una sola regola è positiva, le altre sono negative.

I valori soglia per le regole a parametro singolo riguardanti le analisi del sangue ci sono stati forniti dai medici e sono contenuti nella tabella sottostante, questi sono sempre personalizzabili per ogni paziente.

Nome parametro	Unità di misura	Valore convenzionale minimo	Valore convenzionale massimo
Eritrociti	$\frac{10^{12}}{l}$	4,00	5,20
Emoglobina	$\frac{\mathrm{g}}{dl}$	12	14
MCV (Mean Cell Volume)	fl	80	99
HT (Ematocrito)	%	35	47
Leucociti	$\frac{10^{9}}{l}$	4,30	10
Piastrine	$\frac{10^{9}}{l}$	150	400
Glicemia	$\frac{mg}{dl}$	60	99
Urea	$\frac{mg}{dl}$	17	47
Na (Sodio)	$\frac{mmol}{l}$	135	145
K (Potassio)	$\frac{mmol}{l}$	3,4	4,8
Creatinina	$\frac{mg}{dl}$	0,49	1,19
Colesterolo Totale	$\frac{mg}{dl}$	/	200
Colesterolo HDL	$\frac{mg}{dl}$	50	/
Trigliceridi	$\frac{mg}{dl}$	/	150
PCR (Proteina C reattiva)	$\frac{mg}{l}$	/	5
Albumina	$\frac{m}{l}$	/	/
Linfociti	%	/	/
RDW	%	/	/
ALP	$\frac{u}{l}$	/	/

Tabella 1 – Valori di Riferimento Analisi del Sangue

Per i sottogruppi con regole a parametro multiplo abbiamo tre regole a parametro multiplo mutuamente esclusive, una sola è positiva, le altre sono negative.

Il rapporto $\frac{Colesterolo\ Totale}{Colesterolo\ HDL}$, indica l'indice di rischio cardiovascolare, abbiamo che se questo è minore di 5 è basso (Positivo), se è compreso tra 5 e 10 uguaglianza inclusa è moderato, se è maggiore di 10 è elevato [23].

Se il rapporto $\frac{Urea}{Creatinina}$ è minore di 40 è basso, se è compreso tra 40 e 100 uguaglianza inclusa è normale (Positivo) e se è maggiore di 100 è elevato [24].

Il rapporto $\frac{Trigliceridi}{Colesterolo\ HDL}$ è uno dei più potenti indici predittivi di malattia cardiovascolare. Se è minore o uguale a 2 è ideale (Positivo), se è maggiore di 2 e minore o uguale a 6 è elevato, se è maggiore di 6 è troppo elevato [25].

Le soglie di queste regole a parametro multiplo sono derivate dalla letteratura medica e non sono personalizzabili, sono uguali per tutti i pazienti.

4.3.3 Cibo

Il gruppo relativo al cibo si divide a sua volta in diversi sottogruppi quali: calorie, carboidrati, proteine, grassi, grassi saturi, grassi polinsaturi, grassi monoinsaturi, grassi trans, colesterolo, sodio, potassio, fibre, zuccheri, zuccheri aggiunti, vitamina D, vitamina A, vitamina C, calcio, ferro, $\frac{Sodio}{Potassio}$, proporzione tra grassi saturi, monoinsaturi e polinsaturi, proporzione tra carboidrati, proteine e grassi, bilancio tra calorie immesse e calorie bruciate.

Possiamo notare che oltre ai sottogruppi che contengono regole a parametro singolo e a parametri multipli, è presente anche un sottogruppo di regole a macroarea multipla, ovvero bilancio tra calorie immesse e calorie bruciate.

I dati sul cibo verranno aggregati con granularità minima giornaliera; quindi, saranno considerati valori nutrizionali su base giornaliera, settimanale e mensile.

Per tutti i sottogruppi che hanno regole a parametro singolo abbiamo quattro regole mutuamente esclusive poiché fanno parte dello stesso activation-group, una sola regola è positiva, le altre sono negative.

Le soglie relative a questi parametri sono definite nella tabella seguente:

Nome parametro	Unità di misura	Valore soglia
Calorie [26]	kcal	3000
Carboidrati [27]	g	300
Proteine [28]	g	150
Grassi [29]	g	75
Grassi saturi [30]	g	18
Grassi polinsaturi [30]	g	15
Grassi monoinsaturi [30]	g	40
Grassi trans [31]	g	2
Colesterolo [32]	mg	300
Sodio [33]	mg	2000
Potassio [34]	mg	3510
Fibre [35]	g	35
Zuccheri [36]	g	70
Zuccheri aggiunti [37]	g	50
Vitamina D [38]	µg	25
Vitamina A [39]	µg	950
Vitamina C [40]	mg	90
Calcio [41]	mg	1000
Ferro [42]	mg	14

Tabella 2 – Valori Nutrizionali di Riferimento

Tali valori derivano da ricerche fatte sul web, poiché le nostre conoscenze in materia non sono tali da poterli definire, per questo motivo possono essere personalizzati dai medici per ogni paziente in base alle necessità.

Per il sottogruppo $\frac{Sodio}{Potassio}$, abbiamo due regole a parametro multiplo mutuamente esclusive, una positiva l'altra negativa, in particolare se il rapporto è minore di 1 è ok (Positivo), se è maggiore o uguale a 1 è alto [43].

Per la proporzione tra grassi saturi, monoinsaturi e polinsaturi abbiamo otto regole a parametro multiplo mutuamente esclusive, una positiva e le altre negative. In particolare, abbiamo che se la proporzione tra i grassi è 25% di grassi saturi, 55% di grassi monoinsaturi, e 20% di grassi polinsaturi è ok (Positivo) [30], in tutti gli altri casi possiamo avere alta quantità di uno o più tipi di grassi se superano queste percentuali.

Per la proporzione tra carboidrati, proteine e grassi abbiamo otto regole a parametro multiplo mutuamente esclusive, una positiva e le altre negative. In particolare, abbiamo che se la proporzione è 50% carboidrati, 20% proteine e 30% grassi è ok (Positivo) [44], in tutti gli altri casi possiamo avere alta quantità di uno o più elementi se superano queste percentuali.

Per il bilancio tra calorie immesse e calorie bruciate abbiamo tre regole a macroarea multipla mutuamente esclusive, tutte positive poiché in realtà hanno scopo puramente informativo, poiché il bilancio calorico dipende dalla dieta di ciascuno [45]. In particolare, abbiamo che le calorie immesse possono essere maggiori, minori o uguali a quelle bruciate.

4.3.4 Frequenza cardiaca

Il gruppo frequenza cardiaca contiene le regole relative alla frequenza cardiaca a riposo dei pazienti; in particolare contiene sei regole a parametro singolo mutuamente esclusive poiché fanno parte dello stesso activation-group, una sola regola è positiva, le altre sono negative.

Le soglia minima è 55 bpm, mentre quella massima è 100 bpm [46], questa soglia è personalizzabile dai medici per ciascun paziente.

4.3.5 Sonno

Il gruppo sonno contiene le regole relative alla durata del sonno dei pazienti; in particolare contiene quattro regole a parametro singolo mutuamente esclusive poiché fanno parte dello stesso activation-group, una sola regola è positiva, le altre sono negative.

Le soglia per la durata del sonno è pari a otto ore per gli adulti [47], anche questa soglia è modificabile dai medici per ogni paziente.

4.3.6 Peso

Il gruppo peso contiene due sottogruppi quali peso e BMI dei pazienti; in particolare per ogni sottogruppo contiene quattro regole a parametro singolo mutuamente esclusive poiché fanno parte dello stesso activation-group, una sola regola è positiva, le altre sono negative.

Le soglia di peso è pari a 100 kg, non ha una motivazione scientifica, questa è una soglia che andrebbe definita in ogni caso dal medico in base al singolo paziente.

La soglia di BMI è 30, valore minimo per l'obesità [48], anche questa soglia può essere personalizzata.

Per una descrizione completa di tutte le regole del sistema di recommendations di HealthApp si rimanda all'Allegato 1.

4.4 API

Le API relative alle soglie del sistema di recommendations sono in totale 24, queste ci permettono di aggiungere, recuperare, modificare ed eliminare le soglie relative a tutti i parametri discussi nei paragrafi precedenti, questo permette di avere una flessibilità assoluta, poiché in ogni momento il medico può modificare le soglie per i pazienti a suo piacimento, permettendo di definire le soglie in maniera personalizzata in base alla condizione di ogni paziente.

Le API relative al recupero delle recommendations sono in totale 28, queste ci permettono di recuperare le recommendations per ogni paziente in due formati, JSON o PDF, il primo pensato per un futuro supporto alla visualizzazione direttamente sul client web, il secondo per essere scaricabile e consultabile sia dai pazienti sia dai medici.

Per una descrizione più dettagliata e aggiornata delle API si rimanda alla dashboard di Swagger, consultabile all'indirizzo:

https://dev.verona-experiment.org/swagger-ui/index.html

4.5 Generazione automatica dei report

Nel backend di HealthApp, è stata implementata routine periodica che opera con cadenza settimanale, generando report dettagliati per ciascun gruppo di pazienti e archiviandoli in formato PDF. La chiave di questa funzionalità è fornire un modo efficiente e strutturato per accedere a dati significativi attraverso le API relative.

Al fine di mantenere una gestione ottimale della base dati, abbiamo deciso di conservare questi report per un periodo di un anno. Inoltre, per garantire l'efficienza e la pulizia del sistema, abbiamo implementato una specifica routine di eliminazione che si occupa di rimuovere periodicamente i report più datati, evitando così un appesantimento eccessivo.

Di seguito possiamo vedere un esempio di report PDF prodotto dal nostro recommender system che offre una panoramica dei risultati riguardanti l'attività fisica del paziente nel periodo 01-11-2023 / 08-11-2023, offrendo informazioni utili sia per i pazienti che per i medici coinvolti.

Activity Analysis Result - 2023-11-08

Activity Analysis Result

Steps Analysis Result

Date	Result	Result Week	Result Month	Positive Result
2023-11-01	Steps low	Steps low	Steps low	false
2023-11-02	Steps low	Steps low	Steps low	false
2023-11-03	Steps low	Steps low	Steps low	false
2023-11-04	Steps low	Steps low	Steps low	false
2023-11-05	Steps are ok	Steps low	Steps low	true
2023-11-06	Steps low	Steps low	Steps low	false
2023-11-07	Steps low	Steps low	Steps low	false

Calories Analysis Result

Date	Result	Result Week	Result Month	Positive Result
2023-11-01	Calories out are ok	Calories out are ok	Calories out are ok	true
2023-11-02	Calories out are ok	Calories out are ok	Calories out are ok	true
2023-11-03	Calories out are ok	Calories out are ok	Calories out are ok	true
2023-11-04	Calories out are ok	Calories out are ok	Calories out are ok	true
2023-11-05	Calories out are ok	Calories out are ok	Calories out are ok	true
2023-11-06	Calories out are ok	Calories out are ok	Calories out are ok	true
2023-11-07	Calories out low	Calories out are ok	Calories out are ok	false

Figura 14 – Report Settimanale del Sistema di Recommendations

5. Integrazione dei Servizi Esterni

Nei capitoli precedenti sono state presentate alcune funzionalità dell'applicazione e si è parlato di come quest'ultima si serva di alcuni servizi esterni, in particolare *Fitbit* per raccogliere i dati su attività fisica, sonno, e frequenza cardiaca dei pazienti e *FatSecret*, che offre un dataset di ricette e alimenti con i valori nutrizionali associati. Nei prossimi paragrafi vedremo nel dettaglio come questi servizi esterni sono integrati in HealthApp.

5.1 Fitbit

L'integrazione della nostra app con Fitbit è un elemento significativo per arricchire l'esperienza degli utenti, consentendo ai medici di monitorare in modo semplice, completo e dettagliato lo stato di salute e lo stile di vita dei pazienti, e ai pazienti stessi di tenere traccia dei loro progressi e avere un quadro completo con tutti i dati a disposizione. Attraverso questa integrazione, siamo in grado di recuperare i dati relativi al sonno, alla frequenza cardiaca e all'attività fisica dell'utente.

Gli smartwatch Fitbit permettono di raccogliere questi dati, infatti i pazienti dovranno indossare questi dispositivi per tutta la durata dell'esperimento. I dati raccolti vengono inviati e memorizzati nei server Fitbit, e tramite alcune API che Fitbit mette a disposizione questi vengono recuperati dal backend di HealthApp e memorizzati nell'HealthApp database.

Per poter recuperare i dati e poter utilizzare le API fornite da Fitbit, è necessario che i due server siano in qualche modo connessi; per fare ciò, è necessario che HealthApp sia registrata sulla console di sviluppatori Fitbit [49].

Per registrare l'applicazione è necessario compilare il seguente form, dove viene richiesta la tipologia di applicazione, il nome e qualche altra informazione riguardo all'applicazione che si vuole sviluppare.

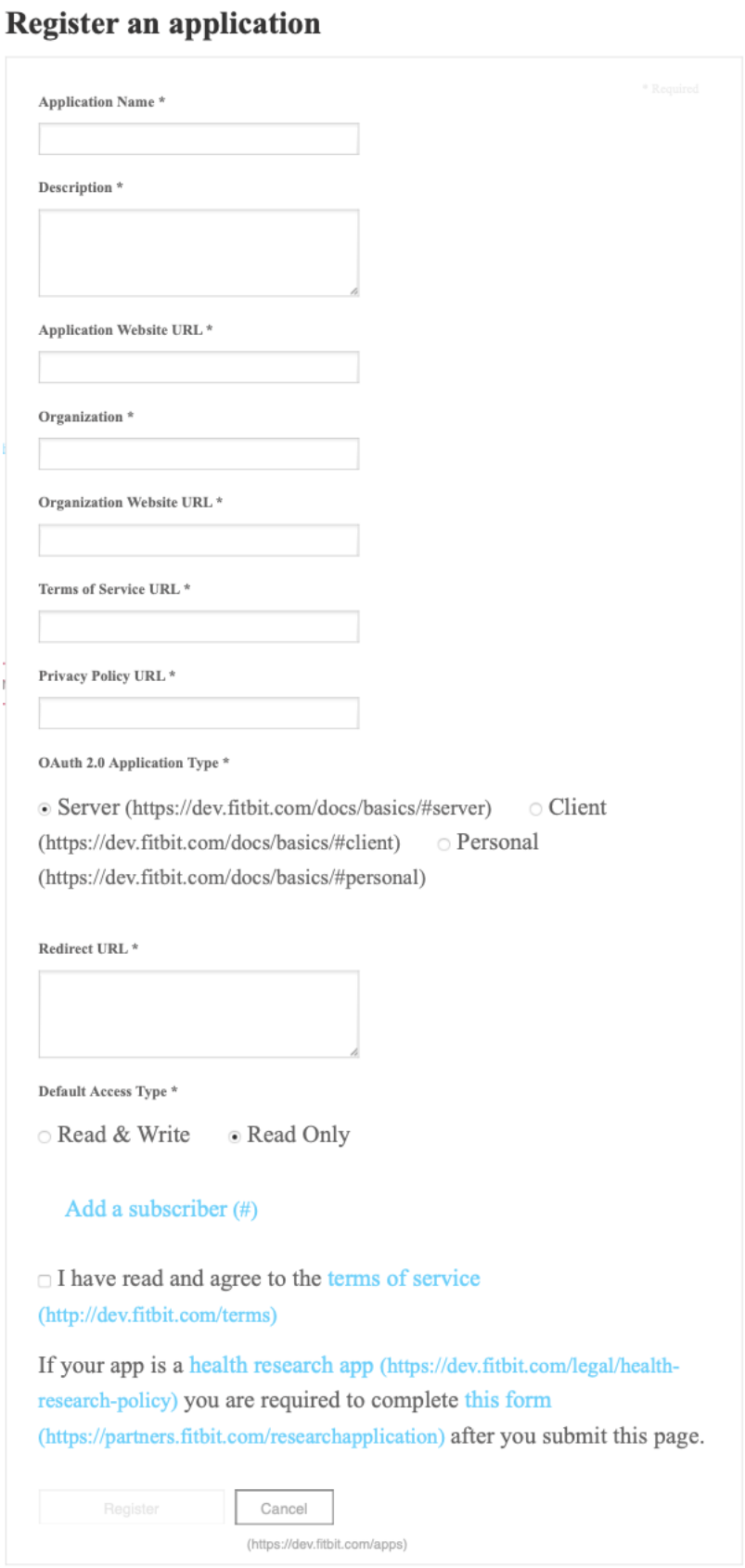

Figura 15 – Form di Registrazione App Fitbit

È possibile scegliere tra varie tipologie di Applicazioni [50]:

- **Server (Consigliata):** Sono generalmente multilivello e presentano una componente server nell'architettura. Forniscono l'autenticazione lato server in cui l'applicazione e il server comunicano tramite un servizio web.

- **Client:** Sono solitamente a livello singolo e non contengono una componente server nell'architettura. Queste applicazioni utilizzano un'autenticazione lato client.

- **Personali:** Possono avere un'architettura client o server. Queste applicazioni sono limitate all'accesso ai dati nell'account Fitbit dello sviluppatore.

Per gli scopi di HealthApp è stata scelta un'applicazione di tipo server, con accesso ai dati in sola lettura, poiché ci serve solo accedere a questi dati e salvarli nel nostro database; quindi, le manipolazioni necessarie avvengono nel nostro backend.

Una volta registrata l'applicazione, i seguenti parametri saranno a disposizione dello sviluppatore, in modo tale da poter integrare realmente Fitbit all'interno dell'applicazione:

- **OAuth 2.0 Client ID:** È un identificatore univoco assegnato all'applicazione durante la registrazione sulla piattaforma Fitbit Developer. Viene utilizzato durante il processo di autenticazione per identificare l'app quando interagisce con le API di Fitbit.

- **Client Secret:** È una chiave segreta associata al Client ID. Viene utilizzato per autenticare l'applicazione nei confronti del server di autorizzazione Fitbit durante il processo di OAuth 2.0.

- **Redirect URL:** È fornito dallo sviluppatore e indica il percorso al quale l'utente verrà rimandato dopo aver autorizzato l'applicazione.

- **OAuth 2.0: Authorization URI:** È utilizzato per avviare il processo di autorizzazione OAuth 2.0. Durante l'autenticazione, l'utente sarà reindirizzato a questo URI per concedere l'accesso all'applicazione Fitbit.

- **OAuth 2.0: Access/Refresh Token:** Dopo aver dato l'autorizzazione, l'applicazione ottiene un Access Token che viene utilizzato per autenticare le richieste API. Questo token ha una durata limitata, infatti viene generato un Refresh Token che consente all'applicazione di ottenere un nuovo Access Token senza richiedere nuovamente l'autorizzazione dell'utente.

Il processo di autorizzazione OAuth 2.0 avviene come segue [51]:

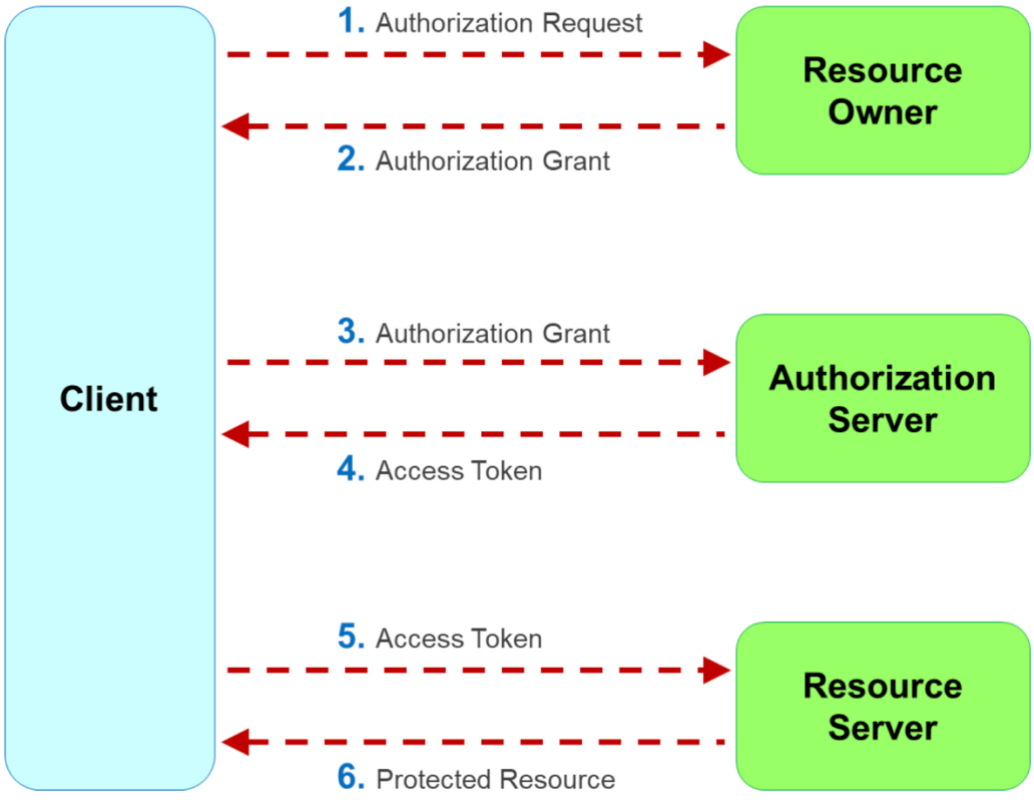

Figura 16 – Processo di Autorizzazione OAuth 2.0

1) Il Client invia una richiesta di autorizzazione all'utente, questa può avvenire direttamente o tramite l'Authorization Server.

2) Al Client viene concessa l'autorizzazione.

3) Il Client richiede un Access Token autenticandosi sull'Authorization Server.

4) L'access Token viene inviato al Client.

5) Il Client richiede una risorsa protetta presentando l'Access Token.

6) Il Resource Server verifica l'Access Token e se è valido invia la risorsa protetta.

Nel caso di HealthApp e Fitbit il processo è pressoché identico, entrando nel dettaglio abbiamo che:

1) Il paziente con l'account appena creato dal medico effettua l'accesso su HealthApp.

2) Al primo accesso, l'utente avrà una sola scelta, quella di cliccare un bottone che effettuerà la chiamata al OAuth 2.0: Authorization URI tramite un URL costruito dal backend.

3) L'Authorization Server di Fitbit riconoscerà l'applicazione e l'utente si troverà sulla schermata di login di Fitbit.

4) Effettuato l'accesso verranno richieste le autorizzazioni necessarie all'utente.

5) Una volta ottenuto il consenso, il *Fitbit Authorization Server* chiamerà l'API di HealthApp contenuta nel *Redirect URL* passando un codice generato sul momento.

6) Il codice ricevuto verrà utilizzato da HealthApp per fare richiesta dell'Access Token effettuando una chiamata all'OAuth 2.0: Access/Refresh Token Request URI.

7) Il Fitbit Authorization Server genererà due token, l'Access Token e il

Refresh Token, il primo ha una durata di otto ore, il secondo viene utilizzato per rinnovare il primo senza richiedere una nuova autorizzazione all'utente.

8) HealthApp riceverà i due token e li salverà nell'HealthApp database associandoli all'utente corretto, avendo così la possibilità di accedere alle risorse protette di Fitbit.

Giornalmente l'Access Token viene rinnovato inviando una richiesta all'OAuth 2.0: Access/Refresh Token inviando il refresh token. I vecchi token verranno quindi sostituiti da quelli ricevuti in risposta alla richiesta di refresh.

Una volta ottenuto l'accesso token, abbiamo la possibilità di accedere ai dati Fitbit degli utenti, così da poterli salvare nel nostro database, per fare questo è necessario utilizzare le api messe a disposizione da Fitbit. Di seguito vedremo le tre API che vengono usate da HealthApp per recuperare i dati di interesse [52]:

- ***GET /1.2/user/[user-id]/sleep/date/[start-date]/[end-date].json*** [53]*:* Ritorna un array contenente i dati riguardanti il sonno dell'utente nell'intervallo di date selezionato, in particolare ritorna la durata del sonno, e la durata delle varie fasi del sonno (Deep, Light, REM, Awake).

- ***GET /1/user/[user-id]/activities/hearth/date/[start-dat]/[end-date].json*** [54]*:* Ritorna un array contenente i dati riguardanti la frequenza cardiaca dell'utente nell'intervallo di date selezionato, in particolare ritorna la media giornaliera dei battiti a riposo e un oggetto contenente i dati relativi alle varie zone cardiache.

- ***GET /1/user/[user-id]/activities/steps/date/[start-dat]/[end-date].json*** [55]: Ritorna un array contenente i dati riguardanti l'attività fisica dell'utente nell'intervallo di date selezionato, in particolare ritorna il numero di passi giornaliero e le calorie bruciate giornalmente.

HealthApp non è un’applicazione che ha bisogno di dati in tempo reale, poiché il suo scopo principale è quello di monitorare il benessere e i progressi dei pazienti nel tempo, per questo motivo abbiamo deciso che l’aggiornamento dei dati avviene tramite alcuni task programmati due volte al giorno, il che è sufficiente per mantenere il livello di dettaglio di cui i medici hanno bisogno per tenere sotto controllo i pazienti.

I dati sul sonno forniscono un'analisi approfondita dei cicli di riposo, consentendo agli utenti di comprendere la qualità del loro sonno e apportare eventuali miglioramenti. La frequenza cardiaca, misurata con precisione dai dispositivi Fitbit, offre un monitoraggio continuo che può essere utilizzato per valutare il livello di attività fisica e il grado di sforzo durante le sessioni di allenamento, dando la possibilità di recuperare informazioni relative sia alla media dei battiti a riposo, sia alle varie zone di frequenza cardiaca. Infine, l'integrazione con le informazioni sull'attività fisica fornisce un quadro completo del livello di movimento quotidiano, aiutando gli utenti a mantenere uno stile di vita attivo e bilanciato.

La scelta di memorizzare i dati di Fitbit nel database di HealthApp è dettata da diversi fattori:

- **Accesso Continuato ai Dati:** Memorizzando i dati internamente, garantiamo che HealthApp abbia accesso completo ai dati degli utenti anche in situazioni in cui il server Fitbit potrebbe essere momentaneamente non raggiungibile. Questo approccio riduce la dipendenza da eventuali interruzioni di servizio da parte di Fitbit.

- **Efficienza nel Recupero Dati:** Fitbit limita la lettura dei dati a un intervallo di cento giorni. Memorizzando internamente i dati, possiamo recuperare le informazioni più recenti con una singola chiamata al server Fitbit anziché effettuare più chiamate per ottenere dati accumulati nel tempo. Ciò migliora l'efficienza e riduce il carico sulle risorse del server Fitbit.

- **Flessibilità per Futuri Wearable/Tracker:** L'approccio di memorizzazione interna consente una progettazione più flessibile per il futuro. Se si desidera estendere il supporto ad altri wearable o tracker, mantenere una base dati centralizzata semplifica l'implementazione di adattatori per la memorizzazione uniforme dei dati provenienti da diverse fonti.

In sintesi, questa strategia di memorizzazione interna mira a garantire un accesso affidabile e continuo ai dati dei pazienti, mantenendo al contempo la flessibilità per futuri sviluppi e integrando altre fonti di dati.

Tutti i dati raccolti tramite Fitbit e memorizzati nell'HealthApp database sono poi accessibili ai vari client tramite le nostre API, tutte di tipo GET, in particolare abbiamo un'api per recuperare il numero di passi dell'utente, una per recuperare le calorie bruciate, una per recuperare i dati relativi al sonno e una per recuperare i dati relativi alla frequenza cardiaca. Tutte le api ricevono un intervallo di date e restituiranno i dati relativi all'intervallo ricevuto. Per una descrizione più dettagliata e aggiornata delle API si rimanda alla dashboard di Swagger, consultabile all'indirizzo:

https://dev.verona-experiment.org/swagger-ui/index.html

5.2 FatSecret

FatSecret viene concepita come un'applicazione dedicata a monitorare l'apporto calorico e i valori nutrizionali associati al consumo alimentare. Il suo vasto database alimentare, composto da circa un milione e mezzo di record [56], è organizzato in sezioni geografiche, e in ogni sezione possono essere presenti cibi o ricette diversi. FatSecret fornisce le API per accedere alle informazioni disponibili nel suo database e nel caso di progetti no-profit come HealthApp è possibile usufruire di una licenza gratuita per l'accesso illimitato alle informazioni della sezione USA.

I valori nutrizionali che FatSecret mette a disposizione e che noi memorizziamo nel nostro database sono:

Nome parametro	Unità di misura
Calorie	kcal
Carboidrati	g
Proteine	g
Grassi	g
Grassi saturi	g
Grassi polinsaturi	g
Grassi monoinsaturi	g
Grassi trans	g
Colesterolo	mg
Sodio	mg
Potassio	mg
Fibre	g
Zuccheri	g
Zuccheri aggiunti	g
Vitamina D	µg
Vitamina A	µg
Vitamina C	mg
Calcio	mg
Ferro	mg

Tabella 3 – Valori Nutrizionali FatSecret

Al netto di tutte le considerazioni fatte, ovvero il vasto numero di alimenti presenti nella base dati e la possibilità di usufruire di una licenza gratuita per gli scopi della nostra applicazione, si è ritenuto che FatSecret fosse una scelta adeguata per i nostri obiettivi.

L'integrazione di FatSecret con HealthApp è simile a quello di Fitbit, infatti, per utilizzare le API di FatSecret, è necessario registrare la propria applicazione nella console per sviluppatori.

A differenza di Fitbit, l'integrazione con FatSecret si basa sul protocollo OAuth 1.0 [57]. Tale scelta è motivata dalla considerazione che, per le esigenze specifiche di questo tipo di servizio, non è richiesto un livello elevato di confidenzialità. Questa decisione è stata presa con l'obiettivo di semplificare il processo di sviluppo. È importante notare che, sebbene l'implementazione con OAuth 2.0 fosse tecnicamente fattibile, tale opzione avrebbe introdotto una complessità aggiuntiva e avrebbe richiesto un periodo di sviluppo più lungo. Pertanto, la preferenza per OAuth 1.0 è stata guidata dalla volontà di garantire un'integrazione agevole ed efficiente.

Una volta accettata la richiesta e attivata la licenza gratuita, anche qui si ottengono le credenziali per finalizzare l'integrazione, in particolare viene ricevuta una coppia di chiavi [57]:

- **Consumer Key:** una chiave univoca che identifica il consumer (nel nostro caso, HealthApp) che vuole accedere alle API.
- **Shared Secret:** una chiave segreta che è condivisa tra il consumer e il server di autenticazione.

Tutte le richieste dovranno essere firmate utilizzando queste chiavi, e in particolare, per ogni richiesta dovranno essere presenti i seguenti parametri [58]:

- **oauth_consumer_key:** L'API key che ci viene fornita quando ci registriamo come sviluppatori.

- **oauth_signature_method:** Metodo usato per generare la firma, HMAC-SHA1 è l'unico supportato.
- **oauth_timestamp:** Data e ora della richiesta.
- **oauth_nonce:** Una stringa randomica generata per la richiesta.
- **oauth_version:** Nel caso di OAuth 1.0 sarà sempre 1.0.

Per creare una richiesta si seguono i seguenti passaggi:

1) Si crea una *Signature Base String* concatenando il metodo della richiesta (GET o POST), il *Request URL* e i parametri precedentemente elencati come segue:

   <HTTP Method>&<Request URL>&<Normalized Parameters>

   I parametri sono scritti nel formato "nome=valore" e ordinati utilizzando l'ordinamento lessicografico del valore dei byte, prima per nome e poi per valore.

   Tutti i parametri della richiesta devono essere codificati usando il meccanismo di *percent-encoding* %xx dello standard *RFC 3986* [59] e concatenati dalla "&".

2) Si calcola il parametro *oauth_signature* utilizzando l'algoritmo di firma HMAC-SHA1 per firmare la richiesta dove *text* è la *Signature Base String* calcolata prima e *key* sono i valori Consumer Secret e Access Secret concatenati e separati da "&" (Nel caso di HealthApp Access Secret è una stringa vuota). Il digest calcolato viene prima codificato in base64 *(RFC 2045)* [60], quindi in percent-encoding *(RFC 3986)* [59], ottenendo *oauth_signature.*

3) Si invia la richiesta inviando tutti i parametri utilizzati per generare la *Signature Base String* tramite il metodo HTTP specificato nella *Signature Base String*, includendo anche *oauth_signature*, se questa

corrisponde con quella generata dal server FatSecret allora il richiedente riceverà la risorsa protetta.

Tutte le richieste indirizzate a FatSecret devono essere dirette al baseURL *https://platform.fatsecret.com/rest/server.api*, in particolare le API che utilizziamo sono due:

- **baseURL?method=food.search:** Riceve la stringa da ricercare e tutti i parametri precedentemente elencati e ritorna un JSON contenente un array di cibi con i valori nutrizionali associati.
- **baseURL?method=food.get.v3:** Riceve l'id di un cibo e tutti i parametri precedentemente elencati e ritorna un JSON contenente un cibo con valori nutrizionali associati.

In questo caso si è deciso di non salvare nell'HealthApp Database l'intero database di FatSecret poiché sarebbe stato molto costoso in termini di spazio di archiviazione, ma soltanto gli id degli alimenti consumati dagli utenti, la quantità consumata, i valori nutrizionali calcolati in base al quantitativo consumato, oltre che la data, la categoria di cibo e il tipo di pasto. Le tipologie di pasto e le categorie di cibo disponibili sono indicati nelle seguenti tabelle.

Tipologia di Pasto	Valore
Colazione	0
Spuntino della Mattina	1
Pranzo	2
Spuntino del Pomeriggio	3
Cena	4
Spuntino Serale o di Mezzanotte	5
Altro	6

Tabella 4 – Tipologie di Pasto

Nome Categoria
Altro
Bevande
Carni Bianche
Carni Rosse
Dolci, Caramelle e Dessert
Fast Food
Formaggi, Latte e Latticini
Frutta
Insalate
Legumi
Noci, Semi e Frutta Secca
Oli e Grassi
Pane, Farine e Cereali
Pasta e Riso
Pesce e Frutti di Mare
Ricette preparate
Salse, Spezie e Creme Spalmabili
Snack
Uova
Verdure e Ortaggi

Tabella 5 – Categorie di Alimenti

L'integrazione di FatSecret con la nostra applicazione permette ai pazienti di cercare gli alimenti e tenere traccia degli alimenti consumati e dei valori nutrizionali. Inoltre, FatSecret coesiste con un dataset proprietario che permette ai pazienti di creare anche delle ricette personalizzate, per offrire un grado di flessibilità maggiore possibile.

Come specificato nei capitoli precedenti, le funzionalità legate all'alimentazione al momento non vengono attivamente utilizzate. Tuttavia, è stato adottato un approccio proattivo, mantenendo e aggiornando tali funzionalità. Tale decisione è stata presa con l'intento di garantire che, qualora dovesse sorgere l'opportunità di riattivare o sfruttare le funzionalità legate all'alimentazione in contesti futuri o esperimenti successivi, il processo di integrazione sarebbe semplificato e richiederebbe solo modifiche specifiche all'applicazione mobile, preservando l'integrità e la funzionalità del backend operativo.

6. Spunti di Miglioramento e Conclusioni

6.1 Spunti di Miglioramento

Il backend di HealthApp è stato sviluppato in modo da lavorare in maniera efficace ed efficiente per quanto concerne le funzionalità che supporta.

Attualmente è in corso l'esperimento nel quale sono coinvolti tre medici e cento pazienti, che nello specifico sono pensionati CISL. Per esperimenti futuri si potrebbe pensare a diversi miglioramenti ed espansioni dell'applicazione attualmente esistente, ad esempio:

- Implementare il supporto ad altri smartwatch diversi dal Fitbit, non rappresenterebbe una sfida troppo impegnativa poiché, con la strategia di mantenere i dati nell'HealthApp database si avrebbero sempre dei dati omogenei.

- Integrare altre tecnologie, come l'intelligenza artificiale o l'apprendimento automatico, per migliorare ulteriormente le capacità di analisi dei dati e rendere le raccomandazioni più personalizzate ed efficaci nel tempo.

- Integrazione di nuovi sensori che raccolgono altri tipi di dati, come ad esempio misuratori di pressione e saturimetri, questo renderebbe le informazioni a disposizione dei medici ancora più complete, permettendo di avere una visione più ampia sullo stato di salute e sullo stile di vita dei pazienti.

Allo stato attuale il backend è funzionante in tutte le sue parti e soddisfa tutte le richieste che sono state fatte dai medici. Se in futuro qualche altra richiesta o necessità dovesse essere presentata sarà sicuramente implementata dai colleghi che lavoreranno in futuro sul progetto HealthApp.

6.2 Conclusioni

In conclusione, il percorso affrontato nel corso di questa tesi di laurea ha rappresentato un'avventura avvincente nel mondo dell'integrazione tra tecnologia informatica e medicina. Fin dall'inizio, ho affrontato questa opportunità con passione e determinazione, consapevole dell'importanza cruciale della medicina e dell'impatto significativo che la tecnologia può avere nel migliorare la nostra salute. Lavorare su HealthApp è stato molto più di un progetto accademico; è stato un viaggio emozionante e arricchente che ha stimolato la mia curiosità e alimentato la mia passione per questa fusione tra informatica e medicina. La salute è un bene prezioso, e contribuire a un progetto che mira a migliorare la gestione e la comprensione dei dati relativi alla salute è stato un privilegio.

Questo percorso mi ha insegnato che il lavoro di squadra è fondamentale per il successo di progetti complessi come HealthApp. La collaborazione con gli altri tesisti e i medici dell'Azienda Ospedaliera Universitaria Integrata di Verona è stata molto arricchente, ho imparato che la combinazione di competenze eterogenee è essenziale per affrontare sfide multidisciplinari e per portare a termine progetti ambiziosi come questo.

La mia connessione personale con il campo della medicina ha reso questo progetto ancora più significativo per me. La speranza è che, attraverso HealthApp, si possa aprire una nuova era nella gestione della salute, contribuendo a migliorare la qualità della vita dei pazienti e facilitando il lavoro degli operatori sanitari.

Guardando al futuro, mi auguro sinceramente che l'esperimento in corso ottenga risultati positivi e che HealthApp possa avere un seguito significativo. Questo progetto ha un potenziale notevole per influenzare positivamente la pratica medica e la vita dei pazienti. È con grande ottimismo che mi immagino il contributo continuo di HealthApp al campo della salute e alle esperienze dei pazienti.

Per concludere, questo percorso di ricerca e sviluppo è stato un vero trampolino di crescita, sia a livello professionale che personale. Ha consolidato la mia convinzione che la tecnologia, in ogni suo aspetto, possa essere una forza incredibilmente positiva e trasformativa in qualsiasi settore, mi auguro che l'informatica possa continuare a svolgere un ruolo sempre più centrale e costruttivo in diversi ambiti, poiché la tecnologia ha il potenziale di apportare cambiamenti significativi e migliorare la qualità della vita per tutti.

7. Bibliografia e Sitografia

7.1 Bibliografia

(1) Fontana L., *The path to longevity: How to Reach 100 with the Health and Stamina of a 40-Year-Old*, s.l., Hardie Grant Books, 6 aprile 2020, I Edizione, 318 pagine, Inglese.

(2) Wijegunaratne I., *Distributed applications engineering: Building new applications and managing legacy applications with distributed technologies*, s.l., Springer., 16 ottobre 1998, I Edizione, 270 pagine, Inglese.

(3) Campbell A., *Agile: Essentials of Team and Project Management. Manifesto for Agile Software Development*., s.l., Independently Published, 2 luglio 2020, 72 pagine, Inglese.

(4) Ponelat J., Rosenstock L., *Designing APIs with Swagger and OpenAPI*, s.l., Manning Publications Co. LLC., 5 luglio 2022, 394 pagine, Inglese.

(5) Momjian, B., *PostgreSQL: Introduction and Concepts*., s.l., Addison-Wesley, 15 dicembre 2000, 462 pagine, Inglese.

(6) Srivastava R., Mallick P. K., Rautaray, S. S., Pandey M., *Computational Intelligence for Machine Learning and Healthcare Informatics*, s.l., de Gruyter GmbH, Walter, 22 giugno 2020, I Edizione, 240 pagine, Inglese.

7.2 Sitografia

[1] Vlasic Scott, Zirous, Three-Tier Architecture Approach for Custom Applications,
https://www.zirous.com/2022/11/15/three-tier-architecture-approach-for-custom-applications-2/,
Consultato il 26/10/2023.

[2] IBM, Cos’è l’architettura three tier?,
https://www.ibm.com/it-it/topics/three-tier-architecture,
Consultato il 26/10/2023.

[3] Medium, *Stereotype Annotations in Spring: The Building Blocks of Efficient Application Development,* https://medium.com/javarevisited/stereotype-annotations-in-spring-the-building-blocks-of-efficient-application-development-710eeb9dfb0a#:~:text=Stereotype%20annotations%20are%20a%20powerful,a%20particular%20role%20or%20purpose., Consultato il 30/10/2023.

[4] Peiretti Gustavo, What are Components and Stereotypes in Spring Boot, https://gustavopeiretti.com/spring-boot-components-stereotypes/, Consultato il 30/10/2023.

[5] GeeksforGeeks, Difference Between @Component, @Repository, @Service, and @Controller Annotations in Spring, https://www.geeksforgeeks.org/difference-between-component-repository-service-and-controller-annotations-in-spring/, Consultato il 02/11/2023.

[6] Medium, Difference between @Controller and @RestController in Spring Boot and Spring MVC?, https://medium.com/javarevisited/difference-between-controller-and-restcontroller-in-spring-boot-and-spring-mvc-216578ad445f, Consultato il 04/11/2023.

[7] Baeldung, Introduction to Spring Data JPA, https://www.baeldung.com/the-persistence-layer-with-spring-data-jpa, Consultato il 04/11/2023.

[8] Riccardo Borgacci, MyPersonalTrainer, Come calcolare il proprio BMI?, https://www.my-personaltrainer.it/calcolo-bmi.html#272522, Consultato il 14/11/2023.

[9] Giulia Bertelli, MyPersonalTrainer, Albumina: a cosa serve l'esame? Significato Albumina alta e bassa, https://www.my-personaltrainer.it/fisiologia/albumina.html#233814, Consultato il 05/11/2023.

[10] Dr. Roberto Gindro, MyPersonalTrainer Valori Normali, Linfociti alti, bassi e valori normali, https://www.valorinormali.com/sangue/linfociti/, Consultato il 05/11/2023.

[11] Giulia Bertelli, MyPersonalTrainer, RDW: Cos'è? Perché si Misura? Cosa Significa?, https://www.my-personaltrainer.it/salute/rdw.html#127356, Consultato il 05/11/2023.

[12] Giulia Bertelli, MyPersonalTrainer, Fotofasi Alcalina (ALP), https://www.my-personaltrainer.it/salute/fosfatasi-alcalina.html#237244, Consultato il 05/11/2023.

[13] DREAM IN * 101, Le 4 fasi del sonno e tutte le tecniche per dormire bene, https://dreamin101.com/blogs/scegliere-il-piumone/fasi-del-sonno-e-riposo, Consultato il 05/11/2023.

[14] Premoneo, Database SQL o NoSQL, quale scegliere?, https://premoneo.com/2022/10/05/database-sql-o-nosql-quale-scegliere/#:~:text=I%20DB%20SQL%20sono%20più,per%20grandi%20moli%20di%20dati, Consultato il 05/11/2023.

[15] John Kapantzakis, Scalable Path, When to Use NoSQL vs SQL: The Ultimate Guide for Choosing a Database, https://www.scalablepath.com/back-end/sql-vs-nosql, Consultato il 06/11/2023.

[16] Geekandjob, DBMS: i 9 Migliori Database da Usare nel 2023, https://blog.geekandjob.com/database/, Consultato il 06/11/2023.

[17] Wikipedia, Business rule engine, https://en.wikipedia.org/wiki/Business_rules_engine, Consultato il 06/11/2023.

[18] Wikipedia, Event condition action, https://en.wikipedia.org/wiki/Event_condition_action, Consultato il 06/11/2023.

[19] Drools, Drools Documentation, https://docs.drools.org/7.73.0.Final/drools-docs/html_single/index.html#decision-engine-con_decision-engine, Consultato il 07/11/2023.

[20] Davide Sottara, Università di Bologna, Drools, http://www.lia.deis.unibo.it/Courses/AI/fundamentalsAI2010-11/esercitazioni/Drools-ita.pdf, Consultato il 07/11/2023.

[21] Jboss.org, Chapter 5. The Rule Language, https://docs.jboss.org/drools/release/5.3.0.Final/drools-expert-docs/html/ch05.html#d0e2818, Consultato il 07/11/2023.

[22] MedicalNewsToday, How many steps should people take per day?, https://www.medicalnewstoday.com/articles/how-many-steps-should-you-take-a-day Consultato il 07/11/2023.

[23] Riccardo Borgacci, MyPersonalTrainer, Rapporto colesterolo Totale/HDL https://www.my-personaltrainer.it/salute/rapporto-colesterolo.html, Consultato il 07/11/2023.

[24] Wikipedia, Rapporto BUN-creatinina, https://it.wikipedia.org/wiki/Rapporto_BUN-creatinina, Consultato il 07/11/2023.

[25] Top Life Project, COLOSTEROLO HDL E TRIGLICERIDI: occhio al rapporto, https://toplifeproject.com/blog/2020/01/02/colesterolo-hdl-e-trigliceridi-occhio-al-rapporto/#:~:text=Oggi%20si%20ritiene%20che%20il,per%20il%20tuo%20colesterolo%20HDL).&text=6%20–%20troppo%20elevato, Consultato il 07/11/2023.

[26] Fattorie Girau, Calcolo delle calorie: ecco come fare, https://www.fattoriegirau.com/magazine/filosofia-alimentare/educazione-alimentare/calcolo-calorie/, Consultato il 07/11/2023.

[27] Daniele Fumi, My Lab Nutrition Group, Quanti carboidrati assumere al giorno?, https://www.mylabnutrition.net/blog/quanti-carboidrati-assumere-al-giorno/, Consultato il 07/11/2023.

[28] Dove e Come Mi Curo, Proteine: funzioni e giusto apporto nella dieta, https://www.micuro.it/enciclopedia/alimentazione/proteine#:~:text=Secondo%20questi%20studi%2C%20un%20apporto,una%20dieta%20da%202000%20calorie, Consultato il 07/11/2023.

[29] MyPersonalTrainer, Fabbisogno di grassi, https://www.my-personaltrainer.it/grassi-fabbisogno1.html, Consultato il 08/11/2023.

[30] MyPersonalTrainer, Percentuali di Grassi Saturi e Insaturi nella Dieta, https://www.my-personaltrainer.it/nutrizione/grassi-saturi-insaturi1.html, Consultato il 08/11/2023.

[31] Fondazione Veronesi, Grassi trans: addio per tutti nel 2023?, https://www.fondazioneveronesi.it/magazine/articoli/alimentazione/grassi-trans-lorganizzazione-mondiale-della-sanita-vuole-eliminarli-entro-il-2023, Consultato il 08/11/2023.

[32] MyPersonalTrainer, Colesterolo e alimentazione, https://www.my-personaltrainer.it/nutrizione/colesterolo.html#:~:text=Quanto%20colesterolo%20assumere%20quotidianamente%3F,colesterolo%20dovrebbe%20essere%20più%20contenuto, Consultato il 08/11/2023.

[33] Ministero della Salute, Cosa fare per ridurre il consumo di sale, https://www.salute.gov.it/portale/temi/p2_6.jsp?id=5520&area=stiliVita&menu=alimentation#:~:text=L%27Organizzazione%20mondiale%20della%20Sanità,grammi%20al%20giorno%20di%20sodio., Consultato il 08/11/2023.

[34] Istituto superiore di sanità, Sale e potassio: il consumo in italia, https://www.cuore.iss.it/prevenzione/ProgettoMinisal#:~:text=L%27OMS%20raccomanda%20di%20assumere,di%20verdura%2C%20frutta%20e%20legumi, Consultato il 08/11/2023.

[35] MyPersonalTrainer, Fibre alimentari: cosa sono, a cosa servono e dove si trovano, https://www.my-personaltrainer.it/nutrizione/fibre-alimentari.html#:~:text=Quanta%20Fibra%20Alimentare%20Assumere%3F,ferro%2C%20calcio%20e%20zinco), Consultato il 08/11/2023.

[36] Project Invictus, Quanti zuccheri al giorno si possono mangiare?, https://www.projectinvictus.it/quanti-zuccheri-al-giorno/, Consultato il 08/11/2023.

[37] Fondazione Veronesi, L'Oms: "Limitare gli zuccheri aggiunti. Non più di 12 cucchiaini al giorno", https://www.fondazioneveronesi.it/magazine/articoli/alimentazione/loms-limitare-gli-zuccheri-aggiunti-non-piu-di-12-cucchiaini-al-giorno, Consultato il 08/11/2023.

[38] Humanitas, Vitamina D, https://www.humanitas.it/enciclopedia/vitamine/vitamina-d/, Consultato il 08/11/2023.

[39] Humanitas, Vitamina A (retinolo), https://www.humanitas.it/enciclopedia/vitamine/vitamina-a-retinolo/, Consultato il 08/11/2023.

[40] Farmacia 33, Quanta vitamina c assumere al giorno? Migliori prodotti., https://farmacia33.it/blog/1617-Quanta-vitamina-c-assumere-al-giorno-Migliori-prodotti#:~:text=Dose%20Giornaliera%20Raccomandata%20di%20Vitamina%20C%20per%20Adulti&text=In%20generale%2C%20gli%20adulti%20sani,di%20vitamina%20C%20al%20giorno, Consultato il 08/11/2023.

[41] Riccardo Borgacci, MyPersonalTrainer, Calcio e Osteoporosi, https://www.my-personaltrainer.it/calcio-osteoporosi.htm, Consultato il 08/11/2023.

[42] Humanitas, Ferro, https://www.humanitas.it/enciclopedia/sali-minerali/ferro/, Consultato il 08/11/2023.

[43] Società italiana di Nutrizione Umana, Sodio, Potassio e Iodio nella dieta degli italiani, https://sinu.it/wp-content/uploads/2019/07/MINISAL-sintesi-risultati.pdf, Consultato il 08/11/2023.

[44] Project Invictus, Fabbisogno proteico: come calcolarlo?, https://www.projectinvictus.it/fabbisogno-proteico/, Consultato il 08/11/2023.

[45] Monteverde Lab, Bilancio calorico e dimagrimento, capiamo il concetto!, https://monteverdelab.it/bilancio-calorico-e-dimagrimento/#:~:text=Per%20bilancio%20calorico%20si%20intende,la%20termogenesi%20stimolata%20dalla%20dieta, Consultato il 08/11/2023.

[46] Riccardo Borgacci, MyPersonalTrainer, Frequenza Cardiaca a Riposo: Quanto deve Essere per la Salute?, https://www.my-personaltrainer.it/allenamento/frequenza-cardiaca/frequenza-cardiaca-riposo.html#220181, Consultato il 08/11/2023.

[47] MyPersonalTrainer, Quante Ore Dormire in Base all'Età, https://www.my-personaltrainer.it/benessere/quante-ore-dormire-eta.html#253086, Consultato il 08/11/2023.

[48] Ministero della Salute, Calcolo Indice massa corporea – IMC (BMI – Body mass index), https://www.salute.gov.it/portale/nutrizione/dettaglioIMCNutrizione.jsp?lingua=italiano&id=5479&area=nutrizione&menu=vuoto, Consultato il 08/11/2023.

[49] Fitbit Developer Console,
https://dev.fitbit.com/apps/new,
Consultato il 10/11/2023.

[50] Fibit Developer, Application Design,
https://dev.fitbit.com/build/reference/web-api/developer-guide/application-design/,
Consultato il 10/11/2023.

[51] Luigi Sbriz, Cyber Security 360, OAuth 2.0: cos'è e come funziona lo standard "aperto" per l'autenticazione sicura online,
https://www.cybersecurity360.it/soluzioni-aziendali/oauth-2-0-cose-e-come-funziona-lo-standard-aperto-per-lautenticazione-sicura-online/,
Consultato il 10/11/2023.

[52] Fitbit Developer, Web API Reference,
https://dev.fitbit.com/build/reference/web-api,
Consultato il 10/11/2023.

[53] Fitbit Developer, Web API Reference, Get Sleep Log by Date Range,
https://dev.fitbit.com/build/reference/web-api/sleep/get-sleep-log-by-date-range/,
Consultato il 10/11/2023.

[54] Fitbit Developer, Web API Reference, Get Hearth-Rate Time Series by Date Range,
https://dev.fitbit.com/build/reference/web-api/heartrate-timeseries/get-heartrate-timeseries-by-date-range/,
Consultato il 10/11/2023.

[55] Fitbit Developer, Web API Reference, Get Activity Intraday by Interval,
https://dev.fitbit.com/build/reference/web-api/intraday/get-activity-intraday-by-interval/,
Consultato il 10/11/2023.

[56] FatSecret Platform,
https://platform.fatsecret.com/platform-api,
Consultato il 13/11/2023.

[57] FatSecret Platform, Authentication,
https://platform.fatsecret.com/docs/guides/authentication,
Consultato il 13/11/2023.

[58] FatSecret Platform, OAuth 1.0,
https://platform.fatsecret.com/docs/guides/authentication/oauth1,
Consultato il 13/11/2023.

[59] IETF Datatracker, RFC 3986 – Uniform Resource Identifier (URI),
https://datatracker.ietf.org/doc/html/rfc3986,
Consultato il 13/11/2023.

[60] IETF Datatracker, RFC 2045 – Multipurpose Internet Mail
Extensions (MIME),
https://datatracker.ietf.org/doc/html/rfc2045,
Consultato il 13/11/2023.

Allegato 1 – Recommender System Rules

Activity

Single Parameter

Steps

- **Steps are ok**: If for given patient and data, steps are above threshold.
- **Steps are null**: If for given patient and data, steps are null.
- **Steps have null threshold**: If for given patient and data, step's threshold is null.
- **Steps low**: If for given patient and data, steps are under threshold.

Calories out

- **Calories out are ok**: If for given patient and data, calories out are above threshold.
- **Calories out are null**: If for given patient and data, calories out are null.
- **Calories out have null threshold**: If for given patient and data, calories' threshold is null.
- **Calories low**: If for given patient and data, calories out are under threshold.

MedicalAnalysis

Single Parameter

Eritrociti

- **Eritrociti are ok:** If for given patient and data, eritrociti are between trheshold_min and threshold_max.
- **Eritrociti are null:** If for given patient and data, eritrociti are null.
- **Eritrociti have min and max thresholds null:** If for given patient and

data, eritrociti' thresholds are null.

- **Eritrociti low:** If for given patient and data, eritrociti are under threshold_min.
- **Eritrociti high:** If for given patient and data, eritrociti are above threshold_max.
- **Eritrociti bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

Emoglobina

- **Emoglobina is ok:** If for given patient and data, emoglobina is between trheshold_min and threshold_max.
- **Emoglobina is null:** If for given patient and data, emoglobina is null.
- **Emoglobina has min and max thresholds null:** If for given patient and data, emoglobina's thresholds are null.
- **Emoglobina low:** If for given patient and data, emoglobina is under threshold_min.
- **Emoglobina high:** If for given patient and data, emoglobina is above threshold_max.
- **Emoglobina bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

MCV

- **MCV is ok:** If for given patient and data, MCV is between trheshold_min and threshold_max.
- **MCV is null:** If for given patient and data, MCV is null.
- **MCV has min and max thresholds null:** If for given patient and data, MCV's thresholds are null.
- **MCV low:** If for given patient and data, MCV is under threshold_min.
- **MCV high:** If for given patient and data, MCV is above threshold_max.
- **MCV bad thresholds:** If for given patient and data, threshold_min is

above threshold_max.

HT

- **HT is ok:** If for given patient and data, HT is between trheshold_min and threshold_max.
- **HT is null:** If for given patient and data, HT is null.
- **HT has min and max thresholds null:** If for given patient and data, HT's thresholds are null.
- **HT low:** If for given patient and data, HT is under threshold_min.
- **HT high:** If for given patient and data, HT is above threshold_max.
- **HT bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

Leucociti

- **Leucociti are ok:** If for given patient and data, leucociti are between trheshold_min and threshold_max.
- **Leucociti are null:** If for given patient and data, leucociti are null.
- **Leucociti have min and max thresholds null:** If for given patient and data, leucociti' thresholds are null.
- **Leucociti low:** If for given patient and data, leucociti are under threshold_min.
- **Leucociti high:** If for given patient and data, leucociti are above threshold_max.
- **Leucociti bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

Piastrine

- **Piastrine are ok:** If for given patient and data, piastrine are between

trheshold_min and threshold_max.

- **Piastrine are null:** If for given patient and data, piastrine are null.
- **Piastrine have min and max thresholds null:** If for given patient and data, piastrine' thresholds are null.
- **Piastrine low:** If for given patient and data, piastrine are under threshold_min.
- **Piastrine high:** If for given patient and data, piastrine are above threshold_max.
- **Piastrine bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

Glicemia

- **Glicemia is ok:** If for given patient and data, Glicemia is between trheshold_min and threshold_max.
- **Glicemia is null:** If for given patient and data, Glicemia is null.
- **Glicemia has min and max thresholds null:** If for given patient and data, Glicemia's thresholds are null.
- **Glicemia low:** If for given patient and data, Glicemia is under threshold_min.
- **Glicemia high:** If for given patient and data, Glicemia is above threshold_max.
- **Glicemia bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

Urea

- **Urea is ok:** If for given patient and data, Urea is between trheshold_min and threshold_max.
- **Urea is null:** If for given patient and data, Urea is null.
- **Urea has min and max thresholds null:** If for given patient and data, Urea's thresholds are null.

- **Urea low:** If for given patient and data, Urea is under threshold_min.
- **Urea high:** If for given patient and data, Urea is above threshold_max.
- **Urea bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

Na

- **Na is ok:** If for given patient and data, Na is between trheshold_min and threshold_max.
- **Na is null:** If for given patient and data, Na is null.
- **Na has min and max thresholds null:** If for given patient and data, Na's thresholds are null.
- **Na low:** If for given patient and data, Na is under threshold_min.
- **Na high:** If for given patient and data, Na is above threshold_max.
- **Na bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

K

- **K is ok:** If for given patient and data, K is between trheshold_min and threshold_max.
- **K is null:** If for given patient and data, K is null.
- **K has min and max thresholds null:** If for given patient and data, K's thresholds are null.
- **K low:** If for given patient and data, K is under threshold_min.
- **K high:** If for given patient and data, K is above threshold_max.
- **K bad thresholds:** If for given patient and data, threshold_min is above threshold_max

Creatinina

- **Creatinina is ok:** If for given patient and data, Creatinina is between trheshold_min and threshold_max.
- **Creatinina is null:** If for given patient and data, Creatinina is null.
- **Creatinina has min and max thresholds null:** If for given patient and data, Creatinina's thresholds are null.
- **Creatinina low:** If for given patient and data, Creatinina is under threshold_min.
- **Creatinina high:** If for given patient and data, Creatinina is above threshold_max.
- **Creatinina bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

Colesterolo Totale

- **Colesterolo Totale is ok:** If for given patient and data, Colesterolo Totale is between trheshold_min and threshold_max.
- **Colesterolo Totale is null:** If for given patient and data, Colesterolo Totale is null.
- **Colesterolo Totale has min and max thresholds null:** If for given patient and data, Colesterolo Totale's thresholds are null.
- **Colesterolo Totale low:** If for given patient and data, Colesterolo Totale is under threshold_min.
- **Colesterolo Totale high:** If for given patient and data, Colesterolo Totale is above threshold_max.
- **Colesterolo Totale bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

Colesterolo HDL

- **Colesterolo HDL is ok:** If for given patient and data, Colesterolo HDL

is between trheshold_min and threshold_max.

- **Colesterolo HDL is null:** If for given patient and data, Colesterolo HDL is null.
- **Colesterolo HDL has min and max thresholds null:** If for given patient and data, Colesterolo HDL's thresholds are null.
- **Colesterolo HDL low:** If for given patient and data, Colesterolo HDL is under threshold_min.
- **Colesterolo HDL high:** If for given patient and data, Colesterolo HDL is above threshold_max.
- **Colesterolo HDL bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

Trigliceridi

- **Trigliceridi are ok:** If for given patient and data, trigliceridi are between trheshold_min and threshold_max.
- **Trigliceridi are null:** If for given patient and data, trigliceridi are null.
- **Trigliceridi have min and max thresholds null:** If for given patient and data, trigliceridi' thresholds are null.
- **Trigliceridi low:** If for given patient and data, trigliceridi are under threshold_min.
- **Trigliceridi high:** If for given patient and data, trigliceridi are above threshold_max.
- **Trigliceridi bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

PCR

- **PCR is ok:** If for given patient and data, PCR is between trheshold_min and threshold_max.
- **PCR is null:** If for given patient and data, PCR is null.
- **PCR has min and max thresholds null:** If for given patient and data,

PCR's thresholds are null.

- **PCR low:** If for given patient and data, PCR is under threshold_min.
- **PCR high:** If for given patient and data, PCR is above threshold_max.
- **PCR bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

Albumina

- **Albumina is ok:** If for given patient and data, Albumina is between trheshold_min and threshold_max.
- **Albumina is null:** If for given patient and data, Albumina is null.
- **Albumina has min and max thresholds null:** If for given patient and data, Albumina's thresholds are null.
- **Albumina low:** If for given patient and data, Albumina is under threshold_min.
- **Albumina high:** If for given patient and data, Albumina is above threshold_max.
- **Albumina bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

Linfociti

- **Linfociti are ok:** If for given patient and data, Linfociti are between trheshold_min and threshold_max.
- **Linfociti are null:** If for given patient and data, Linfociti are null.
- **Linfociti have min and max thresholds null:** If for given patient and data, Linfociti's thresholds are null.
- **Linfociti low:** If for given patient and data, Linfociti are under threshold_min.
- **Linfociti high:** If for given patient and data, Linfociti are above threshold_max.
- **Linfociti bad thresholds:** If for given patient and data, threshold_min

is above threshold_max.

RDW

- **RDW is ok:** If for given patient and data, RDW is between trheshold_min and threshold_max.
- **RDW is null:** If for given patient and data, RDW is null.
- **RDW has min and max thresholds null:** If for given patient and data, RDW's thresholds are null.
- **RDW low:** If for given patient and data, RDW is under threshold_min.
- **RDW high:** If for given patient and data, RDW is above threshold_max.
- **RDW bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

ALP

- ***ALP is ok:*** *If for given patient and data, ALP is between trheshold_min and threshold_max.*
- **ALP is null:** If for given patient and data, ALP is null.
- **ALP has min and max thresholds null:** If for given patient and data, ALP's thresholds are null.
- **ALP low:** If for given patient and data, ALP is under threshold_min.
- **ALP high:** If for given patient and data, ALP is above threshold_max.
- **ALP bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

Multi Parameters

Colesterolo Totale/HDL Ratio

- **Colesterolo Totale/HDL Ratio low:** If for given patient and data

Colesterolo Totale/HDL Ratio is under 5.

- **Colesterolo Totale/HDL Ratio moderate:** If for given patient and data Colesterolo Totale/HDL Ratio is >=5 and <=10.
- **Colesterolo Totale/HDL Ratio high:** If for given patient and data Colesterolo Totale/HDL Ratio is above 10.

Urea/Creatinina Ratio

- **Urea/Creatinina Ratio low:** If for given patient and data Urea/Creatinina Ratio is under 40.
- **Urea/Creatinina Ratio normal:** If for given patient and data Urea/Creatinina Ratio is >=40 and <=100.
- **Urea/Creatinina Ratio high:** If for given patient and data Urea/Creatinina Ratio is above 100.

Trigliceridi/Colesterolo HDL Ratio

- **Ideal Trigliceridi/Colesterolo HDL Ratio:** If for given patient and data Trigliceridi/Colesterolo HDL Ratio is <= 2.
- **Trigliceridi/Colesterolo HDL Ratio high:** If for given patient and data Trigliceridi/Colesterolo HDL Ratio is >2 and <=6.
- **Trigliceridi/Colesterolo HDL Ratio very high:** If for given patient and data Trigliceridi/Colesterolo HDL Ratio is above 6.

Food

Single Parameter

Calories in

- **Calories in are ok:** If for given patient and data, calories in are under threshold.
- **Calories in are null:** If for given patient and data, calories in are null.

- **Calories in have null threshold:** If for given patient and data, calories' threshold is null.
- **Calories in high:** If for given patient and data, calories in are above threshold.

Carbohydrates

- **Carbohydrates are ok:** If for given patient and data, Carbohydrates are under threshold.
- **Carbohydrates are null:** If for given patient and data, Carbohydrates are null.
- **Carbohydrates have null threshold:** If for given patient and data, Carbohydrates' threshold is null.
- **Carbohydrates high:** If for given patient and data, Carbohydrates are above threshold.

Proteins

- **Proteins are ok:** If for given patient and data, Proteins are under threshold.
- **Proteins are null:** If for given patient and data, Proteins are null.
- **Proteins have null threshold:** If for given patient and data, Proteins' threshold is null.
- **Proteins high:** If for given patient and data, Proteins are above threshold.

Fats

- **Fats are ok:** If for given patient and data, Fats are under threshold.
- **Fats are null:** If for given patient and data, Fats are null.
- **Fats have null threshold:** If for given patient and data, Fats' threshold is null.

- **Fats high:** If for given patient and data, Fats are above threshold.

Saturated Fats

- **Saturated Fats are ok:** If for given patient and data, Saturated Fats are under threshold.
- **Saturated Fats are null:** If for given patient and data, Saturated Fats are null.
- **Saturated Fats have null threshold:** If for given patient and data, Saturated Fats' threshold is null.
- **Saturated Fats high:** If for given patient and data, Saturated Fats are above threshold.

Polyunsaturated Fats

- **Polyunsaturated Fats are ok:** If for given patient and data, Polyunsaturated Fats are under threshold.
- **Polyunsaturated Fats are null:** If for given patient and data, Polyunsaturated Fats are null.
- **Polyunsaturated Fats have null threshold:** If for given patient and data, Polyunsaturated Fats' threshold is null.
- **Polyunsaturated Fats high:** If for given patient and data, Polyunsaturated Fats are above threshold.

Monounsaturated Fats

- **Monounsaturated Fats are ok:** If for given patient and data, Monounsaturated Fats are under threshold.
- **Monounsaturated Fats are null:** If for given patient and data, Monounsaturated Fats are null.
- **Monounsaturated Fats have null threshold:** If for given patient and data, Monounsaturated Fats' threshold is null.

- **Monounsaturated Fats high:** If for given patient and data, Monounsaturated Fats are above threshold.

Trans Fats

- **Trans Fats are ok:** If for given patient and data, Trans Fats are under threshold.
- **Trans Fats are null:** If for given patient and data, Trans Fats are null.
- **Trans Fats have null threshold:** If for given patient and data, Trans Fats' threshold is null.
- **Trans Fats high:** If for given patient and data, Trans Fats are above threshold.

Cholesterol

- **Cholesterol is ok:** If for given patient and data, Cholesterol is under threshold.
- **Cholesterol is null:** If for given patient and data, Cholesterol is null.
- **Cholesterol has null threshold:** If for given patient and data, Cholesterol' threshold is null.
- **Cholesterol high:** If for given patient and data, Cholesterol is above threshold.

Sodium

- **Sodium is ok:** If for given patient and data, Sodium is under threshold.
- **Sodium is null:** If for given patient and data, Sodium is null.
- **Sodium has null threshold:** If for given patient and data, Sodium' threshold is null.
- **Sodium high:** If for given patient and data, Sodium is above threshold.

Potassium

- **Potassium is ok:** If for given patient and data, Potassium is under threshold.
- **Potassium is null:** If for given patient and data, Potassium is null.
- **Potassium has null threshold:** If for given patient and data, Potassium' threshold is null.
- **Potassium high:** If for given patient and data, Potassium is above threshold.

Fibers

- **Fibers are ok:** If for given patient and data, Fibers are under threshold.
- **Fibers are null:** If for given patient and data, Fibers are null.
- **Fibers have null threshold:** If for given patient and data, Fibers' threshold is null.
- **Fibers high:** If for given patient and data, Fibers are above threshold.

Sugars

- **Sugars are ok:** If for given patient and data, Sugars are under threshold.
- **Sugars are null:** If for given patient and data, Sugars are null.
- **Sugars have null threshold:** If for given patient and data, Sugars' threshold is null.
- **Sugars high:** If for given patient and data, Sugars are above threshold.

Added Sugars

- **Added Sugars are ok:** If for given patient and data, Added Sugars are under threshold.

- **Added Sugars are null:** If for given patient and data, Added Sugars are null.
- **Added Sugars have null threshold:** If for given patient and data, Added Sugars' threshold is null.
- **Added Sugars high:** If for given patient and data, Added Sugars are above threshold.

Vitamin D

- **Vitamin D is ok:** If for given patient and data, Vitamin D is under threshold.
- **Vitamin D is null:** If for given patient and data, Vitamin D is null.
- **Vitamin D has null threshold:** If for given patient and data, Vitamin D' threshold is null.
- **Vitamin D high:** If for given patient and data, Vitamin D is above threshold.

Vitamin C

- **Vitamin C is ok:** If for given patient and data, Vitamin C is under threshold.
- **Vitamin C is null:** If for given patient and data, Vitamin C is null.
- **Vitamin C has null threshold:** If for given patient and data, Vitamin C' threshold is null.
- **Vitamin C high:** If for given patient and data, Vitamin C is above threshold.

Vitamin A

- **Vitamin A is ok:** If for given patient and data, Vitamin A is under threshold.
- **Vitamin A is null:** If for given patient and data, Vitamin A is null.

- **Vitamin A has null threshold:** If for given patient and data, Vitamin A' threshold is null.
- **Vitamin A high:** If for given patient and data, Vitamin A is above threshold.

Calcium

- **Calcium is ok:** If for given patient and data, Calcium is under threshold.
- **Calcium is null:** If for given patient and data, Calcium is null.
- **Calcium has null threshold:** If for given patient and data, Calcium' threshold is null.
- **Calcium high:** If for given patient and data, Calcium is above threshold.

Iron

- **Iron is ok:** If for given patient and data, Iron is under threshold.
- **Iron is null:** If for given patient and data, Iron is null.
- **Iron has null threshold:** If for given patient and data, Iron' threshold is null.
- **Iron high:** If for given patient and data, Iron is above threshold.

Multi Parameters

Sodium/Potassium Ratio

- **Sodium/Potassium Ratio is ok:** If for given patient and data Sodium/Potassium Ratio is under 1.
- **Sodium/Potassium Ratio is high:** If for given patient and data Sodium/Potassium Ratio is >=1.

Fat proportion

- **Fat proportion is ok:** If for given patient and data, saturated fats are under 25% of total fats, monounsaturated fats under 55% and polyunsaturated fats under 20%.
- **Polyunsaturated Fat proportion high:** If for given patient and data, polyunsaturated fats are above 20% of total fats.
- **Saturated Fat proportion high:** If for given patient and data, saturated fats are above 25% of total fats.
- **Monounsaturated Fat proportion high:** If for given patient and data, monounsaturated fats are above 55% of total fats.
- **Polyunsaturated and Saturated Fat proportion high:** If for given patient and data, polyunsaturated fats are above 20% of total fats and saturated fats are above 25%.
- **Saturated and Monounsaturated Fat proportion high:** If for given patient and data, saturated fats are above 25% of total fats and monounsaturated fats are above 55%.
- **Monounsaturated and Polyunsaturated Fat proportion high:** If for given patient and data, monounsaturated fats are above 55% of total fats and polyunsaturated fats are above 20%.
- **Saturated Monounsaturated and Polyunsaturated Fat proportion high:** If for given patient and data, saturated fats are above 25% of total fats, monounsaturated fats are above 55% and polyunsaturated are above 20%.

Carbs Proteins Fats proportion

- **Carbs Proteins Fats proportion is ok:** If for given patient and data, Carbs are under 50% of total sum, proteins under 20% and fats under 30%.
- **Proteins proportion high:** If for given patient and data, proteins fats are above 20% of total sum.
- **Carbs proportion high:** If for given patient and data, carbs are above

50% of total sum.

- **Fats proportion high:** If for given patient and data, fats are above 30% of total sum.
- **Carbs and Proteins proportion high:** If for given patient and data, carbs are above 50% of total sum and proteins are above 20%.
- **Carbs and Fats proportion high:** If for given patient and data, carbs are above 50% of total sum and fats are above 30%.
- **Proteins and Fats proportion high:** If for given patient and data, proteins are above 20% of total sum and fats are above 30%.
- **Carbs Proteins Fats proportion high:** If for given patient and data, carbs are above 50% of total sum, proteins are above 20% and fats are above 30%.

Calories balance = calories in - calories out

- **Calories in more than calories out:** If for given patient and data, calories in - calories out > 0.
- **Calories in less than calories out:** If for given patient and data, calories in - calories out < 0.
- **Calories in equal to calories out:** If for given patient and data, calories in - calories out = 0.

Hr

Single Parameter

Rest Hr

- **Rest Hr is ok:** If for given patient and data, Rest Hr is between trheshold_min and threshold_max.
- **Rest Hr is null:** If for given patient and data, Rest Hr is null.
- **Rest Hr has min and max thresholds null:** If for given patient and data, Rest Hr's thresholds are null.

- **Rest Hr low:** If for given patient and data, Rest Hr is under threshold_min.
- **Rest Hr high:** If for given patient and data, Rest Hr is above threshold_max.
- **Rest Hr bad thresholds:** If for given patient and data, threshold_min is above threshold_max.

Sleep

Single Parameter

Sleep time

- **Sleep time is ok:** If for given patient and data, sleep time is above threshold.
- **Sleep time is null:** If for given patient and data, sleep time is null.
- **Sleep time has null threshold:** If for given patient and data, sleep time's threshold is null.
- **Sleep time low:** If for given patient and data, sleep time is under threshold.

Weight

Single Parameter

Weight

- **Weight is ok:** If for given patient and data, weight is under threshold.
- **Weight is null:** If for given patient and data, weight is null.
- **Weight has null threshold:** If for given patient and data, weight's threshold is null.
- **Weight high:** If for given patient and data, weight is above threshold.

BMI

- **BMI is ok:** If for given patient and data, BMI is under threshold.
- **BMI is null:** If for given patient and data, BMI is null.
- **BMI has null threshold:** If for given patient and data, BMI's threshold is null.
- **BMI high:** If for given patient and data, BMI is above threshold.