Politecnico di Torino

POLITECNICO DI TORINO

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

Sviluppo Backend e Recommender System per monitorare il benessere degli assistiti medici

Relatore:
Prof. Maurizio MORISIO

Candidato:
Rocco Luca IAMELLO

Ai miei nonni, i pilastri della mia vita

Indice

Elenco delle tabelle III
Elenco delle formule IV
Elenco degli allegati IV
Elenco delle figure V

1) Presentazione 1
1.1) Obiettivi 1
1.2) Metodologia, Architettura e Divisione del Lavoro 2
1.2.1) Architettura 2
1.2.2) Backend 5
1.2.3) Frontend 6

2) Manutenzione di un applicativo server esistente 8
2.1) Sicurezza e Validazione 9
2.2) Documentazione e schemi 10
2.2.1) Diagramma controllo accessi 11
2.2.2) Schema Database 12

3) Espansione di un applicativo server esistente 14
3.1) Peso 14
3.1.1) Implementazione API 15
3.1.2) Osservazioni 16
3.2) Analisi mediche 16
3.2.1) Implementazione API 18
3.2.2) Osservazioni 20
3.3) Dati generati tramite Fitbit 20
3.3.1) Integrazione Fitbit con HealthApp 21
3.3.2) Implementazione API 24
3.3.3) Osservazioni 27
3.4) Cibo e Ricette 27
3.4.1) OpenFoodFacts 28
3.4.2) FatSecret 29
3.4.3) Integrazione FatSecret con HealthApp 30
3.4.4) Implementazione API 32
3.4.5) Ricette 34
3.4.6) Osservazioni 36

4) Recommendations – introduzione e scelta business rule engine 37
4.1) Introduzione al sistema delle recommendations 37
4.2) Scelta business rule engine 38
4.2.1) DecisionRules 39
4.2.2) Hyperion 39
4.2.3) Evrete 39
4.2.4) Drools 39
4.3) Specifiche di Drools 40

5) Implementazione sistema recommendations 45
5.1) Definizione regole 46
5.1.1) Attività 47
5.1.2) Analisi Mediche 48
5.1.3) Cibo 57
5.1.4) Frequenza Cardiaca 66
5.1.5) Sonno 66
5.1.6) Peso 67
5.2) Definizione API 68
5.3) Esecuzione periodica e generazione report 76
5.4) Performance e Conclusioni 77

6) Test su dataset esistenti 79
6.1) Ricerca dataset esistenti 79
6.2) Implementazione adattatori 80
6.3) Risultati test su Simula Dataset 81
6.4) Risultati test su Kaggle Dataset 84
6.5) Risultati test su Dietary Dataset 88
6.6) Risultati test su dati autoprodotti 89
6.7) Conclusioni 90

7) Deployment, spunti di miglioramento e conclusioni 91
7.1) Deployment 91
7.1.1) Containerizing 91
7.1.2) Execution 92
7.2) Spunti di miglioramento 93
7.3) Conclusioni 94

8) Bibliografia e Sitografia 96

Elenco delle tabelle

1. Valori d'interesse analisi mediche 17
2. Valori nutrizionali alimentazione 30
3. Categorie alimenti 32
4. Tipologie di pasto 33
5. Risultati Simula Dataset 82
6. Risultati Kaggle Dataset 84
7. Risultati Dietary Dataset 89
8. Risultati dati autoprodotti 90

Elenco delle formule

1. Formula BMI 15
2. Formula BMI in cm 15

Elenco degli allegati

Allegato 1. Guida registrazione nuovo paziente su HealthApp 102

Elenco delle figure

1. Architettura a tre livelli 2
2. Diagramma architettura (Deployment Diagram) 3
3. Specializzazioni Componente 8
4. Gerarchia Endpoint REST 8
5. Diagramma Controllo Accessi 11
6. Schema Database 12
7. Step OAuth 2.0 tra i server (intra-server) 22
8. Form di autorizzazione Fitbit 23
9. Valori nutrizionali *"Cetriolo"* 28
10. Valori nutrizionali *"Chicchi di riso soffiato al cioccolato"* 28
11. Valori nutrizionali per 100g crema spalmabile alle nocciole barattolo 1kg 29
12. Valori nutrizionali per 100g crema spalmabile alle nocciole barattolo 800g 29
13. Schema Riassuntivo BRE 38
14. Estratto report pdf settimanale 76
15. Lista pazienti, tempo di risposta 77
16. Analisi di tipo attività, tempo di risposta 77
17. Analisi completa, tempo di risposta 78

1. Presentazione

1.1 Obiettivi

L'idea di HealthApp nasce nell'ambito di una collaborazione tra il DAUIN e alcuni medici dell'Azienda Ospedaliera Di Verona, con l'obiettivo di usare la tecnologia informatica in ambito medico.

Il progetto di tesi riguarda lo sviluppo di una applicazione software (con parte back end e front end) per supportare il benessere dei pazienti affetti e non da patologie, aiutandoli a tenere sotto controllo i parametri riguardanti: **alimentazione, movimento, salute e sonno**. I pazienti stessi rappresentano quindi gli utilizzatori principali, seguiti dai medici, i quali assumono il ruolo di "controllori", avendo facoltà di gestire e monitorare in totale flessibilità i propri assistiti.

In seguito ad un'ampia fase di studio e progettazione in cui, grazie alla collaborazione del professore con noi cinque tesisti, partendo dalle quattro macroaree precedentemente elencate si è riusciti ad individuare i casi d'uso principali, operare le giuste scelte architetturali ed implementative, sviluppare ed integrare perfettamente le differenti funzionalità in un unico applicativo.

Attraverso l'utilizzo di quest'ultimo, il paziente può così curare il proprio stile di vita, tenendo giornalmente traccia dei risultati, visualizzandone i progressi e ricevendo suggerimenti su come migliorare i parametri maggiormente critici.

Esaminando con un livello di maggiore dettaglio le differenti aree, è possibile affermare che:

- Per quanto riguarda l'alimentazione il paziente può tener traccia degli alimenti consumati durante i pasti e ricevere informazioni sui valori nutrizionali relativamente alla propria dieta o alla patologia, al fine di evitare la consumazione di alimenti non propriamente indicati.

- Relativamente al movimento, il paziente può analizzare i dati rilevati sull'attività fisica con lo scopo di osservarne il trend temporale e di valutarne la costanza dei risultati ottenuti, nonché i progressi stabiliti.

- La sezione riguardante la salute permette di tenere sotto controllo parametri dalla diversa caratura, come il battito cardiaco, la massa, l'indice di massa corporea (BMI) e i valori delle analisi mediche. Il tutto, ovviamente, con una frequenza differente e con una totale flessibilità di inserimento da parte dell'utente, che sia il medico stesso o l'assistito.

- Infine, attraverso il monitoraggio del sonno è possibile valutare la qualità e l'andamento temporale dello stesso con lo scopo di offrire anche in questa sezione spunti di riflessione e miglioramento per l'assistito.

Superando ora questa fase iniziale prettamente descrittiva, si discuteranno le scelte tecniche effettuate illustrandole e motivandole.

1.2 Metodologia, Architettura e Divisione del Lavoro

Durante le fasi di progettazione e sviluppo è risultata evidente l'esigenza di suddividere il lavoro in differenti task, ciascuno caratterizzato da un peso differente e da una durata differente. Per tale motivo la metodologia di lavoro maggiormente appropriata è quella Agile rispetto a quella Waterfall.

A motivare tale scelta vi sono anche ulteriori ragioni. Innanzitutto, vengono effettuati degli incontri settimanali di pianificazione con il professore e gli altri tesisti, utili a pianificare uno "*sprint*", l'unità base di lavoro in Agile.

I requisiti non sono inoltre del tutto espliciti, ma emergono in seguito a differenti colloqui con i medici o in seguito a periodi di brainstorming. Di conseguenza, è necessario un approccio che miri alla flessibilità, in quanto le specifiche di sviluppo sono in continua evoluzione.

Infine, si sceglie di comune accordo di concentrarsi sullo sviluppo vero e proprio dell'applicativo, producendo solo una documentazione minima e necessaria, senza soffermarsi più di tanto sulla parte esplicativa.

Tali motivazioni volgono a favore dell'approccio Agile e scartano Waterfall. Per supportare al meglio la metodologia scelta si è optato per un'architettura a tre livelli [1] (*three tier architecture*) illustrata di seguito.

1.2.1 Architettura

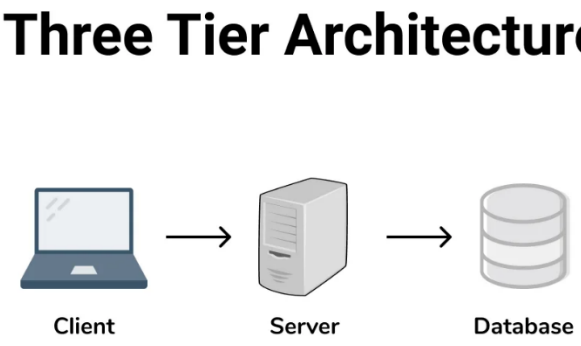


Figura 1 - Architettura a tre livelli

La *three tier architecture* si compone di tre differenti livelli logici e fisici [2]:

- *Data Tier*: in tale sezione vengono memorizzati e gestiti i dati utilizzati dall'applicazione. Nel caso di HealthApp si è optato per un Database Relazionale basato su linguaggio SQL

- *Application Tier*: comunemente chiamato "Server", rappresenta il cuore dell'applicazione dove le informazioni vengono processate e preparate alla presentazione. Tale livello funge da intermediario tra i dati e il client.

- *Presentation Tier*: rappresenta l'interfaccia utente (a cui un client accederà) in cui le informazioni ricevute vengono presentate all'utente, il quale può interagire e sulla base di esse prendere decisioni. All' interno dell' applicazione esistono due tipologie di client: desktop (laptop, notebook ecc.) e mobile (smartphone e tablet)

La figura seguente riassume e chiarifica l'architettura di HealthApp:

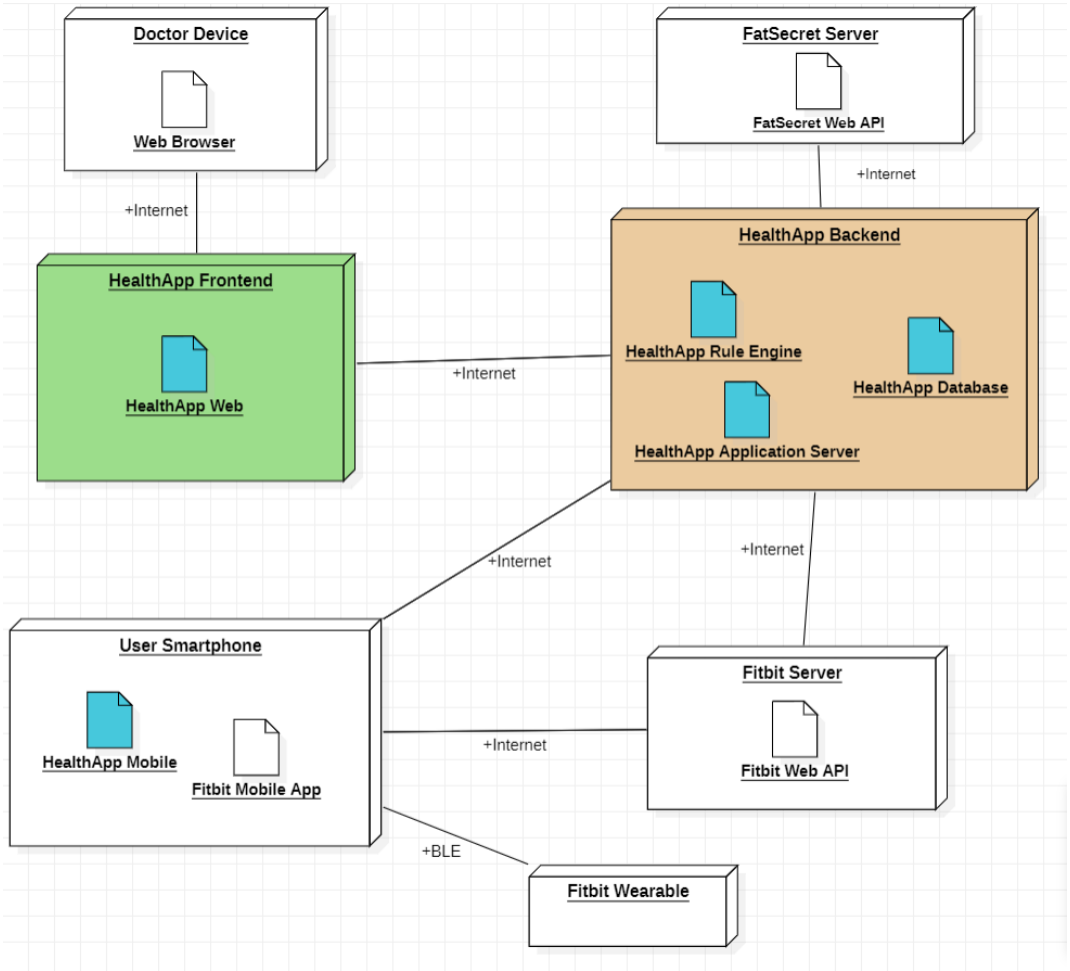


Figura 2 - Diagramma Architettura (Deployment Diagram)

Il diagramma illustra le relazioni tra l'hardware (i nodi) e il software (gli artifacts) all'interno dell'applicazione.

I due nodi principali, ossia back-end e front-end sono evidenziati rispettivamente in arancione e verde. In blu si hanno i componenti che sono oggetto diretto di sviluppo e che quindi rappresentano il fulcro del progetto, mentre in bianco i vari agenti esterni (es. servizi di terze parti, hardware ecc.) che interagiscono con l'applicazione al fine di garantirne il corretto funzionamento.

Entrando nel dettaglio, per quanto riguarda l'hardware si ha:

- **Fitbit Wearable**: braccialetto intelligente indossato dai pazienti che raccoglie i dati su movimento, salute e sonno.

- **User Smartphone**: Si interfaccia con il wearable attraverso un collegamento BLE (Bluetooth Low Energy) e rappresenta il dispositivo grazie al quale i pazienti utilizzano le funzionalità di HealthApp. Si interfaccia al backend mediante un collegamento ad Internet.

- **HealthApp Backend**: Rappresenta la macchina in cui viene eseguito il backend dell' applicazione. Oltre ad interfacciarsi con il frontend, si interfaccia anche con i server di terze parti (*Fitbit* e *FatSecret*) al fine di recuperare alcuni fra i dati necessari nelle differenti macroaree.

- **HealthApp Frontend**: Rappresenta la macchina in cui viene eseguito il frontend web dell'applicazione. Si interfaccia con il backend mediante collegamento ad Internet.

- **Doctor Device**: Si interfaccia con l'HealthApp Frontend attraverso un collegamento Internet e rappresenta il dispositivo (desktop, notebook, tablet etc.) grazie al quale i medici possono effettuare il monitoraggio sugli assistiti.

- **Fitbit Server**: memorizza i dati relativi a movimento, salute e sonno prodotti dai wearable dei vari pazienti e li rende disponibili all'HealthApp Backend mediante API.

- **FatSecret Server**: rende disponibile all'HealthApp Backend, mediante chiamata ad API, una base dati contenente milioni di cibi e ricette mondiali con i relativi valori nutrizionali.

Tra i moduli software si ha invece:

- **HealthApp Mobile**: In esecuzione sul dispositivo mobile ed utilizzata dai pazienti.

- **Fitbit Mobile App**: In esecuzione sul dispositivo mobile ed utilizzata dagli assistiti come strumento di connessione e trasmissione dati attraverso il wearable. Tale App proprietaria caricherà successivamente i dati sul Fitbit Server.

- **HealthApp Web**: In esecuzione sull'HealthApp Frontend ed utilizzata dai medici al fine di monitorare i pazienti.

- **Web Browser**: Artifact necessario a raggiungere l'HealthApp Frontend.

- **Fitbit Web API**: utilizzata dall'HealthApp Backend per interfacciarsi al server Fitbit e recuperare i dati di movimento, salute e sonno degli assistiti provenienti dall'app proprietaria.

- **FatSecret Web API**: utilizzata dall'HealthApp Backend per interfacciarsi al server FatSecret e ricercare gli alimenti consumati dai pazienti nonché recuperarne i valori nutrizionali degli stessi.

- **HealthApp Database**: struttura rappresentante la base dati dell'applicazione e contenente tutte le informazioni necessarie al corretto funzionamento di quest'ultima.

- **HealthApp Rule Engine**: modulo dedito all'analisi dei dati prodotti dagli assistiti e alla produzione di *Recommendations*, consigli per migliorare il benessere.

- **HealthApp Application Server**: Cuore dell'applicazione, in grado di gestire il controllo degli accessi, manipolare i dati nell'HealthApp Database, comunicare con i server di terze parti ed i frontend mobile e web. Tale comunicazione avviene mediante delle API che permettono di consultare le informazioni memorizzate dal server.

Si osserva che nel back-end di HealthApp dati e server coesistono in un'unica macchina, con l'obiettivo di implementare in una fase successiva di distribuzione tale separazione. La comunicazione tra Server e Client avviene attraverso delle API REST.

Conseguenza di questa scelta architetturale è l'emergere di due ruoli chiaramente distinti: la figura che si occupa del livello di presentazione (frontend developer) e quella che si occupa del livello applicativo e dati (backend developer). Nel contesto di HealthApp, vi è un'ulteriore suddivisione all'interno del livello presentazione partizionato nel sottolivello mobile ed in quello web.

La ripartizione del lavoro risulta dunque essere: due testisti dediti al frontend mobile, uno dedito al frontend web e gli ultimi due (me compreso) dediti al backend. Tuttavia, tale suddivisione non è rigida ma consente a ciascun membro del team di svolgere un ruolo differente da quello assegnatogli, qualora sia necessario.

Questo capitolo introduttivo si concluderà con una rapida illustrazione dei framework adottati nello sviluppo dei vari livelli dell'applicazione.

1.2.2 Backend

L'insieme di livello applicativo e dati, comunemente conosciuto come backend deve essere in grado di garantire alcuni requisiti minimi tra cui:

- Disponibilità continua

- Sicurezza nella memorizzazione e manipolazione delle informazioni sensibili

- Robustezza ad eventuali intrusioni esterne

- Consistenza nei dati consultati da diversi client

Volendo adottare uno schema relazionale con cui interfacciarsi, una possibile soluzione è rappresentata dal framework *Express* per *Node.js* per il livello applicativo con *SQL* per il livello dati. I punti di forza di

tale soluzione riguardano la sua semplicità ed immediatezza nell'utilizzo, nonché la flessibilità nel realizzare una configurazione ad hoc. Tuttavia, questa elevata semplicità si scontra con l'esigenza di avere una versione funzionante in tempi ragionevoli in quanto abbatte la produttività ma soprattutto necessita dell'implementazione "a mano" di ogni controllo di sicurezza, metodologia ampiamente sconsigliata in ambito informatico, in quanto espone con molta probabilità il sistema a falle sfruttabili dagli attaccanti.

Per tale motivo si è scelto di utilizzare per la parte applicativa *Spring*, un framework *Java*, combinato con il più recente linguaggio *Kotlin* insieme a *SQL* per la parte dati. Dal canto suo, *Spring* offre la possibilità di realizzare una soluzione che sia: scalabile, modulare (esiste un sistema di dipendenze in cui importare solo i moduli necessari), semplice da sviluppare, adatta ai requisiti ricercati, sicura (in quanto diversi controlli vengono automaticamente effettuati dal framework) e che abbatta il tempo di sviluppo, incrementando quindi la produttività.

Affiancato a *Spring*, è presente un altro framework sempre basato su *Java*, *Drools*. Esso svolge la funzione di *Business Rules Engine (BRE)* e permette di svolgere un'analisi sui dati memorizzati, con lo scopo di elaborare i suggerimenti da mostrare ai pazienti.

Il compito del backend, nella sua interezza, è quello di rispondere alle richieste dei client effettuate attraverso le API REST e di comunicare con i provider di servizi esterni adottati all'interno dell'applicazione (*FatSecret* per il cibo, *Fitbit* per il movimento) di cui si parlerà più approfonditamente in seguito.

1.2.3 Frontend

Il livello presentazione, come anticipato, si divide in due sottolivelli: mobile e web.

Il sottolivello web è realizzato mediante l'utilizzo del framework *React*, per *Javascript*. Esso offre un'elevata facilità di sviluppo mediante un approccio modulare (component-based) e rappresenta inoltre una soluzione versatile che si adatta ad entrambe le classi di utenti a cui è destinata: Medici e Pazienti.

I primi potranno gestire in semplicità i secondi, visualizzandone i dati più rilevanti, manipolando questi ultimi ed integrandoli con informazioni d'interesse.

I pazienti, invece, potranno visualizzare e manipolare solo i propri dati, senza la possibilità di entrare a conoscenza di informazioni riservate, ai fini di sicurezza e protezione di ogni utente.

Il sottolivello mobile è invece realizzato mediante l'utilizzo del framework *React-Native*, per *Javascript*. Tale scelta è motivata, oltre che per la modularità offerta nello sviluppo, soprattutto dalla possibilità di supportare in un colpo solo i due sistemi operativi mobile attualmente più diffusi: *iOS* e *Android*. La produttività viene quindi massimizzata, a discapito delle funzionalità implementabili, ristrette a quelle offerte da entrambi i sistemi mobile, ma più che sufficienti per gli scopi dell'applicazione.

Tale sottolivello è destinato esclusivamente ai pazienti, in quanto lo si ritiene indispensabile per effettuare una raccolta dati e una visualizzazione rapida di suggerimenti e trend, mentre per una consultazione ed un'analisi più approfondita (svolta ragionevolmente da un medico) si preferisce utilizzare un applicativo web.

Prima di concludere il capitolo, un ultimo argomento importante da trattare riguarda le interazioni utente. Come precedentemente ribadito, sono state individuate due classi di utenti: medici e pazienti.

Mentre il medico interagirà esclusivamente con l'applicazione attraverso il suo personal computer, il paziente potrà farlo sia da mobile che da web. Inoltre, la procedura di raccolta dati prevede che quest'ultimo utilizzi un dispositivo wearable, in particolare un activity tracker, per raccogliere i dati su movimento, sonno e parzialmente sulla salute. Tali dati verranno poi gestiti dall'applicazione e articolati nelle quattro macro-sezioni viste in precedenza (Alimentazione, Movimento, Salute, Sonno).

2. Manutenzione di un applicativo server esistente

In questa sezione ed in quelle successive verrà illustrato esclusivamente il lavoro da me svolto; tuttavia, in alcuni casi saranno necessarie delle brevi divagazioni in modo tale da non far perdere il quadro d'insieme del progetto.

Prima di illustrare le funzionalità aggiunte al backend dell'applicativo, è bene fare il punto su quelle che sono le funzionalità già presenti. Si parlerà principalmente di API REST, discutendo la struttura che è presente dietro il loro funzionamento.

Il framework Spring, utilizzato all'interno di HealthApp, suggerisce l'utilizzo di diversi componenti in cui ciascuno di essi ha una diversa responsabilità. Un componente viene dichiarato nel framework utilizzando l'annotazione, ossia una "@" seguita dal tipo di componente in oggetto. L'annotazione base è @Component, dietro la quale esistono differenti specializzazioni, tra le quali: Controller, Service e Repository [3].

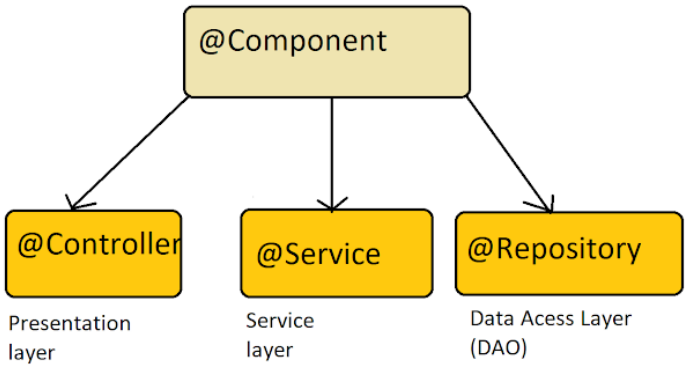


Figura 3 – Specializzazioni Componente

Questi tre componenti permettono di definire una struttura gerarchica sulla quale si basa l'endpoint REST offerto al frontend [4].

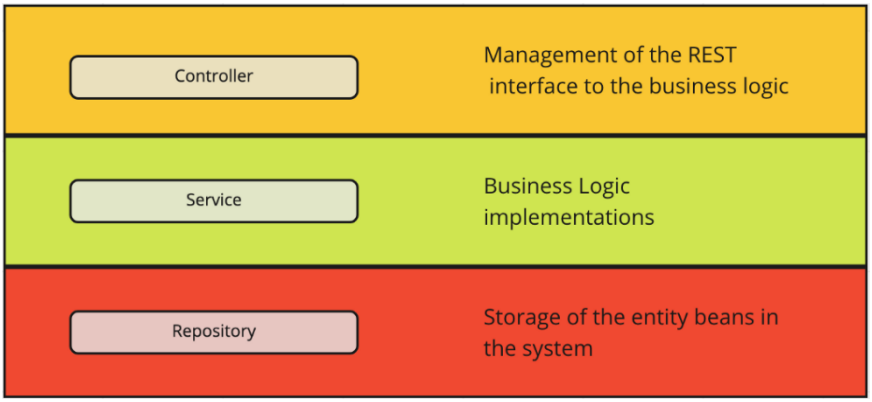


Figura 4 - Gerarchia Endpoint REST

- Il Repository si trova alla base della gerarchia. È il livello di interfacciamento con il database; si occupa della memorizzazione e manipolazione delle entità all'interno di quest'ultimo attraverso l'esecuzione di query predefinite o definite dal programmatore.

- Il Service è il cuore della gerarchia. In questo livello si sviluppa la logica del sistema; i dati vengono trasformati, aggregati, validati per prepararli alla presentazione o alla memorizzazione attraverso il repository.

- Il Controller rappresenta la vetta della gerarchia. A tale livello si definiscono i percorsi (path) che identificano gli endpoints raggiungibili dal client e si valida la ricezione della richiesta ricevuta dal client, verificando che l'utente sia in possesso dei permessi necessari per manipolare le informazioni richieste.

Il backend, nella versione preliminare, permette di creare e definire diversi ruoli, espone gli endpoint per la gestione degli utenti (Pazienti e Dottori) e dei questionari. Questi ultimi rappresentano il metodo per la registrazione di informazioni su alimentazione, movimento, sonno e salute. Nella versione trattata in questa tesi vengono integrate con le altre metodologie di raccolta dati che verranno successivamente descritte. L'insieme di endpoint disponibili è consultabile attraverso il percorso *http(s)://base_url/swagger-ui/index.html* in cui *base_url* rappresenta l'indirizzo del server HealthApp. La lista di API viene aggiornata automaticamente man mano che ne vengono definite di nuove.

Prima di procedere con l'estensione di nuove funzionalità, è importante apportare alcune modifiche al backend già esistente al fine di incrementare la sicurezza e la manutenibilità dello stesso.

2.1 Sicurezza e Validazione

Il primo passo consiste nella validazione degli input inviati mediante le API esistenti dai client web e mobile. Essa va oltre i controlli automatizzati eseguiti dal framework sul tipo e sulla presenza del valore.

I più significativi meccanismi di validazione riguardano:

- Data di nascita durante il processo di registrazione del paziente: è importante che il paziente sia maggiorenne all'atto della creazione dell'account.

- E-mail: tutte le e-mail inviate in input sono validate mediante regular expression per verificarne la correttezza sintattica.

- Intervallo di date: si verifica che la data di inizio intervallo sia minore o uguale della data di fine intervallo.

- Stringhe: si verifica che abbiano una lunghezza minima pari a uno e una lunghezza massima variabile a seconda della natura del valore.

Per quanto riguarda la sicurezza del server, si predispone un sistema che permetta di filtrare le richieste effettuate in base ai permessi posseduti dai client. Oltre al fatto che ogni richiesta (login e reset password con token ricevuto via e-mail esclusi) deve essere autenticata, è importante stabilire quale tipologia di utente debba aver accesso a quale tipologia di risorse. È necessario quindi implementare un sistema di controllo degli accessi.

Per tale motivo vengono definiti tre metodi di pre-autorizzazione:

- **hasIdAndIsPatient (ID: Long) : Boolean** – Tale metodo verifica se l'utente autenticato che ha effettuato la richiesta al server è un paziente ed ha come ID lo stesso delle risorse a cui chiede di accedere (in lettura e/o scrittura). In questo modo il paziente con ID uguale ad 1 potrà accedere solamente ai dati del paziente con ID uguale a 1 e non a quello degli altri pazienti. A seconda dell'esito della verifica, il metodo ritorna un booleano. Inoltre, se la verifica sull'ID non va a buon fine ma l'utente autenticato ha i privilegi di medico o amministratore il metodo ritorna comunque true, permettendo l'accesso ai dati ad account con privilegi maggiori di quello di tipo paziente.

- **isAdminOrSameUsername (userName: String): Boolean** – Tale metodo è utilizzato nelle richieste di accesso ai dati di un utente. Permette ad un utente con un dato username di accedere solamente ai propri dati mentre permette ad un utente con privilegi di amministratore di accedere ai dati di tutti gli utenti. Ritorna anch'esso un booleano a seconda che la verifica sui privilegi vada a buon fine o meno.

- **isAdminOrDoctor(): Boolean** – Tale metodo verifica se l'utente autenticato che richiede di accedere alle risorse presenta i privilegi di tipo amministratore o medico e ritorna un booleano a seconda dell'esito della verifica. Viene principalmente impiegato nel contesto delle recommendations per permettere solamente ai medici (e agli amministratori) di modificare i parametri di analisi personalizzabili per ogni paziente.

2.2 Documentazione e Schemi

Al fine di incrementare la manutenibilità del codice si è rivelato fondamentale creare la documentazione minima e i modelli semplificativi principali dell'applicazione, coerentemente con il modello di sviluppo agile.

Innanzitutto, utilizzando la notazione UML sono stati definiti diversi schemi e modelli che possono essere molto utili per comprendere la struttura e i meccanismi di HealthApp. Essi permettono di avere una visione completa del progetto e risultano utili al fine di localizzare con rapidità le sezioni in cui effettuare manutenzione o revisione del codice.

Si procederà all'illustrazione degli schemi, escludendo il diagramma architetturale (*deployment diagram*) già presentato nel capitolo precedente:

2.2.1 Diagramma Controllo Accessi

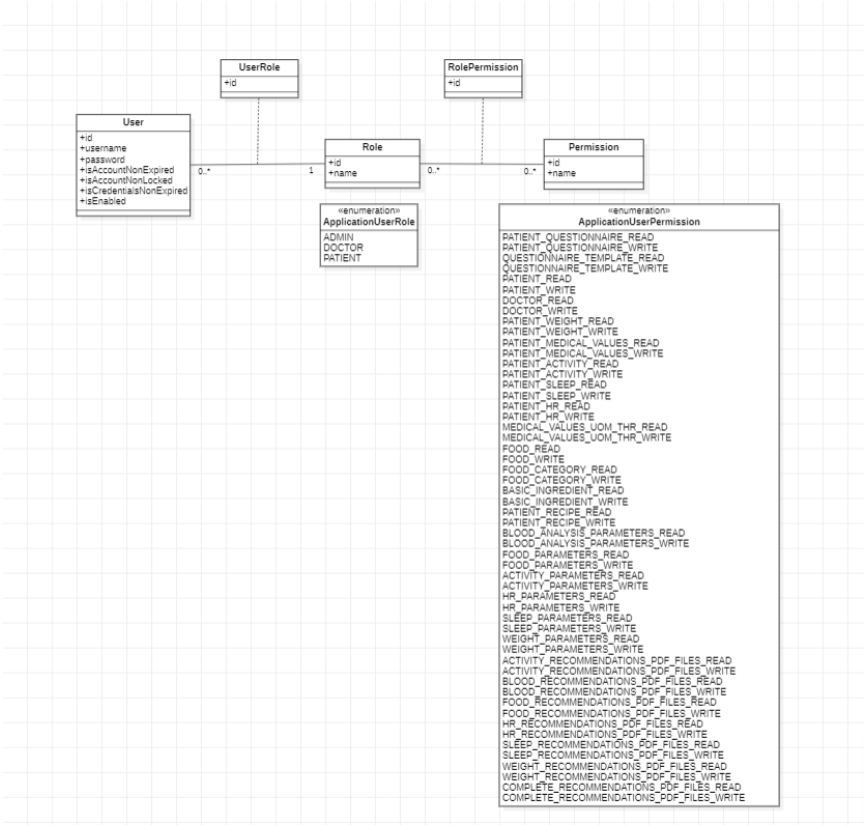


Figura 5 - Diagramma controllo accessi

Il diagramma illustra la definizione di ruoli e permessi che permettono il controllo degli accessi in HealthApp. Ogni utente, identificato da username e password, può assumere un solo ruolo al quale corrispondono dei particolari permessi.

Un permesso rappresenta la possibilità di accedere ad una determinata entità della base dati, in lettura oppure in scrittura.

Tra i ruoli definiti troviamo:

- **PATIENT:** Rappresenta il paziente, utilizzatore principale di HealthApp. I permessi di cui dispone riguardano l'accesso, in lettura e scrittura, ai propri dati relativi ad alimentazione, movimento, salute e sonno e l'accesso in sola lettura ai report delle recommendations.

- **DOCTOR:** Rappresenta il medico, il quale svolge un ruolo di supervisore sull'attività dei pazienti. I permessi di cui dispone sono gli stessi del paziente con la possibilità aggiuntiva di consultare i dati di ogni paziente, definire le soglie per le recommendations e creare dei nuovi account di tipo paziente.

- **ADMIN:** Rappresenta l'amministratore, utilizzatore eccezionale di HealthApp. Ha accesso in lettura e scrittura ad ogni struttura dati dell'applicazione, può manipolare qualsiasi informazione e/o utente.

2.2.2 Schema Database:

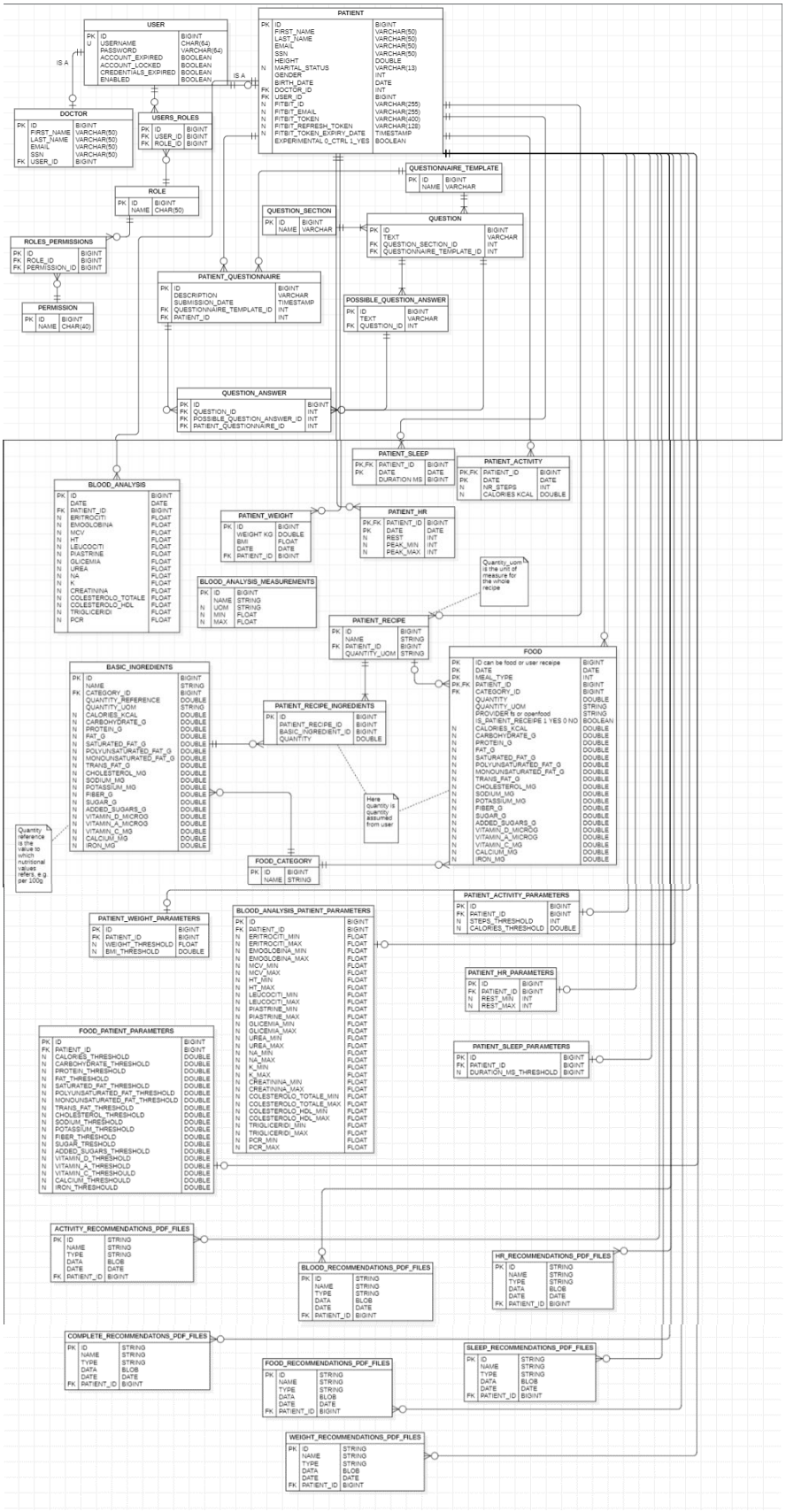


Figura 6 - Schema database

Lo schema del database risulta particolarmente utile per capire la struttura di ogni singola entità presente in HealthApp. Ad ogni entità è associata una tabella nella base dati.

Sono presenti le entità per la rappresentazione degli utenti (*USER)*, dei medici *(DOCTOR)*, dei pazienti *(PATIENT)* , dei ruoli *(ROLE)* e dei permessi *(PERMISSION E ROLES_PERMISSIONS).* Contengono gli attributi necessari a identificare l'utente, la tipologia, i permessi e ruoli associati nonché le credenziali di accesso.

Per la gestione dei questionari vengono invece definite le tabelle *QUESTIONNAIRE_TEMPLATE, QUESTION_SECTION, QUESTION, e POSSIBLE_QUESTION_ANSWER*, contenenti le informazioni sui questionari definiti, sulle loro sezioni, domande e opzioni per ogni domanda. Inoltre, le tabelle *PATIENT_QUESTIONNAIRE e QUESTION_ANSWER* permettono di registrare le risposte di ogni assistito.

Vi è una porzione di database dedita alla memorizzazione delle informazioni relative alle quattro macroaree.

Per l'alimentazione sono definite le entità *FOOD, FOOD_CATEGORY*, *PATIENT_RECIPE, PATIENT_RECIPE_INGREDIENTS E BASIC_INGREDIENTS*. Oltre alla possibilità di avere il proprio diario alimentare, con valori nutrizionali associati, il paziente può definire le proprie ricette utilizzando una base dati di ingredienti definita direttamente in HealthApp dell'ordine di grandezza di mille record. Tali ricette, definite una volta sola, possono essere richiamate infinite volte nel diario alimentare.

Per quanto riguarda il movimento la tabella *PATIENT_ACTIVITY* permette di tener traccia delle calorie consumate e dei passi effettuati da ogni assistito.

Relativamente alla salute, *PATIENT_HR* conterrà le informazioni relative alla frequenza cardiaca, mentre *BLOOD_ANALYSIS* e *BLOOD_ANALYSIS_MEASUREMENTS* conterranno rispettivamente i dati relativi alle analisi mediche, alle unità di misura e ai valori nominali.

Il sonno è implementato mediante la tabella *PATIENT_SLEEP*, che conterrà i dati relativi al tempo di sonno registrato.

Tipicamente ogni record associato ad alimentazione, movimento, salute o sonno presenta una data e il paziente di riferimento per permettere il tracciamento e conservare il trend temporale delle informazioni e poterne analizzare i progressi.

Vi sono successivamente le entità rappresentanti i parametri personalizzabili dai medici per paziente su cui effettuare l'analisi e produrre le recommendations, ossia: *PATIENY_ACTIVITY_PARAMETERS, PATIENT_HR_PARAMETERS, PATIENT_WEIGHT_PARAMETERS, FOOD_PATIENT_PARAMETERS, BLOOD_ANALYSIS_PATIENT_PARAMETERS e PATIENT_SLEEP_PARAMETERS*.

Infine, le tabelle *COMPLETE_RECOMMENDATIONS_PDF_FILES, ACTIVITY_RECOMMENDATIONS_PDF_FILES, BLOOD_RECOMMENDATIONS_PDF_FILES, WEIGHT_RECOMMENDATIONS_PDF_FILES, HR_RECOMMENDATIONS_PDF_FILES, FOOD_RECOMMENDATIONS_PDF_FILES e SLEEP_RECOMMENDATIONS_PDF_FILES* consentono di memorizzare i report delle recommendations prodotti in automatico con cadenza settimanale.

I paragrafi del capitolo successivo illustreranno l'estensione delle funzionalità del backend di HealthApp.

3. Espansione di un applicativo server esistente

Al fine di implementare le quattro macroaree supportate da HealthApp è necessario pensare a delle API che permettano di poter manipolare facilmente i dati prodotti dai pazienti. Per tale ragione, è importante individuare i parametri chiave di cui si vuole tener traccia, che compongono ogni macroarea. Inoltre, al fine di mantenere una separazione tra queste ultime, si sceglie di suddividere le API in gruppi, ciascuno totalmente indipendente dall'altro e corrispondente ad un mapping logico con le tabelle precedentemente illustrate nello schema del database.

I gruppi individuati sono:

- **Peso** – Macroarea salute
- **Analisi mediche** – Macroarea salute
- **Dati generati tramite Fitbit** – Suddiviso nei sottogruppi attività, sonno e frequenza cardiaca (hr) rispettivamente appartenenti alle macroaree movimento, sonno e salute
- **Cibo e Ricette** – Macroarea alimentazione

Si illustrano le varie implementazioni:

3.1 Peso

Il caso d'uso correlato a tale gruppo è quello del paziente che periodicamente rileva la sua massa attraverso strumentazione esterna ad HealthApp (ad esempio una bilancia) e lo registra all'interno dell'applicazione. Il sistema provvederà ad etichettare il valore ricevuto aggiungendovi una data e calcolandone l'indice di massa corporea associato.

Quindi, oltre a massa e data troviamo l'indice di massa corporea (IMC o BMI). Esso rappresenta un dato biometrico nonché l'indicatore di riferimento per la valutazione dello stato di peso forma, anche se non permette di conoscere la reale composizione corporea o la distribuzione del grasso nell'individuo. Può essere quindi visto semplicemente come un primo importante strumento di analisi [5].

L'IMC si calcola con una formula matematica a partire dalla massa e dall'altezza dell'individuo. In HealthApp l'altezza viene memorizzata come un valore associato al paziente ed immutabile nel tempo. Tale assunzione nasce dal fatto che almeno durante la prima fase sperimentale dell'applicativo, l'età dei soggetti coinvolti supera di gran lunga l'età dello sviluppo umano in cui l'altezza subisce mutazioni considerevoli e di conseguenza la si può considerare relativamente stabile nel tempo.

La formula che permette quindi di ottenere l'indice di massa corporea è la seguente [6]:

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

Equazione 1 - Formula BMI

Bisogna tuttavia considerare che l'applicazione tiene traccia della statura in centimetri, è necessario quindi applicare la seguente formula [7]:

$$BMI = \frac{Massa\ (kg)}{[Altezza(cm)]^2} \cdot 10^4$$

Equazione 2 - Formula BMI in cm

Il calcolo del BMI viene effettuato automaticamente dal backend dell'applicazione nel momento in cui il paziente decide di registrare la propria massa attraverso il frontend mobile di HealthApp applicando la formula precedentemente illustrata.

3.1.1 Implementazione API

Le API implementate in questo gruppo seguono il paradigma CRUD (Create, Read, Update, Delete) e utilizzano i metodi http standard appropriati (così come tutte le altre API in HealthApp) per effettuare questo tipo di operazioni (rispettivamente POST, GET, PUT, DELETE).

In particolare, le API definite e messe a disposizione dei client sono:

- GET /api/patients/{patientID}/weights : ritorna un JSON contenente una lista di oggetti peso relativi al paziente identificato dall' ID particolare passato come parametro della chiamata. Ogni oggetto conterrà l'identificativo del particolare record, la massa espressa in kg ed associata al tipo float, il bmi sempre di tipo float e la data in cui il paziente ha registrato tali valori.

- POST /api/patients/{patientID}/weights : riceve un JSON contenente il solo valore della massa espressa in kg e di tipo float. Provvede a salvare tale valore nel database, associandolo all'account del paziente che ha effettuato la chiamata, insieme al valore dell'IMC calcolato e alla data corrente in cui la chiamata all'API è avvenuta.

- PUT /api/patients/{patientID}/weights/{idW}?date={date} : riceve un JSON contenente il solo valore della massa espressa in kg e di tipo float. Provvede ad aggiornare il valore della massa e a calcolare il nuovo IMC associato salvando entrambi nel database nel record avente come identificativo quello passato tramite la variabile idW e associato al paziente identificato dal contenuto della variabile patientID. La data non viene modificata e rimane quella registrata all'atto della creazione.

- DELETE /api/patients/{patientID}/weights/{idW} : elimina il record identificato dal contenuto della variabile idW relativo al paziente identificato dal contenuto della variabile patientID.

3.1.2 Osservazioni

Grazie a queste API e con il supporto dei client il paziente può tener traccia dell'andamento temporale della sua massa e del suo indice di massa corporea che verranno presentati sotto forma di grafico. Anche il medico potrà valutare questi dati, optando per la scelta di comportamenti correttivi da suggerire al suo assistito.

3.2 Analisi mediche

Il caso d'uso delle analisi mediche è molto simile a quello del peso ma non del tutto analogo. Il primo step decisionale riguarda chi, tra paziente e medico, è incaricato di registrare il risultato delle analisi mediche. Per garantire maggiore flessibilità, si sceglie di far registrare al paziente il risultato delle analisi mediche. In questo modo non si hanno problemi qualora egli volesse eseguirle in un presidio differente da quello del medico o rivolgersi ad uno specialista differente dal medico che lo segue nell'applicazione per la lettura e l'interpretazione dei risultati.

Il meccanismo di raccolta dati anche in questo caso non è automatico, bensì manuale. Sarà infatti l'utente a dover "digitalizzare" le analisi, sfruttando dei campi predefiniti in cui è necessario semplicemente comunicare il valore del parametro.

La differenza sostanziale rispetto all'approccio seguito in precedenza consiste nell'associare l'informazione temporale ai dati da memorizzare. Se nel caso del peso ciò avveniva automaticamente, adesso è il paziente che deve comunicare la data in cui le analisi sono state effettuate.

Tale scelta implementativa trova riscontro nel fatto che la misurazione della massa corporea è un procedimento piuttosto rapido da effettuare ed è svolto in prima persona dal paziente stesso che è quindi interessato a registrare nell'immediato il risultato all'interno dell'applicazione. Nel caso delle analisi mediche invece, il paziente non è l'unico attore nel procedimento di misura ma interviene anche un laboratorio di analisi. Siccome la data in cui i campioni vengono prelevati, quella in cui vengono analizzati e prodotti i risultati e quella in cui questi vengono consultati dal paziente differiscono nella stragrande maggioranza dei casi, ne conviene che sia preferibile far inserire all'utente la data da associare alle informazioni da memorizzare.

Inoltre, un altro punto a supporto della scelta è che il paziente difficilmente tende a ricordare la massa che aveva mesi o anni fa (a meno che non ne tenga traccia scritta), mentre avrà con una probabilità superiore traccia di analisi mediche effettuate anche prima di iniziare ad utilizzare HealthApp e potrebbe volerle quindi registrare.

Al di fuori del discorso temporale, è importante individuare quelli che sono i valori d'interesse da memorizzare nella base dati dell'applicazione. Essendo quelle ingegneristiche delle conoscenze totalmente

estranee al settore, si è preferito richiedere questa valutazione ai medici stessi. Tramite uno scambio di e-mail si è arrivati quindi a stabilire che i dati d'interesse fossero:

Tabella 1 – Valori d'interesse analisi mediche

Nome parametro	Unità di misura	Valore convenzionale minimo	Valore convenzionale massimo
Eritrociti	$\frac{10^{12}}{l}$	4,00	5,20
Emoglobina	$\frac{g}{dl}$	12	14
MCV (Volume corpuscolare medio)	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

Oltre ai parametri e alle unità di misura è stato molto importante conoscere i valori convenzionali, laddove presenti. Infatti, essi saranno di rilevante importanza nel sistema delle recommendations in quanto permetteranno, attraverso un'analisi automatizzata, di riconoscere e segnalare rapidamente eventuali valori al di fuori dei range standard.

Ai fini della memorizzazione all'interno della base dati, le unità di misura dei parametri medici ed i valori convenzionali sono considerati immutabili nel tempo (o meglio con rare possibilità di cambiamento) mentre i valori rilevati variano di analisi in analisi ed ovviamente differiscono da paziente a paziente.

È necessario scegliere quindi tra due approcci:

- Il primo consiste nel creare una tabella unica in cui si memorizzano valori, unità di misura, valori convenzionali e la data. Questa soluzione favorisce dei tempi di risposta migliori in fase di lettura dati, in quanto non è necessario eseguire dei Join tra le tabelle ma comporta una ridondanza elevata. Infatti, per ogni record memorizzato, il contenuto dei campi unità di misura e valori convenzionali sarà sempre lo stesso con una conseguente occupazione inutile di spazio.

- Il secondo approccio consiste nel creare due tabelle, una in cui si memorizzano il risultato delle analisi associato alla data e un'altra in cui si memorizzano le unità di misura e i valori convenzionali. In questo modo non si ha una occupazione inutile di spazio in quanto i valori indipendenti dal fattore tempo vengono memorizzati una volta sola. Risulta più elevato il tempo di lettura, in quanto coinvolge due tabelle anziché una singola.

Si predilige la seconda soluzione in quanto l'overhead temporale risulta accettabile rispetto all'occupazione di memoria. Conseguenza di tale scelta è che si preferisce evitare di referenziare le due tabelle e di eseguire dei Join, tenendole distinte. Sarà compito del frontend presentare i valori con le opportune unità di misura, effettuandone la lettura dal server. Le letture dati saranno separate ed anche le API corrispondenti da chiamare saranno separate.

3.2.1 Implementazione API

Per quanto riguarda le operazioni che si sceglie di supportare, per le analisi mediche vi è il classico paradigma CRUD con le quattro api GET, POST, PUT, DELETE. In riferimento alle unità di misura e ai valori convenzionali è importante osservare che essi derivano appunto da convenzioni universali stabilite. Tali convenzioni non sono soggette a variazioni importanti e per tale motivo, come ribadito in precedenza, si possono considerare le informazioni come immutabili e memorizzarle nella base dati direttamente dal codice del backend. Anziché tenerle salvate in constanti (hardcoded), si preferisce avere una routine che inizializzi il database in modo tale da poter intervenire su di esso anziché sul sorgente del server (con conseguente perdita temporanea di disponibilità) in caso vadano modificate.

Le API messe a disposizione dei client sono una GET per le operazioni di lettura e una PUT per quelle di modifica (ristretta solo ad utenti di tipo amministratore o medico).

Entrando nel dettaglio delle API relative alle analisi mediche si trova:

- GET /api/patients/{patientID}/bloodAnalysis : ritorna un JSON contenente una lista di oggetti analisi mediche relativi al paziente identificato dall' ID particolare passato come parametro della chiamata. Ogni oggetto conterrà l'identificativo del particolare record, la data a cui fanno riferimento le analisi e i valori di eritrociti, emoglobina, mcv, ht, leucociti, piastrine, glicemia, urea, na, k, creatinina, colesterolo totale, colesterolo HDL, trigliceridi e pcr tutti di tipo double.

- POST /api/patients/{patientID}/bloodAnalysis : riceve un oggetto JSON contenente un'istanza di analisi mediche. In particolare, insieme alla data di riferimento ci possono essere i valori di eritrociti, emoglobina, mcv, ht, leucociti, piastrine, glicemia, urea, na, k, creatinina, colesterolo totale, colesterolo HDL, trigliceridi e pcr tutti di tipo double. Provvede a salvare tale istanza nel database, associandolo all'account del paziente che ha effettuato la chiamata.

- PUT /api/patients/{patientID}/bloodAnalysis/{idMV} : riceve un oggetto JSON contenente la data i valori di eritrociti, emoglobina, mcv, ht, leucociti, piastrine, glicemia, urea, na, k, creatinina, colesterolo totale, colesterolo HDL, trigliceridi e pcr tutti di tipo double. Provvede ad aggiornare il record della base dati identificato dal contenuto della variabile idMV e relativo al paziente identificato mediante patientID. A differenza delle API del peso, è possibile aggiornare anche la data di riferimento delle analisi. Il motivo di questa scelta è quello di permettere all'utente di poter modificare la data in caso di errore dovuto a un'errata digitazione o ad un clic involontario.

- DELETE /api/patients/{patientID}/bloodAnalysis/{idMV} : Elimina dalla base dati l'istanza di analisi mediche identificata mediante il contenuto della variabile idMV e relativa al paziente identificato mediante il contenuto della variabile-parametro patientID.

È importante sottolineare che i valori di ogni singolo attributo riferiti ad un'istanza di analisi mediche possono assumere anche il valore null (tranne identificativo e data ovviamente). La scelta è motivata dal fatto che è possibile effettuare delle analisi parziali (ad esempio senza misurare il valore dell'urea piuttosto che del pcr e così via) e quindi avere dei dati non disponibili in una particolare data. Inoltre, il database dell'applicazione segue un modello relazionale. Scegliendo un'altra tipologia di modello (es basato sui documenti) si potrebbe evitare di avere dei campi null con una memorizzazione più efficiente. Tuttavia, per mantenere una continuità con il modello di database già presente, si è preferito estendere quello attuale piuttosto che crearne uno ex-novo.

Entrando invece nel dettaglio delle API relative alle unità di misura e valori convenzionali delle analisi mediche si trova:

- GET /api/bloodAnalysis/measurements : ritorna un JSON contenente una lista di oggetti. Ciascuno di essi è rappresentato da un identificativo di tipo intero, dal nome del parametro delle analisi mediche (es. emoglobina, colesterolo totale) di tipo stringa, dall'unità di misura sempre di tipo stringa, dal valore convenzionale minimo di tipo double e dal valore convenzionale massimo sempre di tipo double.

- PUT /api/bloodAnalysis/measurements/{idMVUomThrs} : riceve un JSON contenente un nome ed un'unità di misura entrambi di tipo stringa insieme ai valori convenzionali minimo e massimo di tipo double. Provvede ad aggiornare con le nuove informazioni il record della base dati identificato mediante il contenuto della variabile idMVUomThrs passata come parametro. Può

essere chiamata esclusivamente da un utente di tipo amministratore o medico. Il suo utilizzo all'interno dell'applicazione dovrebbe essere sporadico se non assente del tutto. La necessità di avere tale API nasce dalla possibilità che i valori convenzionali possano cambiare nel corso degli anni.

Come per le analisi mediche, anche le unità di misura e i valori convenzionali possono assumere il valore null. Infatti, è possibile avere delle grandezze assolute o dei valori medici per i quali non è presente il valore convenzionale minimo, piuttosto che il massimo. È invece altamente improbabile l'assenza sia di un valore minimo che di un massimo.

3.2.2 Osservazioni

Per mezzo delle API illustrate il paziente può registrare le proprie analisi mediche nell'applicazione, digitalizzandole se non lo sono già ed evitando di perderle come può accadere con dei documenti cartacei. Il medico, inoltre, può effettuare la lettura e le valutazioni semplicemente aprendo il client web di HealthApp, evitando quindi la necessità di dover ricevere il paziente su appuntamento. In generale, è possibile che i risultati siano già stati digitalizzati dai software proprietari dei laboratori in cui i campioni sono analizzati e resi disponibili al medico. Tuttavia, se il laboratorio appartiene ad un altro plesso ospedaliero o è privato, è difficile che il medico possa consultarli con facilità. Tale soluzione risulta quindi essere abbastanza vantaggiosa, e lo è ancora di più combinata al meccanismo delle recommendations, in quanto si fornisce immediatamente ad un paziente, competente in materia o meno che sia, un primo responso (non ovviamente paragonabile a quello medico) sui risultati delle analisi mediche effettuate.

3.3 Dati generati tramite Fitbit

A differenza degli endpoint già illustrati, il gruppo di API oggetto del paragrafo è quello relativo ai dati generati tramite Fitbit. Esso coinvolge ben tre macroaree (movimento, salute e sonno).

L'elemento fondamentale che permette la raccolta di tali informazioni è il tracker/wearable Fitbit che richiede di essere indossato quotidianamente dai pazienti affinché vi sia un'effettiva produzione di dati. Il caso d'uso ricade infatti nel paziente che produce dati sul proprio battito cardiaco piuttosto che sui passi, sulle calorie consumate o sul sonno. Questi vengono immagazzinati nei server proprietari Fitbit e recuperati dal backend di HealthApp. Il frontend li metterà quindi a disposizione di pazienti e medici, con diversi stili di presentazione e differenti granularità. Inoltre, anche in questo caso, il sistema integrato delle recommendations servirà a fornire un primo responso di analisi su base giornaliera, settimanale o mensile.

Appurato che le informazioni vengono trasmesse dal wearable ai server Fitbit mediante l'uso dell'app proprietaria omonima, è necessario recuperare questi dati e depositarli nella base dati di HealthApp. Questo meccanismo di interazione riguarderà esclusivamente il backend dell'applicazione, rendendo la procedura totalmente trasparente ai client.

Per fare ciò, occorre innanzitutto stabilire un meccanismo di connessione tra i server. Fitbit mette a disposizione degli sviluppatori un insieme di Web API per la lettura delle informazioni prodotte dai dispositivi wearable. Per poter accedere a tali risorse è necessario registrare la propria applicazione sulla console di sviluppatori Fitbit. [8]

3.3.1 Integrazione Fitbit con HealthApp

Il processo di registrazione consiste nella compilazione di un form in cui si indicano i riferimenti principali dell'applicazione che si intende sviluppare, si fornisce una breve descrizione delle più importanti funzionalità supportate e si richiede un accesso ai dati in sola lettura o in lettura e scrittura. Inoltre, la scelta cruciale in questa fase riguarda la tipologia di applicazione che si vuole registrare. Infatti, è possibile scegliere tra tre differenti tipologie [9]:

- **Server**: Un applicativo del genere è dotato di un'architettura a più livelli e presenta un server. L'autenticazione e la gestione degli utenti e dei propri dati avviene lato server. Il server rende consultabile ad ogni utente i propri dati.

- **Client**: Un applicativo del genere è dotato di un'architettura a livello singolo. L'autenticazione e la gestione degli utenti e dei propri dati avviene lato client. Il client rende consultabile ad ogni utente i propri dati.

- **Personale**: Un applicativo di tipo client o server in cui i dati accessibili sono solo quelli dello sviluppatore che si registra alla console e che registra la propria app e che non supporta altri utenti.

Nel caso di HealthApp la scelta ricade su un applicativo di tipo server con accesso in lettura. Una volta compilato il form la registrazione è completata. È necessario che questa venga approvata dal team di Fitbit. Fatto ciò, si dispone finalmente di tutte i parametri necessari per utilizzare le API messe a disposizione degli sviluppatori. Tali parametri sono:

- *OAuth 2.0 Client ID*: è l'identificativo univoco dell'applicazione da utilizzare per l'autenticazione e per le chiamate alle API sui server Fitbit. Il nome "client" può generare confusione ma si riferisce al fatto che HealthApp è client (in fase di lettura dati) del server Fitbit.

- *Client Secret*: Affinché il backend di HealthApp possa autenticarsi sui server Fitbit, deve essere in possesso di un segreto e lo deve conservare in maniera sicura.

- *Redirect URL*: Indica il percorso che verrà seguito quando un utente di HealthApp tenta di connettersi con il proprio account Fitbit. Come verrà illustrato in seguito, infatti, è necessario che egli venga reindirizzato da HealthApp a Fitbit per eseguire l'accesso con le proprie credenziali e fornire tutte le autorizzazioni necessarie all'applicazione per funzionare. Una volta fornite le autorizzazioni, l'utente verrà nuovamente reindirizzato verso il Redirect URL specificato (che in questo caso sarà un URL di Health App Web).

- *OAuth 2.0: Authorization URI*: API che il client dovrà chiamare per avviare la procedura di collegamento dell'account Fitbit ad HealthApp. Tale API ha come percorso: https://www.fitbit.com/oauth2/authorize

- *OAuth 2.0: Access/Refresh Token Request URI*: API che il server dovrà chiamare per avviare la procedura di richiesta o rinnovo token per permettere l'accesso ai dati Fitbit dell'utente. Tale API ha come percorso: https://api.fitbit.com/oauth2/token

Prima di recuperare i dati prodotti tramite il tracker, è necessario che il paziente fornisca l'autorizzazione ad HealthApp per accedere ai propri dati generati. Tale autorizzazione non può essere fornita salvando nella base dati dell'applicazione l'e-mail e la password dell'account Fitbit del paziente, in quanto ciò rappresenterebbe un modo poco sicuro di agire dal punto di vista della sicurezza informatica e della protezione dei dati personali. Infatti, l'utente potrebbe voler condividere solamente le informazioni necessarie al funzionamento dell'applicazione, senza permettere a nessun altro di avere pieno accesso al proprio account Fitbit e ai dati sensibili contenuti all'interno. Inoltre, in caso di compromissione del database di HealthApp, si metterebbero a rischio tutti gli account Fitbit collegati, esponendo a malintenzionati le credenziali di accesso.

Per questo motivo è disponibile una procedura di autenticazione che coinvolge gli utenti, il frontend, il backend di HealthApp e i server Fitbit basata sul protocollo *OAuth 2.0*.
Il protocollo di autenticazione *OAuth 2.0* può essere descritto ad alto livello come segue [10]:

1. L'applicazione invia all' *Authorization Server* una richiesta di autorizzazione per accedere ad una risorsa protetta.

2. Il proprietario della risorsa (l'utente) concede l'accesso.

3. L'*Authorization Server* restituisce un *Access Token* da utilizzare in tutte le successive richieste come una sorta di "cartellino di riconoscimento". Esso avrà durata limitata e andrà rinfrescato periodicamente utilizzando un altro token, il *Refresh Token*.

L'autenticazione utilizzata da Fitbit si configura come una di tipo *Client Credentials Grant Authentication*, basata ossia sull'utilizzo del client ID e del client Secret ottenuti in fase di registrazione. La figura seguente riassume gli scambi tra il server HealthApp e quello Fitbit che portano alla generazione dell'*access token* per ogni utente:

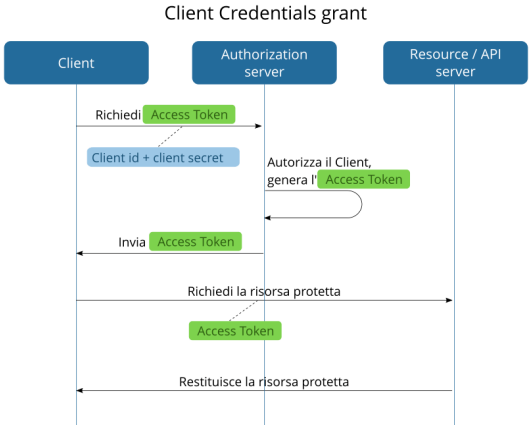


Figura 7 - Step OAuth 2.0 tra i server (intra-server)

Entrando maggiormente nel dettaglio:

1. Il paziente appena registrato su HealthApp si trova per la prima volta a dover effettuare il login.

2. All'atto del primo login, il client web HealthApp effettua la chiamata alla *OAuth 2.0: Authorization URI* passando come parametri il *client ID* in chiaro ed il *client Secret* cifrato mediante una sfida crittografica.

3. Il *Fitbit Authorization Server* riconosce il client che ha originato la richiesta (in questo caso HealthApp) ed offre al paziente la possibilità di accedere con il proprio account Fitbit.

4. Una volta effettuato l'accesso, l'utente viene invitato a fornire le autorizzazioni necessarie all'accesso ai propri dati generati mediante il wearable come mostrato dalla figura seguente.

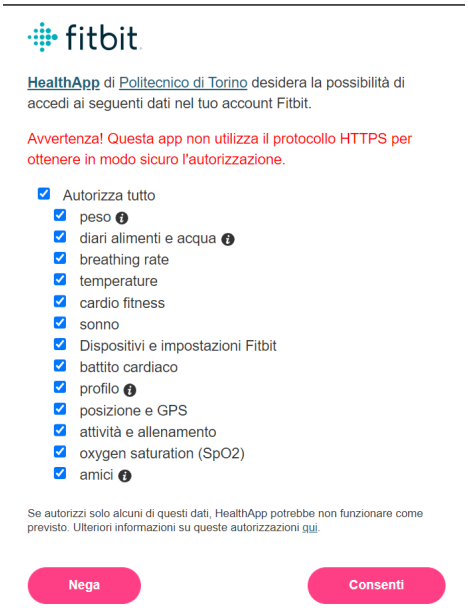


Figura 8 - Form di autorizzazione Fitbit

Alcuni consensi, come quello della lista di amici vengono attualmente richiesti in vista di future espansioni dell'applicazione.

5. Ottenuto il consenso, il *Fitbit Authorization Server* genera un codice usa e getta e chiama l'API di HealthApp contenuta nel *Redirect URL* passando tale codice come parametro.

6. Il codice ricevuto viene utilizzato dal backend HealthApp per effettuare la chiamata all' *OAuth 2.0: Access/Refresh Token Request URI* e richiedere la generazione del token.

7. Il *Fitbit Authorization Server* genera un *access token* e un *refresh token*. Il primo ha una durata fissa di otto ore e permette l'accesso ai dati del paziente. Passate le otto ore, è necessario rinnovarlo. Per fare ciò si utilizza il *refresh token*. Esso può essere utilizzato una volta sola e consente di ottenere un nuovo *access token*. *Access* e *refresh token* vengono inviati come risposta al chiamante dell' API relativa all' *OAuth 2.0: Access/Refresh Token Request URI*.

8. Il Server di HealthApp riceve e salva *access* e *refresh token*, associandoli all'account del paziente e potendo così finalmente accedere ai dati.

La procedura appena illustrata viene effettuata solamente al primo login del paziente su HealthApp. Per tutti i successivi, il server verifica che il token non sia scaduto e quindi ancora valido. Qualora fosse scaduto viene avviata un'altra procedura:

1. Il Server di HealthApp recupera il refresh token del paziente salvato e lo invia al Fitbit Authorization Server mediante chiamata all' *OAuth 2.0: Access/Refresh Token Request URI*.

2. Il *Fitbit Authorization Server* genera una nuova coppia di *Access* e *Refresh Token* e li invia in risposta al Server di HealthApp.

3. Il Server di HealthApp salva i nuovi *access* e *refresh token*.

La procedura di *refresh* è molto più snella e rapida rispetto al primo ottenimento del token. Essa viene ripetuta, oltre che all'atto del login per il singolo paziente se il token è scaduto, anche con una routine giornaliera periodica che verifica e aggiorna eventuali token scaduti per tutti i pazienti. Qualora, all'atto del login, vi siano problemi con il *refresh* del token esistente, si ripeterà l'ottenimento da zero di un nuovo *access token,* con una procedura equivalente a quella effettuata al primo login.

Poiché, per come è concepita l'architettura attuale, un account di tipo paziente può essere creato solamente da un account di tipo medico o amministratore (non esiste il *self sign-up*), verosimilmente l'intera procedura di prima configurazione dell'account paziente verrà eseguita in presenza del medico che dovrà essere istruito sui passaggi da seguire. A tal proposito, è stata creata anche una guida passo-passo in formato PDF facilmente consultabile e consultabile alla voce *Allegato 1. Guida registrazione nuovo paziente su HealthApp*.

3.3.2 Implementazione API

Chiarito il meccanismo con il quale è possibile collegare gli account Fitbit ad HealthApp e ricevere le autorizzazioni necessarie a consultare i dati generati dai pazienti, è necessario capire come questi vengano recuperati. Come ribadito in precedenza, Fitbit mette a disposizione una serie di API REST che permettono di leggere le informazioni d'interesse per un determinato paziente passando come parametro di autenticazione l'*Access Token* associato.

Tra tutti quelli disponibili, gli endpoint di interesse in questo progetto sono:

- GET https://api.fitbit.com/1/user/[user-id]/activities/steps/date/[start-date]/[end-date].json – Ritorna un oggetto JSON contenente un vettore per l'intervallo di date selezionato e per l'account corrispondente all'access token. A sua volta il vettore conterrà un insieme di oggetti; ciascuno descrive una data e il conteggio dei passi.

- GET https://api.fitbit.com/1/user/[user-id]/activities/heart/date/[start-date]/[end-date].json – Ritorna un oggetto JSON contenente un vettore per l'intervallo di date selezionato e per l'account corrispondente all'access token. A sua volta il vettore conterrà un insieme di oggetti; ciascuno descrive una data, le differenti zone cardiache (es picco, cardio, fat-burn ecc.), il consumo calorico

dovuto al movimento e il battito cardiaco a riposo, se disponibile (in alcuni casi in cui non viene rilevato non viene ritornato).

- GET https://api.fitbit.com/1.2/user/[user-id]/sleep/date/[startDate]/[endDate].json - Ritorna un oggetto JSON contenente un vettore per l'intervallo di date selezionato e per l'account corrispondente all'access token. A sua volta il vettore conterrà un insieme di oggetti; ciascuno descrive una data e la durata del tempo di sonno espressa in millisecondi.

Si noti come tutte le API ritornino le informazioni prodotte all'interno di un intervallo di date che può essere lungo al massimo cento giorni. Il backend di HealthApp utilizza sempre il range massimo in fase di lettura, passando come data massima la data in cui avviene la chiamata, e come data minima il massimo diminuito di cento.

Il server di HealthApp effettua la lettura dei dati per singolo paziente all'atto del login di quest'ultimo e presenta inoltra una routine periodica a cadenza giornaliera che effettua la lettura per tutti i pazienti aventi un *access token* valido. I dati letti vengono memorizzati internamente all'interno della base dati dell'applicazione. In particolare, tre tabelle differenti vengono utilizzate: una per l'attività (passi e calorie bruciate), una per il sonno e una per il battito cardiaco (viene memorizzato solo quello a riposo e quello di picco minimo e massimo tra le varie zone disponibili).

La scelta di memorizzare internamente le informazioni piuttosto che utilizzare il server dell'applicazione come tramite tra il client HealthApp e il Server Fitbit è motivata da diversi fattori:

- Fitbit offre la possibilità di leggere dati in un intervallo di cento giorni. È ragionevole che il paziente che utilizza l'applicazione voglia vedere tutti i suoi dati. Quindi, dopo oltre cento giorni di utilizzo, ad esempio mille, se si hanno i dati correttamente memorizzati, non è necessario effettuare dieci chiamate ai server Fitbit per recuperare tutte le informazioni, ma ne basterà una sola per leggere solo quelle più recenti. Inoltre, se l'utente dovesse non accedere per più di cento giorni ad HealthApp troverà comunque tutti i dati prodotti, in quanto la routine giornaliera del server provvede ad aggiornare le informazioni di tutti i pazienti. Non si hanno quindi problemi di perdita o mancanza di dati.

- Qualora in futuro si volesse estendere il supporto ad altri wearable/tracker è bene pensare di avere un'unica base dati di riferimento ed effettuare le letture dai vari server proprietari predisponendo degli adattatori per la memorizzazione dei dati in forma omogenea.

- Qualora i server Fitbit dovessero essere non disponibili, è possibile comunque accedere ai dati dei vari pazienti, anche se questi non conterranno, nel peggiore dei casi, le informazioni relative al periodo più recente.

A memorizzazione avvenuta, i client potranno servirsi degli endpoint esposti dal server HealthApp, tutti di tipo GET in quanto operazioni di lettura, per accedere ai dati. Piuttosto che creare un'unica API per recuperare tutte le informazioni generate mediante Fitbit, si è preferito effettuare una divisione in sottogruppi così composta:

  - Attività: Permette di recuperare separatamente le informazioni relative a passi e calorie bruciate.

- Frequenza Cardiaca: Permette di recuperare separatamente le informazioni relative al battito cardiaco a riposo e ai battiti di picco.

- Sonno: Permette di recuperare le informazioni sul tempo di sonno.

Iniziando dal sottogruppo attività:

- GET /api/patients/{patientID}/activities/steps?startDate={startDate}&endDate={endDate} – Ritorna un oggetto JSON contenente un vettore di oggetti passi per l'intervallo di date passato e relativi al paziente identificato mediante il contenuto della variabile patientID. Ogni oggetto passo contiene la data di riferimento e il conteggio dei passi effettuati in tale data.

- GET /api/patients/{patientID}/activities/calories?startDate={startDate}&endDate={endDate} – Ritorna un oggetto JSON, relativo all'intervallo di date passato e al paziente identificato da patientID, contenente un array di oggetti; ognuno di essi contiene la data di riferimento e il conteggio delle calorie bruciate dal paziente in tale data.

Per quanto riguarda la frequenza cardiaca:

- GET /api/patients/{patientID}/hrs/rest?startDate={startDate}&endDate={endDate} – Ritorna un oggetto JSON contenente un insieme di oggetti relativo al paziente identificato mediante patientID e all'intervallo di date passato. Ciascun oggetto contiene la data di riferimento e il valore del battito a riposto in tale data.

- GET /api/patients/{patientID}/hrs/peak?startDate={startDate}&endDate={endDate} – Ritorna un oggetto JSON contenente un vettore di oggetti relativo all'intervallo di date passato e al paziente identificato mediante il contenuto della variabile patientID. Ogni oggetto contiene la data di riferimento e i valori di picco minimo e picco massimo in tale data.

Relativamente al sonno:

- GET /api/patients/{patientID}/sleep/duration?startDate={startDate}&endDate={endDate} – Ritorna un oggetto JSON, relativo all'intervallo di date passato e al paziente identificato mediante patientID, contenente un array di oggetti; ciascuno è composto dalla data di riferimento e dal tempo di sonno espresso in millisecondi.

Un fattore comune a tutte le API è che qualora in una determinata data non fossero disponibili le informazioni richieste, non viene proprio ritornato l'oggetto anziché ritornare null. Il comportamento è coerente con quanto viene ritornato dalle API Fitbit in assenza di dati.

Per quanto riguarda i privilegi di accesso agli endpoint appena illustrati, ogni paziente potrà accedere solamente ai propri dati. Il medico (e l'amministratore) potrà leggere i dati di tutti i pazienti. Non si applicano delle restrizioni al solo gruppo di pazienti in cura per ciascun medico in modo tale da permettere eventuali riscontri anche ad altri medici, qualora fosse necessario.

3.3.3 Osservazioni

Riassumendo, tutta la procedura di comunicazione e recupero dati da Fitbit è totalmente trasparente ai client, essendo gestita dal server dell'applicazione. In questo modo, la difficoltà percepita dal paziente utilizzatore di HealthApp è minima ed è ridotta ad un overhead nel primo login in cui bisogna fornire un consenso per l'utilizzo dei dati. Successivamente, tutte le operazioni di recupero e presentazione delle informazioni sono automatizzate grazie alla sinergia tra backend e frontend.

Non rimane che illustrare l'ultimo gruppo di API relative a questo capitolo, ossia quello riguardante Cibo e Ricette.

3.4 Cibo e Ricette

Il paragrafo seguente illustra l'implementazione lato server degli endpoint per l'ultima macroarea, quella dell'alimentazione. L'obiettivo in partenza è quello di discostarsi dalla stragrande maggioranza delle applicazioni già presenti sulle varie piattaforme (mobile e non) che offrono un semplice diario alimentare in cui l'utente registra i propri pasti quotidiani. Nonostante quasi sempre venga fornito un riscontro in termini di calorie, l'obiettivo di HealthApp è quello di permettere al paziente di mantenere un diario con un riscontro approfondito che comprenda i diversi valori nutrizionali che compongono i cibi.

Il primo passo da fare riguarda quindi la ricerca di un dataset esistente contenente la maggioranza dei cibi consumati. I vincoli di tale ricerca impongono che:

1. Vi siano delle API che restituiscano i dati come risultato di ricerca. Il caso d'uso è infatti quello del paziente che registra il proprio pasto inserendo il nome del piatto e ricevendo una lista di riscontri possibili. A partire da tale lista potrà selezionare l'occorrenza opportuna e aggiungerla al diario scegliendone la quantità consumata e osservando i valori nutrizionali variare di conseguenza. Non si ritiene opportuno importare l'intera base dati internamente a causa dell'occupazione elevata di spazio che l'operazione richiederebbe.

2. Vi siano delle API che associno ad ogni piatto una serie di valori nutrizionali d'interesse.

3. Il dataset sia liberamente consultabile.

Tra i risultati della ricerca, solamente i due provider più importanti vengono riportati, discutendone pro e contro e motivando la scelta finale.

3.4.1 OpenFoodFacts

Questo provider offre un database di quasi tre milioni di record ed è completamente disponibile sotto licenza Open Database (quindi liberamente consultabile ed integrabile). È mantenuto dalla comunità, quindi in linea di principio chiunque può aggiungere dei record e contribuire alle informazioni.

Tipicamente, ogni prodotto presenta i valori nutrizionali più generali come calorie, grassi, carboidrati, proteine e fibre. In alcuni casi tali dati possono addirittura essere mancanti o possono essere presenti ulteriori informazioni come vitamine, zuccheri, sale ecc.

Inoltre, vengono messi a disposizione degli endpoint per ricercare i prodotti mediante una query (una stringa testuale) e recuperare le informazioni per ogni singolo cibo.

Se tra i pro della soluzione si ha l'estensione in termini di record della base dati e la totale libertà di consultazione senza limiti di chiamate o di dati consultabili, esistono anche diversi contro che impediscono ad OpenFoodFacts di essere utilizzato come fornitore di informazioni per HealthApp:

- È presente un'elevata eterogeneità tra le informazioni associate ai prodotti che è possibile reperire. In alcuni casi sono disponibili solamente i valori principali, mentre altri contengono dettagli sulle vitamine piuttosto che sugli zuccheri ed altri ancora non contengono alcun valore nutrizionale. Risulta quindi parecchio difficile riscontrare un'omogeneità dei dati con il rischio di importanti alterazioni nel calcolo totale dei valori nutrizionali mostrato al paziente.

Valori nutrizionali	Come venduto per 100 g / 100 ml
Grassi	?
Acidi Grassi saturi	?
Carboidrati	?
Zuccheri	?
Fibra alimentare	?
Proteine	?
Sale	?

Figura 9 - Valori nutrizionali "Cetriolo"

Valori nutrizionali	Come venduto per 100 g / 100 ml	Come venduto per porzione (30g)
Energia	1.621 kj (382 kcal)	486 kj (115 kcal)
Grassi	1,9 g	0,57 g
Acidi Grassi saturi	0,9 g	0,27 g
Acido alfa-linoleico	0,008 g	0,002 g
Carboidrati	84 g	25,2 g
Zuccheri	17 g	5,1 g
Fibra alimentare	3 g	0,9 g
Proteine	6,3 g	1,89 g
Sale	0,65 g	0,195 g
Vitamina D (colecalciferolo)	8,4 µg	2,52 µg
Vitamina B1 (tiamina)	0,91 mg	0,273 mg
Vitamina B2 (Riboflavina)	1,2 mg	0,36 mg
Vitamina B3	13,3 mg	3,99 mg
Vitamina B6 (piridoxina)	1,2 mg	0,36 mg
Vitamina B9 (Acido folico)	166 µg	49,8 µg
Vitamina B12 (Cobalamina)	2,1 µg	0,63 µg
Ferro	8 mg	2,4 mg
Frutta, verdura, noci e olio di colza, noci e oliva (stima dall'analisi dell'elenco degli	0 %	0 %

Figura 10 – Valori nutrizionali "Chicchi di riso soffiato al cioccolato"

- Essendo un database libero e basato sul lavoro della comunità, chiunque può contribuire aggiungendo record o modificando quelli esistenti. Per tale motivo non è garantita l'affidabilità dei dati immessi con il rischio di presentare informazioni errate ai pazienti ed ai medici. Nelle figure seguenti si faccia il confronto tra i valori nutrizionali per lo stesso prodotto di un noto brand dolciario:

Valori nutrizionali	Come venduto per 100 g / 100 ml
Energia	2.252 kj (539 kcal)
Grassi	30,9 g
Acidi Grassi saturi	10,6 g
Carboidrati	57,5 g
Zuccheri	56,3 g
Fibra alimentare	0 g
Proteine	6,3 g
Sale	0,107 g
Frutta, verdura, noci e olio di colza, noci e oliva (stima dall'analisi dell'elenco degli ingredienti)	0 %

Figura 11 – Valori nutrizionali per 100g crema spalmabile alle nocciole barattolo 1kg

Valori nutrizionali	Come venduto per 100 g / 100 ml
Energia	2.278 kj (539 kcal)
Grassi	31,6 g
Acidi Grassi saturi	11 g
Carboidrati	57,6 g
Zuccheri	56,8 g
Fibra alimentare	?
Proteine	6 g
Sale	0,114 g
Frutta, verdura, noci e olio di colza, noci e oliva (stima dall'analisi dell'elenco degli ingredienti)	0 %

Figura 12 – Valori nutrizionali per 100g crema spalmabile alle nocciole barattolo 800g

- Infine, un fattore di esclusione fondamentale del provider è dato dal fatto che la stragrande maggioranza dei record contiene solo prodotti preparati di origine industriale, senza concepire il piatto cucinato da zero partendo dalla ricetta. Ad esempio, cercando "carbonara" è possibile consultare le miriadi di versioni già pronte da consumare e non si ha nessun riscontro sul prodotto preparato partendo dagli ingredienti base.

Le motivazioni appena illustrate portano all'esclusione di tale dataset.

3.4.2 FatSecret

FatSecret nasce come un'applicazione che permette di tener traccia delle calorie consumate e dei valori nutrizionali associati. Presenta una base dati di cibi contenente un milione e mezzo [11] di record organizzati per divisioni geografiche. La divisione Italia, ad esempio, contiene piatti pronti come "Carbonara" o "Pasta al pesto" oltre che i corrispondenti preparati in busta, mentre tali dati non sono disponibili nella divisione USA.

Sono inoltre offerte delle API per accedere a questi dataset ma, trattandosi di soluzioni proprietarie, il piano base offre un limite di cinquemila chiamate giornaliere con possibilità di avere chiamate illimitate sottoscrivendo il piano premium a pagamento. Fortunatamente, per le startup e i progetti no-profit come HealthApp viene messa a disposizione una licenza gratis che consente di accedere illimitatamente al dataset USA che offre comunque un numero di informazioni sui piatti di maggiore diffusione abbastanza interessante. In seguito, si potrebbe pensare di sottoscrivere il piano premium per il dataset italiano.

La mancanza di alcuni piatti dovuti alla diversità geografica è sicuramente il fattore negativo riscontrato in tale soluzione. Tuttavia, i punti forti colmano tutte le lacune presentate da OpenFoodFacts:

- Per tutte le tipologie di record presenti sono presenti gli stessi valori nutrizionali, quindi i dati sono omogenei.

- Le informazioni sono verificate e non si basano interamente sul contributo della comunità. È possibile suggerire l'inserimento di dati mancanti ma questi vengono validati a dovere [12].

- Sono presenti sia valori nutrizionali per piatti industriali pronti e per piatti preparati a partire dagli ingredienti base.

Tutti i vincoli vengono soddisfatti se si considera che sono disponibili degli endpoint per effettuare la ricerca sulla base di una stringa e selezionare il miglior risultato per consultare le proprietà. I valori nutrizionali disponibili sono:

Tabella 2 - Valori nutrizionali alimentazione

Nome	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 A	µg
Vitamina C	mg
Vitamina D	µg
Calcio	mg
Ferro	mg

FatSecret risulta essere quindi la scelta più appetibile per i vincoli di HealthApp. Il passo successivo consiste nell'integrazione e nell'implementazione delle API da mettere a disposizione dei client.

3.4.3 Integrazione FatSecret con HealthApp

In maniera molto simile a quanto fatto per Fitbit, al fine di poter utilizzare le API messe a disposizione degli sviluppatori è necessario registrare nella console proprietaria la propria applicazione. Anche in questo caso è necessario fornire gli estremi e una breve descrizione del progetto per procedere. Completata la

registrazione, è necessario attendere la revisione della richiesta e l'approvazione del piano premium offerto gratuitamente per i progetti no profit.

In seguito alla registrazione si ottengono le credenziali OAuth 2.0 (client ID e client Secret). L'utilizzo di tale protocollo di autenticazione è però scoraggiato dalla presenza di una whitelist di IP implementata come controllo aggiuntivo da parte del server FatSecret; questo meccanismo rende maggiormente complicato il processo di sviluppo, in quanto la macchina locale che prova a connettersi con le API dei cibi lo fa sempre da indirizzi diversi e l'aggiornamento della whitelist richiede 24 ore per essere effettivo.

Per tale motivo si preferisce utilizzare le API attraverso l'autenticazione fornita dal protocollo OAuth 1.0 nonostante questo presenti dei problemi di confidenzialità [13]. Poiché i dati scambiati non si ritengono essere sensibili, si ritiene di non considerare di primaria importanza il problema.

Si procederà con un'illustrazione del meccanismo di autenticazione. Nel momento in cui l'applicazione viene approvata, si riceve una coppia di chiavi:

- *Consumer key*: una chiave univoca identificativa del consumer (HealthApp in questo caso) che intende accedere alle API.

- *Consumer secret*: una chiave segreta condivisa tra il consumer e il server di autenticazione.

Tale coppia verrà utilizzata per autenticare le richieste effettuate. Immaginiamo di voler accedere, utilizzando il metodo http GET, ad una risorsa che è raggiungibile tramite il percorso fittizio https://fatsecret.com/rest/api, la procedura da seguire è la seguente [13]:

1. Creazione della stringa base da firmare – nella richiesta dovranno essere inviati i seguenti parametri:
    - *oauth_consumer_key*: la chiave ricevuta dopo la registrazione
    - *oauth_signature_method*: HMAC-SHA1 è l'unico supportato da FatSecret.
    - *oauth_timestamp*: Data e ora all'atto della richiesta.
    - *oauth_nonce*: Stringa generata in maniera random all'atto della richiesta.
    - *oauth_version*: Assume sempre il valore "1.0"
    - *Parametri custom*: variabili a seconda della tipologia di Endpoint chiamato.

Partendo quindi dal metodo, dall'URL e dai parametri normalizzati (ossia ordinati alfabeticamente) si produce una stringa del tipo: *<HTTP Method>&<Request URL>&<Normalized Parameters>* in cui il carattere di concatenazione è la "&".

Tutta la stringa così generata deve essere codificata utilizzando il meccanismo *percent-encoding (%xx)* definito in *RFC3986* [14].

2. La stringa ottenuta al punto 1 deve essere firmata utilizzando l'algoritmo *HMAC-SHA1* in cui la chiave è la concatenazione tra *Consumer Secret* e *Access Secret* (che nel caso di HealthApp non è utilizzato ed è quindi pari alla stringa vuota) utilizzando come carattere di congiunzione la "&".

Il digest così ottenuto deve essere prima codificato in *base-64* secondo *RFC2045* [15], poi in *percent-encoding* secondo *RFC3986* [14]. Il risultato corrisponderà al parametro *oauth_signature*.

3. Si invia la richiesta mediante il metodo http scelto all'URL scelto includendo i parametri firmati più il parametro *oauth_signature* contenente la firma calcolata. Il server di FatSecret provvederà ad elaborare la richiesta, a ricostruire la stringa base e a firmarla utilizzando il segreto condiviso ottenendo così la sua versione di *oauth_signature*. Se le due versioni corrispondono, allora la richiesta sarà autenticata con successo e verrà fornita in risposta la risorsa richiesta.

3.4.4 Implementazione API

Partendo dalla procedura illustrata al punto 3.4.3 si include il riferimento alle API di interesse offerte da FatSecret:

- GET https://platform.fatsecret.com/rest/server.api?method=foods.search – Riceve i parametri di autenticazione e la stringa da ricercare sempre come parametro. Ritorna un oggetto JSON contenente un array di oggetti cibi. Ciascun oggetto è composto da un identificatore, un nome, un tipo (Brand o Generico), il nome del brand (presente solo se il tipo è uguale a Brand) e una breve descrizione dei valori nutrizionali.

- GET https://platform.fatsecret.com/rest/server.api?method=food.get.v2 – Riceve i parametri di autenticazione e l'id del record da recuperare sempre come parametro. Ritorna un oggetto JSON contenente l'id del cibo, il nome, la categoria e i valori nutrizionali indicati nella tabella 2.

Attraverso l'utilizzo di queste due API il backend di HealthApp provvede a memorizzare nella base dati solamente gli indentificativi degli alimenti consumati dal paziente, il quantitativo e i corrispondenti valori nutrizionali in proporzione. Inoltre, ogni record, oltre ad essere associato ad una data è anche associato ad una categoria e ad una tipologia di pasto. Le categorie sono in tutto venti e presentano lo stesso identificativo e lo stesso nome presenti su FatSecret:

Tabella 3 - Categorie Alimenti

Nome Categoria
Legumi
Bevande
Pane, Farine e Cereali
Formaggi, Latte e Latticini
Uova
Fast Food
Pesce e Frutti di mare
Frutta
Carni Rosse
Carni Bianche
Noci, Semi e Frutta Secca
Pasta e Riso
Insalate
Salse, Spezie e Creme Spalmabili
Snacks

Dolci, Caramelle e Desserts
Verdure e Ortaggi
Ricette Preparate
Altro
Oli e Grassi

Invece le tipologie di pasto sono codificate mediante una mappa in cui ad ogni intero corrisponde una tipologia:

Tabella 4 - Tipologie di pasto

Tipologia di pasto	**Valore intero corrispondente**
Colazione	0
Spuntino della Mattina	1
Pranzo	2
Spuntino del pomeriggio	3
Cena	4
Spuntino serale o di mezzanotte	5
Altro	6

L'accesso alle informazioni memorizzate e la ricerca di alimenti tramite l'uso di FatSecret avvengono mediante le API che il backend di HealthApp espone ai client:

- GET /api/patients/{patientID}/foods?startDate={startDate}&endDate={endDate} – Ritorna un oggetto JSON contenente un array di oggetti cibo relativi all'intervallo di date passato e al paziente identificato mediante il contenuto della variabile patientID. Ogni oggetto contiene un identificativo, la data, la categoria, la quantità consumata, la tipologia di pasto associata e i valori nutrizionali secondo la tabella 2.

- GET /api/patients/{patientID}/foods/meal?startDate={startDate}&endDate={endDate}&mealType={mealType} – Ritorna un oggetto JSON contenente un array di oggetti cibo relativi all'intervallo di date, alla tipologia di pasto e al paziente passati come parametri. Ogni oggetto contiene un identificativo, la data, la categoria, la quantità consumata, la tipologia di pasto associata e i valori nutrizionali secondo la tabella 2.

- GET /api/foods/categories – Ritorna un oggetto JSON contenente un vettore di oggetti categoria. Ogni oggetto è caratterizzato da un identificativo e dal nome della categoria di alimenti che descrive.

- PUT /api/patients/{patientID}/foods/{date}/{mealType}/{idF} – Riceve un oggetto JSON contenente l'id di un alimento, la data in cui è stato consumato, la tipologia di pasto, il nome,

l'identificativo della categoria, la quantità consumata. Provvede ad aggiornare il record cibo presente nella base dati, relativo al paziente il cui identificativo è uguale al contenuto della variabile patientID, identificato mediante la tripla data, tipologia di pasto e identificatore. Inoltre, ne calcola gli opportuni valori nutrizionali con il supporto dei dati forniti da FatSecret.

- POST /api/foods/search – Riceve un oggetto JSON contenente una stringa corrispondente all'espressione da ricercare. Inoltra la chiamata a FatSecret e ritorna un oggetto JSON contenente un array di oggetti cibi. Ciascun oggetto è composto da un identificatore, un nome, un tipo (Brand o Generico), il nome del brand (se il tipo è uguale a Brand) e una breve descrizione dei valori nutrizionali contenuti.

- POST /api/patients/{patientID}/foods - Riceve un oggetto JSON contenente l'id di un alimento, la data in cui è stato consumato, la tipologia di pasto, il nome, l'identificativo della categoria, la quantità consumata. Provvede a salvare il record cibo presente nella base dati, relativo al paziente il cui identificativo è uguale al contenuto della variabile patientID, calcolandone gli opportuni valori nutrizionali con il supporto dei dati forniti da FatSecret.

- DELETE /api/patients/{patientID}/foods/{date}/{mealType}/{idF} – Provvede ad eliminare il record cibo presente nella base dati, relativo al paziente il cui identificativo è uguale al contenuto della variabile patientID, e identificato mediante la tripla data, tipologia di pasto ed identificatore.

Gli endpoint appena descritti consentono quindi un'implementazione lato server piuttosto esaustiva della macroarea alimentazione. Anche in questo caso il meccanismo di recupero dati da un server di terze parti è totalmente trasparente ai client e quindi agli utenti utilizzatori, i quali non dovranno minimamente preoccuparsi di gestire la complessità dell'autenticazione e della formulazione delle richieste.

Tuttavia, lo scopo di questa macroarea, illustrato all'inizio del paragrafo, è quello di fornire un approccio innovativo al cosiddetto diario alimentare che vada oltre la semplice aggiunta dei valori nutrizionali. Per questo motivo, si descriverà nel paragrafo successivo l'implementazione del sistema di ricette.

3.4.5 Ricette

L'idea del sistema di ricette nasce dal fatto che le persone al giorno d'oggi fanno sempre più attenzione alla linea e agli alimenti che consumano, a maggior ragione se sono dei pazienti in cura presso una clinica ospedaliera. Per questo motivo, è molto frequente trovare delle varianti di piatti in cui "l'alimento più grasso viene escluso" oppure "viene utilizzato un ingrediente sostitutivo perché presenta valori nutrizionali differenti". Per tale motivo si è pensato di introdurre all'interno della macroarea alimentazione una sezione dedicata alle ricette definite dal paziente stesso.

Tale sezione, totalmente indipendente dal provider di terze parti FatSecret, costituisce una funzionalità esclusiva di HealthApp. Essa si basa su un dataset di ingredienti base interamente proprietario che conta attualmente 568 record i cui valori nutrizionali e le categorie seguono rispettivamente le tabelle 2 e 3 e sono quindi coerenti e omogenei con i dati offerti da FatSecret.

Il paziente può definire le proprie ricette come composizione di ingredienti base in quantità arbitrarie. Esse verranno salvate all'interno della base dati dell'applicazione e, sulla base delle quantità degli alimenti base selezionati, verranno calcolati i valori nutrizionali complessivi della ricetta. All'atto della registrazione nel diario alimentare, il paziente può quindi scegliere di ricercare un cibo presente nel dataset FatSecret o di inserire la sua ricetta, in una quantità consumata anche differente da quella registrata: il sistema provvederà automaticamente a calcolare i valori nutrizionali aggiornati e ad associare il record alla data e alla tipologia di pasto, procedendo al salvataggio nella base dati. Cibi provenienti da FatSecret e ricette definite dall'utente risultano quindi essere perfettamente compatibili e offrono un quadro omogeneo per quanto riguarda le informazioni fornite.

Le API precedentemente definite relative al cibo rimangono quasi del tutto invariate, ad eccezione di alcuni valori in più da gestire:

- Per ogni cibo o ricetta registrata, modificata o letta sarà presente un campo che determinerà se l'origine dei dati è FatSecret oppure il dataset interno di HealthApp. Nel primo caso l'identificativo corrisponderà a quello del prodotto su FatSecret, nel secondo a quello della ricetta personale del paziente.

Si mettono a disposizione dei client degli ulteriori endpoint per il supporto alle operazioni CRUD per le ricette e per la lettura degli ingredienti base:

- GET /api/foods/ingredients - Ritorna un oggetto JSON costituito da un array di ingredienti base. Ogni oggetto rappresenta un ingrediente base e contiene un identificativo, un nome, una categoria, un quantitativo di riferimento e un insieme di valori nutrizionali coerente con la tabella 2.

- GET /api/patients/{patientID}/foods/recipes – Ritorna un oggetto JSON che comprende un vettore di ricette relative al paziente identificato dal contenuto del parametro patientID. Ogni ricetta conterrà un identificativo, un nome, una lista di ingredienti base (costituiti da un identificativo, un nome, una categoria, dal quantitativo presente e dall'insieme dei valori nutrizionali previsti dalla tabella 2) e il calcolo totale dei valori nutrizionali della ricetta (calcolato come somma dei valori nutrizionali degli ingredienti base).

- PUT /api/patients/{patientID}/foods/recipes/{idR} – Riceve un oggetto JSON contenente un nome e una lista di ingredienti base (costituiti da un identificativo, un nome e dal quantitativo presente); provvede ad aggiornare la ricetta identificata mediante idR e relativa al paziente identificato da patientID, recuperando i valori nutrizionali per ciascun ingrediente base e calcolando il totale dei valori nutrizionali della ricetta.

- POST /api/patients/{patientID}/foods/recipes – Riceve un oggetto JSON contenente un nome e una lista di ingredienti base (costituiti da un identificativo, un nome e dal quantitativo presente); provvede a salvare la ricetta all'interno della base dati associandola al paziente identificato da patientID, recuperando i valori nutrizionali per ciascun ingrediente base e calcolando il totale dei valori nutrizionali della ricetta.

- DELETE /api/patients/{patientID}/foods/recipes/{idR} – Provvede ad eliminare dalla base dati la ricetta identificata mediante idR ed associata al paziente identificato mediante il contenuto della variabile patientID.

I dati relativi ai valori nutrizionali ritornati dal backend di HealthApp ai client possono assumere il valore *null* in quanto in alcuni casi non è detto che siano presenti tutte le informazioni, specialmente nel caso di alcuni ingredienti base esotici o bevande alcoliche. Nel conteggio dei valori nutrizionali totali di una ricetta, quando è presente il valore *null* si assume per semplicità che questo sia pari a zero.

3.4.6 Osservazioni

Ricapitolando, l'idea di HealthApp è quella di supportare la macroarea alimentazione attraverso un diario alimentare corredato dei valori nutrizionali per alimento ed esteso con la possibilità di definire ricette personalizzate per paziente. Tale funzionalità dovrebbe permettere la registrazione di un rendiconto alimentare completo ed anche la possibilità di seguire una particolare dieta formata da specifiche ricette ricevute in seguito ad indicazioni mediche.

I valori nutrizionali di per sé risultano essere molto importanti ai fini delle recommendations. Infatti, il sistema di analisi è in grado di rilevare eventuali eccessi di carboidrati, minerali, vitamine, proteine o grassi e indicarlo prontamente a paziente e medico, favorendo una tracciabilità che, al contrario di analisi mediche, attività e peso risulterebbe più complicata da misurare.

Completata l'illustrazione della struttura complessiva del backend a supporto delle quattro macroaree, la discussione si sposta nel prossimo capitolo sul modulo delle recommendations.

4. Recommendations – introduzione e scelta business rule engine

4.1 Introduzione al sistema delle recommendations

Il sistema delle recommendations rappresenta un modulo fondamentale di HealthApp. Il suo scopo è quello di fornire al paziente degli spunti su come migliorare il proprio stile di vita analizzando le informazioni relative alle quattro macroaree (Alimentazione, Movimento, Sonno e Salute) e proponendo dei consigli variabili a seconda del risultato dell'analisi.

Tale soluzione si propone come innovativa rispetto a quanto offerto dalle altre applicazioni di settore in quanto consente al paziente di non avere solamente un banale resoconto sui progressi effettuati ma anche di comprendere il perché alcune azioni (ad esempio assumere un tot di carboidrati piuttosto che fare un tot di passi al giorno) possano giovare alla salute. Inoltre, i consigli proposti possono servire anche al medico curante come spunti di valutazione e riflessione del percorso effettuato dal paziente.

Poiché questa porzione dell'applicazione risulta essere abbastanza indipendente rispetto alle altre, la si considera come un modulo a sé.

L'elemento fondamentale del modulo risulta essere l'analisi dei dati. Occorre individuare un meccanismo grazie al quale si possano selezionare i parametri da tenere in considerazione con estrema flessibilità. Infatti, dovrà essere possibile personalizzare l'analisi per ogni paziente, in base alle caratteristiche di quest'ultimo.

Un ulteriore obiettivo è quello di mantenere la comprensibilità e la manutenibilità del codice; risulta quindi impossibile creare una routine con dei blocchi "if-then-else" o degli "switch-case" in quanto la complessità tenderebbe ad aumentare al crescere dei parametri da analizzare.

La soluzione adottata è quindi quella del Business Rule Engine (BRE). Un BRE è un modulo software che gestisce delle regole e dei dati in input producendo dei dati in output. [16]

Una regola non è altro che un flusso di esecuzione che segue una struttura di tipo ECA (Event Condition Action) [17].

- Event: è il segnale che scatena l'invocazione delle regole. Nel nostro caso tale segnale può essere implicito (ossia prodotto da una routine eseguita settimanalmente dal server) oppure esplicito (ossia generato da una chiamata ad un'API REST da parte del paziente/medico)

- Condition: è la parte logica della regola, che se verificata permette l'esecuzione dell'azione.

- Action: è la conseguenza di una condizione verificata e produce in output dei dati.

Il Business Rule Engine riceve quindi dei dati in input, li applica su un set di regole e produce dei risultati in output.

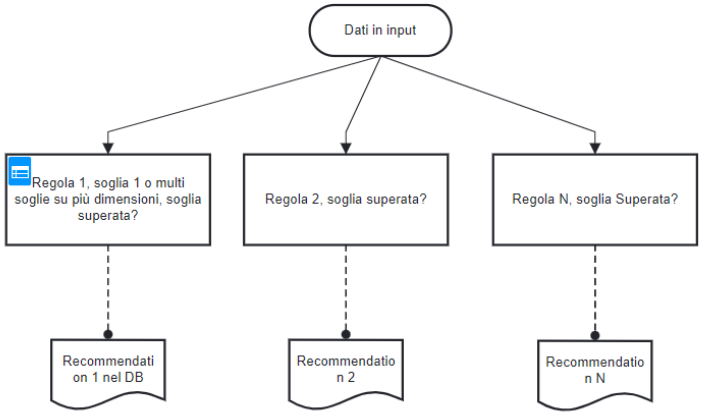


Figura 13 - Schema Riassuntivo BRE

Ogni regola potrà contenere una o più condizioni relativa ad uno o più parametri (da verificare contemporaneamente) di una macroarea che, se verificati, comporteranno la generazione di un consiglio da mostrare al paziente. Esisteranno poi regole maggiormente complesse che verteranno su più macroaree in contemporanea (es Movimento e Alimentazione). Le condizioni riguarderanno delle soglie che possono rappresentare un limite superiore o inferiore e l'azione dipenderà dal superamento o meno di tali soglie. Nel caso in cui i dati analizzati rimangano all'interno delle soglie previste, la recommendation prodotta sarà di tipo positivo, volta principalmente a far notare al paziente che il suo percorso, relativamente ai parametri analizzati, è corretto. Viceversa, in caso di superamento delle soglie, viene prodotta una recommendation di tipo negativo, in cui si spiega soprattutto il perché tale superamento possa essere pericoloso per la salute.

4.2 Scelta business rule engine

Facendo riferimento ai requisiti illustrati in precedenza, occorre trovare una soluzione che permetta di definire un numero arbitrario di regole, di dividerle in categorie (nel caso di HealthApp per macroaree), e di definire dei parametri singoli/multipli che siano variabili e personalizzabili per ogni paziente.

Tra le soluzioni valide vi sono:

- DecisionRules
- Hyperion
- Evrete
- Drools

4.2.1 DecisionRules

DecisionRules è un BRE abbastanza leggero che offre la possibilità di creare, gestire e distribuire regole. Tuttavia, l'interazione con tale sistema avviene attraverso chiamate ad API al server proprietario e di conseguenza non risulta integrabile all'interno dell'applicazione sotto forma di modulo. Inoltre, il piano free permette di definire un massimo di due regole [18], altamente limitante per le prospettive illustrate in HealthApp.

Di conseguenza questa prima soluzione viene esclusa.

4.2.2 Hyperion

Hyperion si presenta come un BRE con caratteristiche molto simili a DecisionRules. A differenza del precedente, esso non è solamente utilizzabile attraverso chiamate ad API verso il server proprietario, ma può essere utilizzabile anche come una libreria Java integrabile nella propria applicazione. Tale soluzione sarebbe perfetta, in quanto il BRE diventerebbe un modulo di HealthApp. Tuttavia, anche questa è da escludere a causa dell'assenza di un piano free. Non esistono limitazioni sulle regole definibili ma il costo della licenza annuale parte da 10000$, decisamente insostenibile per un progetto di tesi come HealthApp [19].

4.2.3 Evrete

Evrete è un rule engine open source totalmente integrabile in Java, senza alcuna dipendenza esterna. Rappresenta una soluzione abbastanza leggera che permette di definire regole secondo lo schema ECA. Le limitazioni di tale soluzione sono tuttavia date da un meccanismo di definizione delle regole abbastanza verboso e poco chiaro, nonché da uno scarso supporto della community. Per tale motivo si preferisce utilizzare Drools anziché Evrete, soluzione che verrà illustrata approfonditamente nei paragrafi successivi.

4.2.4 Drools

Drools è un BRE totalmente open source, scritto in Java. È possibile utilizzarlo come libreria da integrare nella propria applicazione, è ampiamente supportato dalla community, è possibile integrarlo nel framework Spring e presenta un enorme vantaggio: è possibile scrivere le regole con un linguaggio molto poco verboso ed è possibile separare la logica (quindi la scrittura delle regole) dal codice di configurazione ed esecuzione del modulo. Inoltre, è presente un DMN (*Decision Model and Notation*), ossia un linguaggio che permette di creare schemi molto facilmente comprensibili (come quello visibile in figura 13).

Drools rappresenta quindi la soluzione maggiormente appetibile per HealthApp.

4.3 Specifiche di drools

Drools opera utilizzando i seguenti componenti base [20]:

- **Regole:** Regole definite dal programmatore, con la possibilità di utilizzare variabili come soglie e che quindi si adattano alla necessità di avere parametri personalizzati per ogni paziente.
- **Fatti:** Dati in input, utilizzati per verificare se le condizioni delle regole sono soddisfatte.
- **Production Memory:** Area di memoria in cui vengono salvate le regole
- **Working Memory**: Area di memoria in cui vengono salvati i fatti
- **Agenda:** Area dove le regole attivate vengono registrate e ordinate (in base alla priorità, se definita) in preparazione all'esecuzione

L'intero sistema è parte di un progetto chiamato KIE (*Knowledge is everything*) che copre differenti moduli. In HealthApp verrà utilizzato solamente Drools. Per avviare il motore di esecuzione regole è necessario definire una sessione. Le sessioni possono essere di due tipi, *stateful* o *stateless* [21].

Una sessione di tipo *stateful* è caratterizzata dalla presenza di uno stato, ossia il sistema ricorda i dati tra l'invocazione di una sessione ed un'altra. I dati vengono inseriti nella sessione prima dell'esecuzione delle regole e dopo l'esecuzione delle regole deve essere chiamato il metodo *dispose*() al fine di evitare memory leaks. Inoltre, ogni modifica alla working memory effettuata mediante i metodi *insert ()* o *modify ()*, metodi che permettono di inserire dati nella sessione, viene notificata al motore di esecuzione. Offre il metodo *fireAllRules()* per eseguire tutte le regole di un file indicato. Si adatta particolarmente ad un'esecuzione generata dalla chiamata ad un'API o innescata da una routine periodica.

Una sessione di tipo *stateless* non presenta alcuno stato, ed è adatta per esecuzioni one-shot delle regole. Non permette di mantenere i dati tra le sessioni e non supporta il comando *fireAllRules()* per l'esecuzione multipla delle regole. Inoltre, il motore di esecuzione non viene notificato di eventuali cambiamenti ai fatti mediante i metodi *insert ()* o *modify()*.

Nell'integrazione in HealthApp vengono utilizzate sessioni *stateful*, in modo tale che se all'interno di una sessione vi è un cambiamento dei dati, le regole verranno eseguite sulle nuove informazioni. Inoltre, è possibile eseguire tutte le regole di un file selezionato con un unico comando.

I file che contengono le regole sono file di testo che hanno estensione .drl; essi fanno parte delle risorse dell'applicazione (si trovano nella cartella *resources*) favorendo così il disaccoppiamento della logica dal codice. È possibile creare più file e selezionare per ogni sessione quale eseguire. Nel caso di HealthApp esistono differenti file corrispondenti alle categorie principali delle varie macroaree.

I componenti di un DRL file sono i seguenti:

package

import

function // Opzionale

query // Opzionale

declare // Opzionale

global // Opzionale

rule "nome regola"

// Attributi

when

// Condizioni

then

// Azioni

end

rule "rule2 name"

...

Ogni package raggruppa una serie di asset correlati e fornisce un namespace unico per ogni regola. Le regole devono avere un nome univoco all'interno del package, e tipicamente vengono scritte in sequenza come ultimo componente del file.

"Global" rappresenta invece una variabile globale che sarà condivisa tra tutte le regole componenti il file DRL.

Una regola Drools ha una specifica struttura riassumibile con il seguente schema:

rule "nome regola"

// Attributi

when

// Condizioni

then

// Azioni

end

Le parole chiave definiscono le differenti componenti delle regole. *"Rule"* seguito dal nome della regola tra virgolette serve a dichiarare che tutto quello che segue fino a *"End"* è parte della regola identificata da quel particolare nome.

È possibile dichiarare molteplici attributi per ogni regola. Quelli di maggiore interesse per HealthApp sono:

- *salience*: è un numero intero che può essere positivo o negativo e rappresenta la priorità associata ad una regola. Il valore di default è zero. L'ordine di esecuzione è dato dalla *salience* maggiore. Regole a priorità maggiore saranno eseguite prima, altrimenti in caso di parità non è garantito un determinato ordine di esecuzione.

- *agenda-group*: è una stringa che permette di definire dei gruppi di regole specificando ogni regola a quale gruppo appartiene. È utile in quanto è possibile scegliere di eseguire solo le regole di un determinato gruppo.

- *activation-group*: è una stringa che permette di definire un sottogruppo di mutua esclusione tra le regole. Tra tutte le regole aventi lo stesso *activation-group* ne viene eseguita solamente una per cui le condizioni sono verificate mentre tutte le altre vengono automaticamente escluse e non valutate dal motore di esecuzione. Tale meccanismo è molto importante per scrivere diverse regole che rappresentano idealmente i rami di un blocco "*if-else if-else if ... else*"

- *dialect*: una stringa che può assumere i valori "*Java*" oppure "*Mvel*". Rappresenta il linguaggio con cui verranno scritte le regole. *Mvel*, che è la soluzione adottata all'interno dell'applicazione, è un linguaggio molto simile al linguaggio naturale e quindi semplice da scrivere e comprendere; permette di scrivere in maniera molto pulita il corpo di una regola.

In ogni sessione è possibile scegliere un particolare gruppo di regole da eseguire utilizzando il metodo *getAgendaGroup("nome del gruppo").setFocus()*. Per mantenere i differenti gruppi di regole separati ed ordinati, si è preferito assegnare un file DRL a ciascun'agenda group.

"*When*" rappresenta la parte sinistra di una regola (*LHS o Left End Side*). In questa sezione vengono dichiarate le condizioni che devono essere verificate affinché l'azione venga eseguita. Tipicamente, le condizioni fanno riferimento a degli attributi di un oggetto ed è possibile utilizzare la seguente notazione per esprimerle:

Object1(attr1 <>= valore o variabile && || attr2 <>= val ...)

Object2(...)

È possibile specificare più oggetti nel LHS di una regola ma le condizioni descritte valgono solo in regime di AND, non è possibile utilizzare l'operatore OR. All'interno delle condizioni di un singolo oggetto è invece possibile utilizzare un qualsiasi operatore logico.

Qualora si dovessero confrontare due attributi di due oggetti differenti è possibile definire delle variabili. Una variabile è identificata dal simbolo "$" seguito dal nome della variabile. Per espandere il

contenuto di una variabile si usa la stessa notazione utilizzata in fase di definizione della stessa. Ad esempio:

Object1 ($var1 : attr1)

Object2 ($var2 : attr2, $var1 <>= $var2)

Una cosa molto importante da verificare ,per evitare eccezioni durante l'esecuzione delle regole, riguarda il fatto che gli attributi oggetto delle condizioni siano diversi da null, ossia contengano una referenza valida.

"*Then*" identifica la parte destra di una regola (*RHS o right end side*). In questa sezione è possibile definire l'azione da eseguire nel momento in cui la condizione di una regola è verificata. Tipicamente tale azione può consistere nel manipolare dei dati, restituire un oggetto di ritorno, depositare informazioni in un db, generare un log o un report.

Nel caso di HealthApp viene utilizzato un oggetto globale comune a tutte le regole dello stesso gruppo (e file), contenente come attributo una lista di risultati. Ogni risultato fa riferimento ad un determinato sottogruppo di regole mutuamente esclusivo e contiene, tra tutti i parametri, una stringa che rappresenta il consiglio relativo a quel sottogruppo di regole. Tale insieme di risultati verrà poi trasmesso ai client sotto forma di report PDF o di oggetto JSON.

Per rendere meglio l'idea sul concetto espresso, la struttura del risultato seguirà il seguente schema:

Risposta [

Sottogruppo 1 : "consiglio X",

Sottogruppo 2 : "consiglio Y",

Sottogruppo 3 : "consiglio Z",

...

]

I file di regole definiti vengono raccolti in un unico file di configurazione che espone il cosiddetto "container". Il container rappresenta un'istanza di esecuzione del BRE, nel quale è possibile manipolare le sessioni.

Ogni file, come ribadito in precedenza, descrive un particolare gruppo di regole ed è associato ad un servizio Spring. Tale servizio è incaricato di creare e terminare una sessione, inserendovi i dati e invocandone l'esecuzione delle regole.

Ogni servizio, coerentemente con quanto visto nel capitolo introduttivo, è infine associato ad un controller che espone le API da chiamare per innescare l'esecuzione delle regole. Oltre alla presenza

delle API REST, una routine periodica, schedulata ogni settimana, esegue le regole applicandole sulle informazioni relative a tutti i pazienti presenti nel sistema.

Il capitolo seguente illustrerà nel dettaglio l'implementazione delle regole all'interno di HealthApp.

5. Implementazione sistema recommendations

Per implementare il recommender system è necessario scegliere innanzitutto le regole e le soglie. Sulla base di esse si concentrerà l'analisi dei dati relativi alle quattro macroaree che poterà alla produzione delle recommendations per il paziente.

Per effettuare tale scelta è possibile seguire due approcci differenti:

- *Theory-Based*: prevede la consultazione di documentazione, manuali, testi e convenzioni in ambito medico, al fine di individuare dei criteri che definiscano le regole. Queste ultime saranno quindi stabilite dalla teoria esistente o da pareri di soggetti esperti nel settore. L'approccio *theory-based* presenta il vantaggio di non necessitare di un'ampia base dati per estrarre i pattern rilevanti ed è immediatamente applicabile. Tuttavia, permette di definire delle soglie statiche che potrebbero risultare poco efficaci nell'analisi di parametri relativi a pazienti con particolari patologie che necessiterebbero di soglie ad-hoc.

- *Data-Driven*: basato sull'analisi di un'elevata mole di dati attraverso l'utilizzo di algoritmi e tecniche di data science come ad esempio il data mining, il machine learning e l'applicazione delle reti neurali. Permette di estrarre dei pattern comuni che vanno a determinare le soglie sulla base delle informazioni presenti. In tal caso è possibile avere diversi cluster di pazienti, ciascuno con regole personalizzate e quindi definire regole significative per diverse classi di assistiti. Tuttavia, è necessario che i dati siano accurati e verificati, altrimenti si rischia di produrre delle regole "artificiali" e poco applicabili alla realtà.

Nell'implementazione all'interno di HealthApp, non essendo disponibili dei dataset accurati e vasti al tempo stesso, si sceglie di seguire un approccio di tipo *Theory-Based*. Di conseguenza le regole definite hanno origine da nozioni mediche ampiamente verificate. Vengono classificate a seconda del numero e della tipologia di parametri coinvolti; infatti, è possibile definire regole:

- A parametro singolo: caratterizzate da un solo parametro relativo ad una singola macroarea. Tali regole hanno ovviamente una complessità inferiore e sono quelle presenti in maggior numero all'interno del recommender system.

- A parametri multipli: caratterizzate da due o più parametri relativi ad una singola macroarea. Tipicamente, i parametri vengono messi in relazione tra di loro mediante operatori logici o algebrici.

- A macroarea multipla: caratterizzate da due o più parametri relativi a due o più macroaree. Tipicamente, i parametri vengono messi in relazione tra di loro mediante operatori logici e algebrici.

Ognuna di queste classi avrà delle soglie statiche definite mediante un sistema nozionistico, che come ribadito in precedenza, offre poca flessibilità verso particolari classi di assistiti che necessitano di parametri ad hoc. L'idea iniziale è quindi quella di avere una o più tabelle nel database contenenti tali soglie, uguali per tutti i pazienti, che verranno recuperate all'atto dell'applicazione delle regole. Di base, le soglie statiche si riferiscono al "caso peggiore", in modo tale da coprire l'intera popolazione di assistiti. Ad esempio, nella regola per le calorie immesse, utilizzare la soglia media di una donna sarebbe un errore in quanto si

scatenerebbero delle recommendations di tipo negativo dovute ad una copertura parziale. Infatti, il fabbisogno calorico medio di un uomo è mediamente più alto di quello di una donna e non deve per tale motivo essere valutato negativamente dal recommender system. Tale discorso perde ovviamente il suo valore qualora sia il medico a determinare una particolare soglia per il suo assistito.

Poiché un approccio interamente *theory-based* risulterebbe poco funzionale nell'applicazione, si sceglie di introdurre maggiore flessibilità attraverso un sistema di soglie variabili per paziente. Anziché avere un insieme di criteri comuni, ogni assistito avrà associate le proprie soglie. Queste ultime, inizialmente, corrisponderanno alle soglie statiche definite dalla teoria, registrate all'atto della creazione dell'account di tipo paziente, ma potranno essere modificate successivamente dai medici per supportare al meglio ogni necessità. Di conseguenza, all'interno della base dati vi sarà un'occupazione maggiore di spazio in quanto ogni tabella conterrà i valori personalizzati per ogni paziente. L'overhead spaziale risulta ampiamente accettabile in quanto offre una flessibilità molto maggiore rispetto ad un approccio puramente *theory-based*. Rimangono escluse le regole a macroarea multipla e parzialmente quelle a parametri multipli, generalmente basate su rapporti o comparazioni tra differenti parametri ed il cui risultato deve essere compreso in un range ben preciso (così come le regole relative alle analisi mediche).

Una peculiarità delle sole regole a parametro singolo è che l'analisi, oltre ad essere effettuata a livello giornaliero, viene effettuata anche a livello settimanale e mensile. I valori settimanali e mensili vengono calcolati come la media aritmetica dei valori disponibili. Sul risultato ottenuto viene applicata la regola, producendo l'output settimanale piuttosto che mensile. Tale metodologia non viene applicata sulle regole a parametri multipli o a macroarea multipla. Il motivo di tale scelta è da ricercare nel modo in cui i dati verranno presentati. Se per le regole a parametro singolo ha senso calcolare i valori medi settimanali e mensili e applicarvi le regole, in quanto vengono associati ad indicatori presenti sul frontend, non viene ritenuto importante farlo per le altre regole, in quanto non vi è alcun indicatore associato.

Dopo aver classificato con precisione le varie tipologie di regole presenti in HealthApp, occorre procedere con la loro scrittura e implementazione.

5.1 Definizione regole

La definizione delle regole avviene seguendo le quattro macroaree componenti l'applicazione (alimentazione, movimento, salute, sonno). Ogni macroarea viene a sua volta suddivisa in gruppi d'interesse, per ognuno dei quali si definisce un insieme di risorse associato ben preciso. Ogni gruppo conterrà:

- Una tabella contenente i parametri personalizzabili per paziente.
- Una tabella contenente i report periodici generati dal recommender system sotto forma di file.
- Un file di regole associato contenente un insieme di regole aventi la stessa "*agenda-group*" che identifica un gruppo di regole. Esso conterrà a sua volta dei sottogruppi eseguiti in mutua esclusione.

- Un insieme di API associate per permettere ai client di eseguire l'analisi su un determinato paziente.

I gruppi individuati sono i seguenti:

1. Attività: Contenente regole a parametro singolo su passi effettuati e calorie bruciate.

2. Analisi Mediche: Contenente regole a parametro singolo e a parametri multipli sui valori che descrivono le analisi mediche e che sono stati indicati nella tabella 1 del capitolo 3.

3. Cibo: Contenente regole a parametro singolo sui valori che descrivono il cibo individuati dalla tabella 2 del capitolo 3. Vi sono anche regole a macroarea multipla e a parametri multipli.

4. Frequenza Cardiaca: Contenente regole a parametro singolo relative alla frequenza cardiaca a riposo.

5. Sonno: Contenente regole a parametro singolo relative al tempo di sonno.

6. Peso: Contenente regole a parametro singolo relative a massa e a indice di massa corporea (BMI).

Le regole a macroarea multipla vengono definite, per comodità, all'interno del file relativo al gruppo di maggiore rilevanza. Ad esempio, la regola riguardante la relazione tra calorie immesse e bruciate viene definita all'interno del gruppo cibo.

Nei paragrafi seguenti verranno descritte nel dettaglio le regole relative ai diversi gruppi.

5.1.1 Attività

Il gruppo attività si compone di due sottogruppi di regole: passi e calorie bruciate.

Per quanto riguarda i passi, vi sono quattro regole appartenenti allo stesso *activation-group*, ossia eseguite in mutua esclusione. In ogni *activation-group* è presente una sola regola associata ad una recommendation di tipo positivo, ossia quella in cui tutti i parametri si mantengono nella giusta relazione con le rispettive soglie. Tutte le ulteriori regole sono di tipo negativo:

- "*Steps are ok*" – Questa regola è verificata se e solo se, per il paziente e la data indicata, esiste il valore dei passi e della soglia. Inoltre, i passi sono maggiori o uguali alla soglia corrispondente. Corrisponde alla regola del gruppo associata al risultato ideale.

- "*Steps are null*" – Questa regola è verificata nel momento in cui i passi associati al paziente in tale data assumono il valore null.

- *"Steps have null threshold"* – Questa regola è verificata nel momento in cui la soglia associata ai passi per il paziente assume il valore null.

- *"Steps low"* – Questa regola è verificata quando, per un dato paziente ed in una determinata data, sia i passi che la soglia esistono ed assumono un valore differente da null. Inoltre, il numero di passi effettuati è inferiore alla soglia.

La soglia di riferimento per i passi è di base fissata al valore di diecimila. Tale valore è determinato da un importante studio di riferimento. [22]

In relazione al sottogruppo relativo alle calorie bruciate si hanno le seguenti regole:

- *"Calories are ok"* – Questa regola è verificata se e solo se, per il paziente e la data indicata, esiste il valore delle calorie bruciate e della soglia. Inoltre, le calorie bruciate sono maggiori o uguali alla soglia. A tale regola è associata una recommendation di tipo positivo.

- *"Calories are null"* – Tale regola è verificata se, per il paziente e la data indicate, si ha che il valore di calorie bruciate è pari a null.

- *"Calories have null threshold"* – Tale regola è verificata se, per il paziente e la data indicate, si ha che il valore di soglia è pari a null.

- *"Calories low"* – Tale regola è verificata se, per il paziente e la data indicate, si ha che i valori di calorie bruciate e della soglia esistono. Inoltre, le calorie bruciate sono inferiori alla soglia.

La soglia standard relativa alle calorie bruciate è pari al valore di 1200. Tale soglia risulta piuttosto bassa rispetto al consumo calorico di un essere umano adulto, indipendentemente dal sesso, ma è motivata dal fatto che tale valore si riferisce al consumo calorico ideale per una persona anziana di età superiore ai 74 anni. [23] È necessario considerare il caso limite per determinare la soglia. Essa potrà tuttavia essere modificata dai medici.

5.1.2 Analisi mediche

Il gruppo analisi mediche si articola in differenti sottogruppi: Eritrociti, Emoglobina, Mcv, Ht, Leucociti, Piastrine, Glicemia, Urea, Na, K, Creatinina, Colesterolo Totale, Colesterolo HDL, Trigliceridi, Pcr, Rapporto Colesterolo Totale / Colesterolo HDL , Rapporto Urea / Creatinina e Rapporto Trigliceridi / Colesterolo HDL.

I primi quindici sottogruppi contengono regole a parametro singolo secondo la tabella 1 del capitolo 3. Le soglie sono quelle illustrate nella medesima tabella e stabilite dalle convenzioni mediche. Gli ultimi tre sottogruppi costituiscono insiemi di regole a parametri multipli in cui viene valutato il rapporto tra differenti elementi di analisi mediche.

Tra le regole associate al primo sottogruppo, quello degli eritrociti si ha:

- *"Eritrociti are ok"* - Verificata se e solo se, per la rispettiva data e il paziente dato, esistono eritrociti e soglie. Inoltre, il valore degli eritrociti è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"Eritrociti are null"* - Verificata se, per il paziente e la data forniti, il valore degli eritrociti è pari a null.

- *"Eritrociti have min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"Eritrociti bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"Eritrociti low"* – Verificata se, per il paziente e la data forniti, il valore degli eritrociti è inferiore alla soglia minima.

- *"Eritrociti high"* – Verificata se, per il paziente e la data forniti, il valore degli eritrociti è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a $4{,}00 \cdot \frac{10^{12}}{l}$ e $5{,}20 \cdot \frac{10^{12}}{l}$

Per quanto riguarda invece l'emoglobina:

- *"Emoglobina is ok"* - Verificata se e solo se, per la rispettiva data e il paziente dato, esistono emoglobina e soglie. Inoltre, il valore dell'emoglobina è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"Emoglobina is null"* - Verificata se, per il paziente e la data forniti, il valore dell'emoglobina è pari a null.

- *"Emoglobina has min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"Emoglobina has bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"Emoglobina low"* – Verificata se, per il paziente e la data forniti, il valore dell'emoglobina è inferiore alla soglia minima.

- *"Emoglobina high"* – Verificata se, per il paziente e la data forniti, il valore dell'emoglobina è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a $12\,\frac{g}{dl}$ e $14\,\frac{g}{dl}$

Il sottogruppo dell'MCV (volume corpuscolare medio) comprende invece:

- *"MCV is ok"* - Verificata se e solo se, per la rispettiva data e il paziente dato, esistono MCV e soglie. Inoltre, il valore dell'MCV è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"MCV is null"* - Verificata se, per il paziente e la data forniti, il valore dell'MCV è pari a null.

- *"MCV has min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"MCV has bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"MCV low"* – Verificata se, per il paziente e la data forniti, il valore dell'MCV è inferiore alla soglia minima.

- *"MCV high"* – Verificata se, per il paziente e la data forniti, il valore dell'MCV è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a $80fl$ e $99fl$.

Tra le regole corrispondenti all'HT (ematocrito) si ha:

- *"HT is ok"*- Verificata se e solo se, per la rispettiva data e il paziente dato, esistono HT e soglie. Inoltre, il valore dell'HT in percentuale è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"HT is null"*- Verificata se, per il paziente e la data forniti, il valore dell'HT è pari a null.

- *"HT has min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"HT has bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"HT low"* – Verificata se, per il paziente e la data forniti, il valore dell'HT in percentuale è inferiore alla soglia minima.

- *"HT high"* – Verificata se, per il paziente e la data forniti, il valore dell'HT in percentuale è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a 14% e 35%.

Per quanto riguarda il sottogruppo dei leucociti:

- *"Leucociti are ok"* - Verificata se e solo se, per la rispettiva data e il paziente dato, esistono leucociti e soglie. Inoltre, il valore dei leucociti è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"Leucociti are null"* - Verificata se, per il paziente e la data forniti, il valore dei leucociti è pari a null.

- *"Leucociti have min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"Leucociti bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"Leucociti low"* – Verificata se, per il paziente e la data forniti, il valore dei leucociti è inferiore alla soglia minima.

- *"Leucociti high"* – Verificata se, per il paziente e la data forniti, il valore dei leucociti è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a $4{,}30 \cdot \frac{10^9}{l}$ e $10 \cdot \frac{10^9}{l}$

Passando alle piastrine:

- *"Piastrine are ok"* - Verificata se e solo se, per la rispettiva data e il paziente dato, esistono piastrine e soglie. Inoltre, il valore delle piastrine è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"Piastrine are null"* - Verificata se, per il paziente e la data forniti, il valore delle piastrine è pari a null.

- *"Piastrine have min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"Piastrine bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"Piastrine low"* – Verificata se, per il paziente e la data forniti, il valore delle piastrine è inferiore alla soglia minima.

- *"Piastrine high"* – Verificata se, per il paziente e la data forniti, il valore delle piastrine è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a $150 \cdot \frac{10^9}{l}$ e $400 \cdot \frac{10^9}{l}$

Relativamente al sottogruppo glicemia si hanno le seguenti regole:

- *"Glicemia is ok"* - Verificata se e solo se, per la rispettiva data e il paziente dato, esistono glicemia e soglie. Inoltre, il valore della glicemia è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"Glicemia is null"* - Verificata se, per il paziente e la data forniti, il valore della glicemia è pari a null.

- *"Glicemia has min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"Glicemia has bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"Glicemia low"* – Verificata se, per il paziente e la data forniti, il valore della glicemia è inferiore alla soglia minima.

- *"Glicemia high"* – Verificata se, per il paziente e la data forniti, il valore della glicemia è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a $60\,\frac{mg}{dl}$ e $99\,\frac{mg}{dl}$.

Per quanto riguarda l'urea si ha:

- *"Urea is ok"*- Verificata se e solo se, per la rispettiva data e il paziente dato, esistono urea e soglie. Inoltre, il valore dell'urea è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"Urea is null"*- Verificata se, per il paziente e la data forniti, il valore dell'urea è pari a null.

- *"Urea has min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"Urea has bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"Urea low"* – Verificata se, per il paziente e la data forniti, il valore dell'urea è inferiore alla soglia minima.

- *"Urea high"* – Verificata se, per il paziente e la data forniti, il valore dell'urea è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a $17\,\frac{mg}{dl}$ e $47\,\frac{mg}{dl}$.

Il sottogruppo successivo è relativo al sodio (Na). Troviamo:

- *"Na is ok"*- Verificata se e solo se, per la rispettiva data e il paziente dato, esistono sodio e soglie. Inoltre, il valore del sodio è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"Na is null"*- Verificata se, per il paziente e la data forniti, il valore del sodio è pari a null.

- *"Na has min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"Na has bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"Na low"* – Verificata se, per il paziente e la data forniti, il valore del sodio è inferiore alla soglia minima.

- *"Na high"* – Verificata se, per il paziente e la data forniti, il valore del sodio è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a $135\,\frac{mmol}{l}$ e $145\,\frac{mmol}{l}$.

Relativamente al potassio (K) si ha:

- *"K is ok"* - Verificata se e solo se, per la rispettiva data e il paziente dato, esistono potassio e soglie. Inoltre, il valore del potassio è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"K is null"* - Verificata se, per il paziente e la data forniti, il valore del potassio è pari a null.

- *"K has min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"K has bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"K low"* – Verificata se, per il paziente e la data forniti, il valore del potassio è inferiore alla soglia minima.

- *"K high"* – Verificata se, per il paziente e la data forniti, il valore del potassio è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a $3{,}4\,\frac{mmol}{l}$ e $4{,}8\,\frac{mmol}{l}$.

Se invece si considera il sottogruppo creatinina, si ha:

- *"Creatinina is ok"* - Verificata se e solo se, per la rispettiva data e il paziente dato, esistono creatinina e soglie. Inoltre, il valore della creatinina è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"Creatinina is null"* - Verificata se, per il paziente e la data forniti, il valore della creatinina è pari a null.

- *"Creatinina has min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"Creatinina has bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"Creatinina low"* – Verificata se, per il paziente e la data forniti, il valore della creatinina è inferiore alla soglia minima.

- *"Creatinina high"* – Verificata se, per il paziente e la data forniti, il valore della creatinina è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a $0{,}49\,\frac{mg}{dl}$ e $1{,}19\,\frac{mg}{dl}$.

Relativamente al colesterolo totale si ha:

- *"Colesterolo totale is ok"* - Verificata se e solo se, per la rispettiva data e il paziente dato, esistono colesterolo totale e soglie. Inoltre, il valore del colesterolo totale è compreso tra la soglia minima

e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"Colesterolo totale is null"* - Verificata se, per il paziente e la data forniti, il valore del colesterolo totale è pari a null.

- *"Colesterolo totale has min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"Colesterolo totale has bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"Colesterolo totale low"* – Verificata se, per il paziente e la data forniti, il valore del colesterolo totale è inferiore alla soglia minima.

- *"Colesterolo totale high"* – Verificata se, per il paziente e la data forniti, il valore del colesterolo totale è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a $null\ (valore\ non\ presente)$ e $200\frac{mg}{dl}$.

Per quanto riguarda invece il sottogruppo del colesterolo HDL, si hanno le seguenti regole:

- *"Colesterolo HDL is ok"* - Verificata se e solo se, per la rispettiva data e il paziente dato, esistono colesterolo HDL e soglie. Inoltre, il valore del colesterolo HDL è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"Colesterolo HDL is null"* - Verificata se, per il paziente e la data forniti, il valore del colesterolo *HDL* è pari a null.

- *"Colesterolo HDL has min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"Colesterolo HDL has bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"Colesterolo HDL low"* – Verificata se, per il paziente e la data forniti, il valore del colesterolo HDL è inferiore alla soglia minima.

- *"Colesterolo HDL high"* – Verificata se, per il paziente e la data forniti, il valore del colesterolo HDL è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a $50\frac{mg}{dl}$ e $null\ (valore\ non\ presente)$.

Il penultimo sottogruppo di regole a parametro singolo è relativo ai trigliceridi. Si hanno le seguenti regole:

- *"Trigliceridi are ok"* - Verificata se e solo se, per la rispettiva data e il paziente dato, esistono trigliceridi e soglie. Inoltre, il valore dei trigliceridi è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"Trigliceridi are null"* - Verificata se, per il paziente e la data forniti, il valore dei trigliceridi è pari a null.

- *"Trigliceridi have min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"Trigliceridi bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"Trigliceridi low"* – Verificata se, per il paziente e la data forniti, il valore dei trigliceridi è inferiore alla soglia minima.

- *"Trigliceridi high"*– Verificata se, per il paziente e la data forniti, il valore dei trigliceridi è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a $null$ $(valore\ non\ presente)$ e $150\frac{mg}{dl}$.

Tra le regole relative alla proteina C reattiva (PCR) si ha:

- *"PCR is ok"* - Verificata se e solo se, per la rispettiva data e il paziente dato, esistono PCR e soglie. Inoltre, il valore del PCR è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"PCR is null"* - Verificata se, per il paziente e la data forniti, il valore del PCR è pari a null.

- *"PCR has min and max thresholds null"*– Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"PCR has bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"PCR low"* – Verificata se, per il paziente e la data forniti, il valore del PCR è inferiore alla soglia minima.

- *"PCR high"* – Verificata se, per il paziente e la data forniti, il valore del colesterolo totale è superiore alla soglia massima.

I valori di soglia minima e massima convenzionali sono rispettivamente pari a $null$ $(valore\ non\ presente)$ e $5\frac{mg}{l}$.

Definite le regole a parametro singolo dettate dalle soglie convenzionali, si hanno ulteriori regole a parametri multipli.

Innanzitutto, vi sono le regole relative al rapporto $\frac{Colesterolo\ Totale}{Colesterolo\ HDL}$:

- *"Colesterolo totale / HDL basso"* – verificata se e solo se il rapporto tra colesterolo totale e HDL è minore di 5. Rappresenta la regola associata alla recommendation positiva, tra quelle del sottogruppo.

- *"Colesterolo totale / HDL moderato"* – verificata se e solo se il rapporto tra colesterolo totale e HDL è compreso tra 5 e 10 (uguaglianza inclusa).

- *"Colesterolo totale / HDL elevato"* - verificata se e solo se il rapporto tra colesterolo totale e HDL è maggiore di 10.

Le soglie (ossia i risultati del rapporto) sono dettate da una versione semplificata dell'indice di rischio cardiovascolare (IRC), definito dall'Istituto Superiore di Sanità. [24] Ovviamente le recommendations non rappresentano un'indicazione in grado di sostituire il parere medico ma servono semplicemente come consigli per migliorare il benessere e come tali devono essere considerate.

Un altro sottogruppo di regole a parametri multipli riguarda il rapporto $\frac{Urea}{Creatinina}$:

- *"Rapporto Urea / Creatinina basso"* – verificata se e solo se il rapporto tra urea e creatinina è minore di 40.

- *"Rapporto Urea / Creatinina normale"* – verificata se e solo se il rapporto tra urea e creatinina è compreso tra 40 e 100 (uguaglianza inclusa). Rappresenta la regola associata alla recommendation positiva, tra quelle del sottogruppo.

- *"Rapporto Urea / Creatinina elevato"* - verificata se e solo se il rapporto tra urea e creatinina è maggiore di 100.

Anche in questo caso i valori che definiscono il rapporto sono noti dalla letteratura medica e riportati da diverse fonti d'informazione. [25]

L'ultimo sottogruppo da presentare riguarda il rapporto $\frac{Trigliceridi}{Colesterolo\ HDL}$:

- *"Trigliceridi /Colesterolo HDL ideale"*– verificata se e solo se il rapporto tra trigliceridi e colesterolo HDL è minore o uguale a 2. Rappresenta la regola associata alla recommendation positiva, tra quelle del sottogruppo.

- *"Trigliceridi /Colesterolo HDL elevato"* – verificata se e solo se il rapporto tra trigliceridi e colesterolo HDL è maggiore di 2 e minore o uguale a 6.

- *"Trigliceridi /Colesterolo HDL molto elevato"* - verificata se e solo se il rapporto tra trigliceridi e colesterolo HDL è maggiore di 6.

Come nei precedenti sottogruppi, anche in questo caso si hanno numerose testimonianze in letteratura che motivano la scelta di determinati valori come soglie che permettono di discriminare tra un rapporto ideale e non ideale. [26]

5.1.3 Cibo

Il gruppo cibo si articola nei sottogruppi individuati dalla tabella 2 del capitolo 3, formati da insiemi di regole a parametro singolo: calorie immesse, carboidrati, proteine, grassi, grassi saturi, grassi polinsaturi, grassi monoinsaturi, grassi trans, colesterolo, sodio, potassio, fibre, zuccheri, zuccheri aggiunti, vitamina A, vitamina C, vitamina D, calcio e ferro.

Si hanno inoltre dei sottogruppi formati da regole a parametri multipli: rapporto sodio/potassio, proporzione tra grassi saturi, monoinsaturi e polinsaturi ed infine proporzione tra carboidrati, proteine e grassi.

Vi è anche un sottogruppo di regole a macroarea multipla: bilancio tra calorie immesse e calorie bruciate riguardante l'alimentazione e il movimento.

Poiché, come è stato descritto nel capitolo 3, i pasti della macroarea alimentazione vengono divisi in differenti tipologie (colazione, pranzo, cena ecc.) è necessario effettuare un'aggregazione dei dati prima di effettuarne l'analisi. Infatti, l'applicazione delle regole avverrà con una granularità minima giornaliera. Di conseguenza, vengono considerati l'insieme dei valori nutrizionali su base giornaliera, settimanale o mensile (le ultime due solo per le regole a parametro singolo) e non sulla base della tipologia di pasto. Tale scelta è motivata dal fatto che i dati sul singolo pasto non risultano rilevanti ai fini dell'analisi, come è possibile evincere nella totalità delle applicazioni di settore che offrono ad esempio il conteggio di calorie immesse. Qualora, all'atto dell'aggregazione, alcuni parametri assumano il valore null, li si considera pari a zero per semplicità.

Si procede con l'illustrazione delle regole. Il primo sottogruppo oggetto di analisi è quello relativo alle calorie immesse:

- *"Calories in are ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di calorie immesse e soglia esistono. Inoltre, la quantità di calorie immesse è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Calories in are null"* – Regola verificata se la quantità di calorie immesse non è presente, ossia assume il valore null.

- *"Calories in have null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Calories in high"* – Regola verificata se il quantitativo di calorie immesse dal paziente nella data considerata è superiore alla soglia.

La soglia standard considerata è pari a tremila kilocalorie, riferite al fabbisogno calorico medio di un uomo in età adulta. [27] Ovviamente per le donne, o per assistiti con esigenze particolari, tale soglia dovrà essere rivista dai medici in quanto nell'applicazione si è considerato il "caso peggiore".

Il sottogruppo successivo è relativo ai carboidrati; si ha:

- *"Carbohydrates are ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di carboidrati e soglia esistono. Inoltre, la quantità di carboidrati assunti è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Carbohydrates are null"* – Regola verificata se la quantità di carboidrati assunti non è presente, ossia assume il valore null.

- *"Carbohydrates have null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Carbohydrates high"* – Regola verificata se il quantitativo di carboidrati assunti dal paziente nella data considerata è superiore alla soglia.

La soglia standard di riferimento in questo caso è fissata a trecento grammi di carboidrati, riferita sempre al consumo medio di un uomo in età adulta non del tutto sedentario. [28]

Per quanto riguarda invece le proteine, si ha:

- *"Proteins are ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di proteine e soglia esistono. Inoltre, la quantità di proteine assunte è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Proteins are null"* – Regola verificata se la quantità di proteine assunte non è presente, ossia assume il valore null.

- *"Proteins have null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Proteins high"* – Regola verificata se il quantitativo di proteine assunte dal paziente nella data considerata è superiore alla soglia.

Per le proteine, si sceglie un valore di soglia pari a 150 grammi. [29] Valgono le considerazioni espresse nelle sezioni calorie immesse e carboidrati.

Il sottogruppo successivo è quello dei grassi composto dalle seguenti regole:

- *"Fats are ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di grassi e soglia esistono. Inoltre, la quantità di grassi assunti è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Fats are null"* – Regola verificata se la quantità di grassi assunti non è presente, ossia assume il valore null.

- *"Fats have null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Fats high"* – Regola verificata se il quantitativo di grassi assunti dal paziente nella data considerata è superiore alla soglia.

Il valore di soglia è fissato a 75 grammi. [30]

Si hanno poi le regole relative al sottogruppo grassi saturi:

- *"Saturated Fats are ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di grassi saturi e soglia esistono. Inoltre, la quantità di grassi saturi assunti è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Saturated Fats are null"* – Regola verificata se la quantità di grassi saturi assunti non è presente, ossia assume il valore null.

- *"Saturated Fats have null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Saturated Fats high"* – Regola verificata se il quantitativo di grassi saturi assunti dal paziente nella data considerata è superiore alla soglia.

Il valore di soglia è pari a 18g, ossia circa il 25% dell'apporto totale di grassi. [31]

Per quanto riguarda invece il sottogruppo relativo ai grassi polinsaturi:

- *"Polyunsaturated Fats are ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di grassi polinsaturi e soglia esistono. Inoltre, la quantità di grassi polinsaturi assunti è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Polyunsaturated Fats are null"* – Regola verificata se la quantità di grassi polinsaturi assunti non è presente, ossia assume il valore null.

- *"Polyunsaturated Fats have null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Polyunsaturated Fats high"* – Regola verificata se il quantitativo di grassi polinsaturi assunti dal paziente nella data considerata è superiore alla soglia.

In questo caso, la soglia da considerare è pari al 20% dell'apporto totale di grassi, quindi 15g. [31]

Un altro sottogruppo relativo ai grassi è quello dei grassi monoinsaturi. Le regole che lo compongono sono:

- *"Monounsaturated Fats are ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di grassi monoinsaturi e soglia esistono. Inoltre, la quantità di grassi monoinsaturi assunti è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Monounsaturated Fats are null"* – Regola verificata se la quantità di grassi monoinsaturi assunti non è presente, ossia assume il valore null.

- *"Monounsaturated Fats have null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Monounsaturated Fats high"* – Regola verificata se il quantitativo di grassi monoinsaturi assunti dal paziente nella data considerata è superiore alla soglia.

L'apporto che definisce la soglia è pari al 55%, ossia circa 40g. [31]

L'ultimo sottogruppo riguardante i grassi è quello dei grassi trans:

- *"Trans Fats are ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di grassi trans e soglia esistono. Inoltre, la quantità di grassi trans assunti è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Trans Fats are null"* – Regola verificata se la quantità di grassi trans assunti non è presente, ossia assume il valore null.

- *"Trans Fats have null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Trans Fats high"* – Regola verificata se il quantitativo di grassi trans assunti dal paziente nella data considerata è superiore alla soglia.

Il limite ai grassi trans è fissato sui due grammi, secondo le disposizioni dell'OMS. [32]

Per quanto riguarda invece il colesterolo, si ha:

- *"Cholesterol is ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di colesterolo e soglia esistono. Inoltre, la quantità di colesterolo assunta è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Cholesterol is null"* – Regola verificata se la quantità di colesterolo assunta non è presente, ossia assume il valore null.

- *"Cholesterol has null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Cholesterol high"* – Regola verificata se il quantitativo di colesterolo assunto dal paziente nella data considerata è superiore alla soglia.

Il quantitativo che definisce la soglia è pari a 300mg, individuata da uno studio medico del 2013. [33]

Spostando l'attenzione sui minerali, il primo sottogruppo è quello del sodio. Le regole che lo compongono sono:

- *"Sodium is ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di sodio e soglia esistono. Inoltre, la quantità di sodio assunta è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Sodium is null"* – Regola verificata se la quantità di sodio assunta non è presente, ossia assume il valore null.

- *"Sodium has null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Sodium high"* – Regola verificata se il quantitativo di sodio assunto dal paziente nella data considerata è superiore alla soglia.

La soglia relativa al sodio, secondo le raccomandazioni di ISS e OMS, è pari a 2000 mg. [34]

Successivamente, si hanno le regole associate al potassio:

- *"Potassium is ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di potassio e soglia esistono. Inoltre, la quantità di potassio assunta è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Potassium is null"* – Regola verificata se la quantità di potassio assunta non è presente, ossia assume il valore null.

- *"Potassium has null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Potassium high"* – Regola verificata se il quantitativo di potassio assunto dal paziente nella data considerata è superiore alla soglia.

Per quanto riguarda la soglia, raccomandata da ISS e OMS, assume il valore di 3510 mg. [35]

Seguono le regole relative al calcio:

- *"Calcium is ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di calcio e soglia esistono. Inoltre, la quantità di calcio assunta è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Calcium is null"* – Regola verificata se la quantità di calcio assunta non è presente, ossia assume il valore null.

- *"Calcium has null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Calcium high"* – Regola verificata se il quantitativo di calcio assunto dal paziente nella data considerata è superiore alla soglia.

La soglia standard ha un valore di 1000mg, indicato dall'ISS come valore di assunzione ideale per gli anziani. [36]

L'ultimo sottogruppo dei minerali riguarda le regole relative al ferro:

- *"Iron is ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di ferro e soglia esistono. Inoltre, la quantità di ferro assunta è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Iron is null"* – Regola verificata se la quantità di ferro assunta non è presente, ossia assume il valore null.

- *"Iron has null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Iron high"* – Regola verificata se il quantitativo di ferro assunto dal paziente nella data considerata è superiore alla soglia.

La soglia relativa al ferro è pari a 14mg, in linea con il valore di riferimento europeo. [37]

Un altro sottogruppo è costituito dalle fibre, si ha:

- *"Fibers are ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di fibre e soglia esistono. Inoltre, la quantità di fibre assunte è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Fibers are null"* – Regola verificata se la quantità di fibre assunte non è presente, ossia assume il valore null.

- *"Fibers have null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Fibers high"* – Regola verificata se il quantitativo di fibre assunto dal paziente nella data considerata è superiore alla soglia.

La soglia di fibre considerata è pari a 35g, come raccomandato dai grandi istituti di sanità (FDA, AIRC, OMS, LARN). [38]

Segue il sottogruppo degli zuccheri così composto:

- *"Sugars are ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di zuccheri e soglia esistono. Inoltre, la quantità di zuccheri assunti è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Sugars are null"* – Regola verificata se la quantità di zuccheri assunti non è presente, ossia assume il valore null.

- *"Sugars have null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Sugars high"* – Regola verificata se il quantitativo di zuccheri assunto dal paziente nella data considerata è superiore alla soglia.

Per quanto riguarda la soglia, il valore è pari a circa 70g, come determinato dall'OMS. [39]

Si ha anche l'insieme di regole relative agli zuccheri aggiunti:

- *"Added Sugars are ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di zuccheri aggiunti e soglia esistono. Inoltre, la quantità di zuccheri aggiunti è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Added Sugars are null"* – Regola verificata se la quantità di zuccheri aggiunti non è presente, ossia assume il valore null.

- *"Added Sugars have null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Added Sugars high"* – Regola verificata se il quantitativo di zuccheri aggiunti assunto dal paziente nella data considerata è superiore alla soglia.

La soglia in questo caso è di 50g, secondo le linee guida LARN. [40]

Per quanto riguarda le regole a parametro singolo, si hanno infine i sottogruppi relativi alle vitamine. Il primo è quello della Vitamina A, così composto:

- *"Vitamin A is ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di vitamina A e soglia esistono. Inoltre, la quantità di vitamina A assunta è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Vitamin A is null"* – Regola verificata se la quantità di vitamina A assunta non è presente, ossia assume il valore null.

- *"Vitamin A has null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Vitamin A high"* – Regola verificata se il quantitativo di vitamina A assunto dal paziente nella data considerata è superiore alla soglia.

La soglia considerata in tali regole è pari a 950µg, pari alla quantità necessaria in fase di allattamento (caso peggiore). [41]

Segue la Vitamina C con le seguenti regole:

- *"Vitamin C is ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di vitamina C e soglia esistono. Inoltre, la quantità di vitamina C assunta è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Vitamin C is null"* – Regola verificata se la quantità di vitamina C assunta non è presente, ossia assume il valore null.

- *"Vitamin C has null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Vitamin C high"* – Regola verificata se il quantitativo di vitamina C assunto dal paziente nella data considerata è superiore alla soglia.

Per quanto riguarda la soglia, dalla letteratura medica è noto che assume il valore di 90mg. [42]

Infine, si ha la vitamina D, comprendente:

- *"Vitamin D is ok"* – Regola verificata se e solo se, per il paziente e la data considerati, i valori di vitamina D e soglia esistono. Inoltre, la quantità di vitamina D assunta è inferiore o uguale alla soglia. Corrisponde alla recommendation positiva del sottogruppo.

- *"Vitamin D is null"* – Regola verificata se la quantità di vitamina D assunta non è presente, ossia assume il valore null.

- *"Vitamin D has null threshold"* – Regola verificata se la soglia di riferimento non è definita, ossia assume il valore null.

- *"Vitamin D high"* – Regola verificata se il quantitativo di vitamina D assunto dal paziente nella data considerata è superiore alla soglia.

La soglia relativa a tale sottogruppo è pari a 25µg, pari al valore massimo assumibili in presenza di fattori di rischio o deficit. [43]

Per quanto riguarda invece i sottogruppi formati da regole a parametri multipli, come accennato in precedenza si ha il rapporto $\frac{Sodio}{Potassio}$ così composto:

- *"Sodium / potassium ratio is ok"* – Regola verificata se e solo se, per il paziente e la data in questione, si ha che il rapporto sodio / potassio è minore di 1. Corrisponde alla recommendation positiva.

- *"Sodium / potassium ratio is high"* – Regola verificata se il rapporto sodio / potassio è maggiore o uguale a 1.

Il valore di riferimento del rapporto, ossia uno, deriva da uno studio condotto dal ministero della salute. [44]

Un ulteriore sottogruppo di regole è quello relativo alla proporzione tra grassi saturi, polinsaturi e monoinsaturi:

- *"Fat proportion is ok"* - Regola verificata se e solo se, per il paziente e la data in questione, si ha che la proporzione tra i grassi corrisponde al 25% di grassi saturi, il 55% di grassi monoinsaturi e il 20% di grassi polinsaturi . Corrisponde alla recommendation positiva.

- *"Fat proportion has high quantity of polyunsaturated fats"* – Verificata quando la quantità di grassi polinsaturi eccede il 20% dei grassi totali.

- *"Fat proportion has high quantity of saturated fats"* – Verificata quando la quantità di grassi saturi eccede il 25% dei grassi totali.

- *"Fat proportion has high quantity of monounsaturated fats"* – Verificata quando la quantità di grassi monoinsaturi eccede il 55% dei grassi totali.

- *"Fat proportion has high quantity of saturated and polyunsaturated fats"* – Verificata quando la quantità di grassi saturi e polinsaturi eccedono rispettivamente il 25% ed il 20% dei grassi totali.

- *"Fat proportion has high quantity of saturated and monounsaturated fats"* - Verificata quando la quantità di grassi saturi e monoinsaturi eccedono rispettivamente il 25% ed il 55% dei grassi totali.

- *"Fat proportion has high quantity of monounsaturated and polyunsaturated fats"* - Verificata quando la quantità di grassi monoinsaturi e polinsaturi eccedono rispettivamente il 55% ed il 20% dei grassi totali.

- *"Fat proportion has high quantity of saturated, monounsaturated and polyunsaturated fats"* - Verificata quando la quantità di grassi saturi, monoinsaturi e polinsaturi eccedono rispettivamente il 25%, il 55% ed il 20% dei grassi totali.

Le percentuali della proporzione tra i grassi sono note dalla letteratura. [45]

Per quanto riguarda le regole a parametri multipli, si ha infine il sottogruppo relativo alla proporzione tra carboidrati proteine e grassi così composto:

- *"Carbs, proteins and fats proportion is ok"*- Regola verificata se e solo se, per il paziente e la data in questione, si ha che la proporzione tra carboidrati, proteine e grassi è rispettivamente del 50%, 20% e 30% del totale. Corrisponde alla recommendations positiva.

- *"Carbs, proteins and fats proportion has high quantity of carbohydrates"* – Verificata se la percentuale di carboidrati, rispetto al totale, è superiore al 50%

- *"Carbs, proteins and fats proportion has high quantity of proteins"* – Verificata se la percentuale di proteine, rispetto al totale, è superiore al 20%

- *"Carbs, proteins and fats proportion has high quantity of fats"* – Verificata se la percentuale di carboidrati, rispetto al totale, è superiore al 30%

- *"Carbs, proteins and fats proportion has high quantity of carbohydrates and proteins"* – Verificata se la percentuale di carboidrati e proteine, rispetto al totale, è superiore rispettivamente al 50% ed al 20%

- *"Carbs, proteins and fats proportion has high quantity of carbohydrates and fats"* – Verificata se la percentuale di carboidrati e grassi, rispetto al totale, è superiore rispettivamente al 50% ed al 30%

- *"Carbs, proteins and fats proportion has high quantity of proteins and fats"* – Verificata se la percentuale di proteine e grassi, rispetto al totale, è superiore rispettivamente al 20% ed al 30%

- *"Carbs, proteins and fats proportion has high quantity of carbohydrates, proteins and fats"* – Verificata se la percentuale di carboidrati, proteine e grassi, rispetto al totale, è superiore rispettivamente al 50%, al 20% ed al 30%

Anche in questo caso le soglie della proporzione sono determinate dalla letteratura e da uno studio della società italiana di nutrizione umana (SINU) [46].

L'ultimo sottogruppo di regole trattato in questo paragrafo riguarda il bilancio tra calorie immesse e calorie bruciate, relativo alle macroaree alimentazione e movimento. Si hanno le seguenti regole:

- *"Calories in more than calories out"* – Regola verificata se e solo se, per il paziente e la data considerati, il quantitativo di calorie immesse è maggiore di quello delle calorie bruciate.

- *"Calories in less than calories out"* – Regola verificata se e solo se, per il paziente e la data considerati, il quantitativo di calorie immesse è minore di quello delle calorie bruciate.

- *"Calories in equal to calories out"* – Regola verificata se e solo se, per il paziente e la data considerati, il quantitativo di calorie immesse è esattamente uguale a quello delle calorie bruciate.

In questo caso, tutte e tre le regole sono associate a delle recommendations di tipo positivo, che hanno semplicemente un fine informativo, ossia mettere a conoscenza il medico ed il paziente del bilancio calorico. Tale scelta è motivata dal fatto che il bilancio calorico ottimale dipende dal tipo di dieta che si segue [47], e non esiste quindi un caso negativo assoluto.

5.1.4 Frequenza cardiaca

Il gruppo frequenza cardiaca contiene le sole regole relative alla frequenza a riposo in quanto tale parametro è attualmente l'unico di interesse in HealthApp. Viene dedicato comunque un file apposito offrendo la possibilità di estendere tale gruppo in sviluppi futuri. Relativamente alla frequenza a riposo si hanno le seguenti regole, tutte a parametro singolo:

- *"Rest Hr is ok"* - Verificata se e solo se, per la rispettiva data e il paziente dato, esistono battito a riposo e soglie. Inoltre, il valore del battito a riposo è compreso tra la soglia minima e quella massima (uguaglianza inclusa). Tale regola corrisponde ad una recommendation di tipo positivo.

- *"Rest Hr is null"* - Verificata se, per il paziente e la data forniti, il valore del battito a riposo è pari a null.

- *"Rest Hr has min and max thresholds null"* – Verificata se, per il paziente e la data forniti, entrambe le soglie sono pari a null.

- *"Rest Hr has bad thresholds"* – Verificata se, per il paziente e la data forniti, la soglia minima è maggiore della soglia massima.

- *"Rest Hr low"* – Verificata se, per il paziente e la data forniti, il valore del battito a riposo è inferiore alla soglia minima.

- *"Rest Hr high"* – Verificata se, per il paziente e la data forniti, il valore del battito a riposo è superiore alla soglia massima.

Le soglie minima e massima individuate corrispondono rispettivamente a 55 e 100 battiti. Esse sono definite dalla maggior parte della letteratura e associazioni mediche. [48]

5.1.5 Sonno

In tale gruppo, similarmente a quanto visto con la frequenza cardiaca, vi è un solo set di regole, ossia quelle del tempo di sonno. Valgono le considerazioni viste in precedenza; attualmente il tempo di sonno è l'unico parametro d'interesse ma si mantiene un file separato per garantire una facile manutenibilità e supportare efficientemente eventuali estensioni future.

Tra le regole si ha:

- *"Sleep time is ok"* – Verificata se e solo se, per la rispettiva data e il paziente dato, esistono tempo di sonno e soglia. Inoltre, il valore del tempo di sonno è maggiore o uguale a quello della soglia. Tale regola corrisponde ad una recommendation di tipo positivo.

- *"Sleep time is null"* – Verificata se, per il paziente e la data forniti, il valore del tempo di sonno è pari a null.

- *"Sleep time has null threshold"* – Verificata se, per il paziente e la data forniti, la soglia è pari a null.

- *"Sleep time low"* - Verificata se, per il paziente e la data forniti, il valore del tempo di sonno è inferiore alla soglia.

La soglia determinata per il tempo di sonno è pari a otto ore, corrispondenti a 28800000ms (il tempo di sonno viene registrato nella base dati in ms). La durata raccomandata [49] è infatti di circa otto ore nei soggetti adulti (i bambini e neonati sono esclusi in quanto ad HealthApp si possono registrare solamente soggetti maggiorenni).

5.1.6 Peso

L'ultimo gruppo di regole riguarda il peso. Si articola in due sottogruppi: massa e BMI (indice di massa corporea).

Tra le regole della massa si ha:

- *"Weight is ok"* – Verificata se e solo se, per la rispettiva data e il paziente dato, esistono massa e soglia. Inoltre, il valore della massa è minore o uguale a quello della soglia. Tale regola corrisponde ad una recommendation di tipo positivo.

- *"Weight is null"* – Verificata se, per il paziente e la data forniti, il valore della massa è pari a null.

- *"Weight has null threshold"* – Verificata se, per il paziente e la data forniti, la soglia è pari a null.

- *"Weight high"* - Verificata se, per il paziente e la data forniti, il valore della massa è superiore alla soglia.

La soglia massa individuata è pari a 100kg ed è puramente indicativa [50] in quanto non rappresenta un valore piuttosto significativo. È opportuno che tale soglia sia definita dal medico.

Infine, per quanto riguarda il sottogruppo del BMI, si ha:

- *"BMI is ok"* – Verificata se e solo se, per la rispettiva data e il paziente dato, esistono BMI e soglia. Inoltre, il valore del BMI è minore o uguale a quello della soglia. Tale regola corrisponde ad una recommendation di tipo positivo.

- *"BMI is null"* – Verificata se, per il paziente e la data forniti, il valore del BMI è pari a null.
- *"BMI has null threshold"* – Verificata se, per il paziente e la data forniti, la soglia è pari a null.
- *"BMI high"* - Verificata se, per il paziente e la data forniti, il valore del BMI è superiore alla soglia.

La soglia di BMI è fissata a 30, corrispondente valore minimo dell'intervallo riferito all'obesità di I grado. [51]

Conclusa l'illustrazione delle regole, il paragrafo successivo riguarderà la definizione delle API che permetteranno ai client di innescare l'esecuzione delle regole e di consultare i report prodotti.

5.2 Definizione API

Le API o endpoint sono necessarie a gestire e personalizzare l'esecuzione delle regole. Tali endpoint vengono classificati in tre macrogruppi distinti:

- Soglie recommendations: Permettono di effettuare le operazioni CRUD sulle soglie delle recommendations associate a ciascun paziente.
- Esecuzione recommendations: Permettono di innescare l'esecuzione delle regole.
- Consultazione report automaticamente prodotti: Permettono di consultare i report prodotti periodicamente dal sistema (approfonditi nel paragrafo successivo).

Relativamente alle API per manipolare le soglie per ogni paziente, esse sono chiamabili solamente da un account di tipo medico o amministratore. Tale scelta è motivata dal fatto che il paziente non è in grado di determinare correttamente le soglie opportune, non avendo nella maggioranza dei casi conoscenze di settore medico. I metodi supportati sono GET, POST, PUT e DELETE. Entrando nel dettaglio:

- *GET /api/patients/{patientID}/recommendations/thresholds/hr* – Ritorna un JSON contenente un array con un singolo elemento. Esso contiene un identificativo, le informazioni sul paziente, identificato mediante la variabile patientID, a cui le soglie sono associate e i valori di soglia minima e massima per il battito a riposo.
- *POST /api/patients/{patientID}/recommendations/thresholds/hr* – Riceve un JSON contenente i valori di soglia minima e massima per il battito a riposo. Provvede a salvare tali informazioni nella base dati associandole al paziente identificato mediante il contenuto della variabile patientID.
- *PUT /api/patients/{patientID}/recommendations/thresholds/hr/{idHrP}* - Riceve un JSON contenente i valori di soglia minima e massima per il battito a riposo. Provvede ad aggiornare il record della base dati identificato mediante idHrP, inserendo le nuove soglie e associandole al paziente identificato mediante il contenuto della variabile patientID.

- *DELETE /api/patients/{patientID}/recommendations/thresholds/hr/{idHrP}* – Provvede ad eliminare le soglie relative al battito a riposo, identificate nella base dati con idHrP e associate al paziente identificato mediante patientID.

- *GET /api/patients/{patientID}/recommendations/thresholds/activity*– Ritorna un JSON contenente un array con un singolo elemento. Esso contiene un identificativo, le informazioni sul paziente, identificato mediante la variabile patientID, a cui le soglie sono associate e i valori di soglia per passi e calorie consumate.

- *POST /api/patients/{patientID}/recommendations/thresholds/activity*– Riceve un JSON contenente i valori di soglia per passi e calorie consumate. Provvede a salvare tali informazioni nella base dati associandole al paziente identificato mediante il contenuto della variabile patientID.

- *PUT /api/patients/{patientID}/recommendations/thresholds/activity/{idAP}* - Riceve un JSON contenente i valori di soglia per passi e calorie consumate. Provvede ad aggiornare il record della base dati identificato mediante idAP, inserendo le nuove soglie e associandole al paziente identificato mediante il contenuto della variabile patientID.

- *DELETE /api/patients/{patientID}/recommendations/thresholds/activity/{idAP}* – Provvede ad eliminare le soglie relative a passi e calorie consumate, identificate nella base dati con idAP e associate al paziente identificato mediante patientID.

- *GET /api/patients/{patientID}/recommendations/thresholds/sleep*– Ritorna un JSON contenente un array con un singolo elemento. Esso contiene un identificativo, le informazioni sul paziente, identificato mediante la variabile patientID, a cui la soglia è associata e il valore della soglia per il tempo di sonno.

- *POST /api/patients/{patientID}/recommendations/thresholds/sleep*– Riceve un JSON contenente il valore di soglia per il tempo di sonno. Provvede a salvare tale informazione nella base dati associandola al paziente identificato mediante il contenuto della variabile patientID.

- *PUT /api/patients/{patientID}/recommendations/thresholds/sleep/{idSP}* - Riceve un JSON contenente il valore di soglia per il tempo di sonno. Provvede ad aggiornare il record della base dati identificato mediante idSP, inserendo la nuova soglia e associandola al paziente identificato mediante il contenuto della variabile patientID.

- *DELETE /api/patients/{patientID}/recommendations/thresholds/sleep/{idSP}* – Provvede ad eliminare la soglia relativa al tempo di sonno, identificata nella base dati con idSP e associata al paziente identificato mediante patientID.

- *GET /api/patients/{patientID}/recommendations/thresholds/weight* – Ritorna un JSON contenente un array con un singolo elemento. Esso contiene un identificativo, le informazioni sul paziente, identificato mediante la variabile patientID, a cui le soglie sono associate e i valori di soglia per massa e BMI.

- *POST /api/patients/{patientID}/recommendations/thresholds/weight* – Riceve un JSON contenente i valori di soglia per massa e BMI. Provvede a salvare tali informazioni nella base dati associandole al paziente identificato mediante il contenuto della variabile patientID.

- *PUT /api/patients/{patientID}/recommendations/thresholds/weight/{idWP}* - Riceve un JSON contenente i valori di soglia per massa e BMI. Provvede ad aggiornare il record della base dati identificato mediante idWP, inserendo le nuove soglie e associandole al paziente identificato mediante il contenuto della variabile patientID.

- *DELETE /api/patients/{patientID}/recommendations/thresholds/weight/{idWP}* – Provvede ad eliminare le soglie relative a massa e BMI, identificate nella base dati con idWP e associate al paziente identificato mediante patientID.

- *GET /api/patients/{patientID}/recommendations/thresholds/food*– Ritorna un JSON contenente un array con un singolo elemento. Esso contiene un identificativo, le informazioni sul paziente, identificato mediante la variabile patientID, a cui le soglie sono associate e i valori di soglia per calorie immesse, carboidrati, proteine, grassi, grassi saturi, grassi monoinsaturi, grassi polinsaturi, grassi trans, colesterolo, sodio, potassio, fibre, zuccheri, zuccheri aggiunti, vitamina A, vitamina C, vitamina D, calcio e ferro.

- *POST /api/patients/{patientID}/recommendations/thresholds/food* – Riceve un JSON contenente i valori di soglia per calorie immesse, carboidrati, proteine, grassi, grassi saturi, grassi monoinsaturi, grassi polinsaturi, grassi trans, colesterolo, sodio, potassio, fibre, zuccheri, zuccheri aggiunti, vitamina A, vitamina C, vitamina D, calcio e ferro. Provvede a salvare tali informazioni nella base dati associandole al paziente identificato mediante il contenuto della variabile patientID.

- *PUT /api/patients/{patientID}/recommendations/thresholds/food/{idFP}*– Riceve un JSON contenente i valori di soglia per calorie immesse, carboidrati, proteine, grassi, grassi saturi, grassi monoinsaturi, grassi polinsaturi, grassi trans, colesterolo, sodio, potassio, fibre, zuccheri, zuccheri aggiunti, vitamina A, vitamina C, vitamina D, calcio e ferro. Provvede ad aggiornare il record della base dati identificato mediante idFP, inserendo le nuove soglie e associandole al paziente identificato mediante il contenuto della variabile patientID.

- *DELETE /api/patients/{patientID}/recommendations/thresholds/food/{idFP}*– Provvede ad eliminare le soglie relative a calorie immesse, carboidrati, proteine, grassi, grassi saturi, grassi monoinsaturi, grassi polinsaturi, grassi trans, colesterolo, sodio, potassio, fibre, zuccheri, zuccheri aggiunti, vitamina A, vitamina C, vitamina D, calcio e ferro, identificate nella base dati con idFP e associate al paziente identificato mediante patientID.

- *GET /api/patients/{patientID}/recommendations/thresholds/bloodAnalysis*– Ritorna un JSON contenente un array con un singolo elemento. Esso contiene un identificativo, le informazioni sul paziente, identificato mediante la variabile patientID, a cui le soglie sono associate e i valori di soglia, minima e massima, per eritrociti, emoglobina, MCV, HT, leucociti, piastrine, glicemia, urea, sodio, potassio, creatinina, colesterolo totale, colesterolo HDL, trigliceridi e PCR, qualora disponibili.

- *POST /api/patients/{patientID}/recommendations/thresholds/bloodAnalysis*– Riceve un JSON contenente i valori di soglia, minima e massima, per eritrociti, emoglobina, MCV, HT, leucociti, piastrine, glicemia, urea, sodio, potassio, creatinina, colesterolo totale, colesterolo HDL, trigliceridi e PCR, qualora disponibili. Provvede a salvare tali informazioni nella base dati associandole al paziente identificato mediante il contenuto della variabile patientID.

- *PUT /api/patients/{patientID}/recommendations/thresholds/bloodAnalysis/{idBAP}*– Riceve un JSON contenente i valori di soglia, minima e massima, per eritrociti, emoglobina, MCV, HT, leucociti, piastrine, glicemia, urea, sodio, potassio, creatinina, colesterolo totale, colesterolo HDL, trigliceridi e PCR, qualora disponibili. Provvede ad aggiornare il record della base dati identificato mediante idBAP, inserendo le nuove soglie e associandole al paziente identificato mediante il contenuto della variabile patientID.

- *DELETE /api/patients/{patientID}/recommendations/thresholds/bloodAnalysis/{idBAP}*– Provvede ad eliminare le soglie relative a eritrociti, emoglobina, MCV, HT, leucociti, piastrine, glicemia, urea, sodio, potassio, creatinina, colesterolo totale, colesterolo HDL, trigliceridi e PCR, qualora disponibili, identificate nella base dati con idBAP e associate al paziente identificato mediante patientID.

Mediante la personalizzazione delle soglie di ciascun paziente è possibile sfruttare nella maniera più efficace possibile il recommender system, evitando la produzione di recommendations insensate o poco coerenti con lo stile di vita adottato.

La definizione delle soglie rappresenta quindi lo step in grado di avviare la macchina delle recommendations per un determinato assistito. Dal momento successivo allo svolgimento di tale operazione, essa è in grado di funzionare in completa autonomia. Esiste tuttavia, da parte dei client, la possibilità di innescare esplicitamente l'esecuzione delle regole mediante chiamate ad API che ritornano dei report. Tali report possono essere più o meno approfonditi e avere un intervallo temporale di analisi personalizzato; infatti, sono disponibili differenti tipologie di analisi corrispondenti ai gruppi di regole precedentemente individuati:

1. Attività: Analisi sulle sole regole appartenenti al gruppo attività.

2. Analisi Mediche: Analisi sulle sole regole appartenenti al gruppo analisi mediche.

3. Cibo: Analisi sulle sole regole appartenenti al gruppo cibo.

4. Frequenza Cardiaca: Analisi sulle sole regole appartenenti al gruppo frequenza cardiaca.

5. Sonno: Analisi sulle sole regole appartenenti al gruppo sonno.

6. Peso: Analisi sulle sole regole appartenenti al gruppo peso.

7. Completa: Analisi su tutti e sei gruppi di regole appena elencati.

I client possono richiedere il report in formato PDF o JSON. Il primo formato è adatto per una consultazione da parte dell'assistito o del medico. Indipendentemente dalla tipologia di account che lo richiede, viene ritornata una sintesi dei risultati, senza fornire spiegazioni prolisse. La motivazione risiede nel fatto che un report può essere già abbastanza lungo, di conseguenza non è consigliato introdurre informazioni dettagliate, pena perdita di attenzione da parte del lettore. Il formato JSON è invece pensato per funzionare in due modalità differenti. Se ad invocare l'API è un medico, ritorna il risultato "in sintesi", con l'idea di offrire una presentazione alternativa al report PDF e nativa del client web di HealthApp. Se, invece, l'API viene invocata da un client su cui è loggato un account di tipo paziente viene ritornato un report molto più dettagliato. L'idea in questo caso è quella di selezionare determinate tipologie di recommendations (idealmente quelle negative) e mostrarle a video in una sezione specifica dell'applicazione, con ovviamente una presentazione adeguata.

Per supportare questi diversi casi d'uso, gli endpoint disponibili sono:

- *POST /api/patients/{patientID}/recommendations/activity* – Riceve un oggetto JSON contenente un intervallo di date. Ritorna un JSON contenente i risultati dell'analisi, relative alle regole del gruppo attività, riferite all'intervallo di date passato e al paziente identificato mediante patientID. Se il metodo è invocato da un account di tipo paziente ritorna le informazioni in modo dettagliato, altrimenti in forma sintetica.

- *POST /api/patients/{patientID}/recommendations/activityPdf* - Riceve un oggetto JSON contenente un intervallo di date. Ritorna un PDF contenente i risultati dell'analisi, solamente in forma sintetica, relative alle regole del gruppo attività, riferite all'intervallo di date passato e al paziente identificato mediante patientID.

- *POST /api/patients/{patientID}/recommendations/hr* – Riceve un oggetto JSON contenente un intervallo di date. Ritorna un JSON contenente i risultati dell'analisi, relative alle regole del gruppo frequenza cardiaca, riferite all'intervallo di date passato e al paziente identificato mediante patientID. Se il metodo è invocato da un account di tipo paziente ritorna le informazioni in modo dettagliato, altrimenti in forma sintetica.

- *POST /api/patients/{patientID}/recommendations/hrPdf* - Riceve un oggetto JSON contenente un intervallo di date. Ritorna un PDF contenente i risultati dell'analisi, solamente in forma sintetica, relative alle regole del gruppo frequenza cardiaca, riferite all'intervallo di date passato e al paziente identificato mediante patientID.

- *POST /api/patients/{patientID}/recommendations/sleep* – Riceve un oggetto JSON contenente un intervallo di date. Ritorna un JSON contenente i risultati dell'analisi, relative alle regole del gruppo sonno, riferite all'intervallo di date passato e al paziente identificato mediante patientID. Se il metodo è invocato da un account di tipo paziente ritorna le informazioni in modo dettagliato, altrimenti in forma sintetica.

- *POST /api/patients/{patientID}/recommendations/sleepPdf* - Riceve un oggetto JSON contenente un intervallo di date. Ritorna un PDF contenente i risultati dell'analisi, solamente in forma sintetica, relative alle regole del gruppo sonno, riferite all'intervallo di date passato e al paziente identificato mediante patientID.

- *POST /api/patients/{patientID}/recommendations/weight* – Riceve un oggetto JSON contenente un intervallo di date. Ritorna un JSON contenente i risultati dell'analisi, relative alle regole del gruppo peso, riferite all'intervallo di date passato e al paziente identificato mediante patientID. Se il metodo è invocato da un account di tipo paziente ritorna le informazioni in modo dettagliato, altrimenti in forma sintetica.

- *POST /api/patients/{patientID}/recommendations/weightPdf* - Riceve un oggetto JSON contenente un intervallo di date. Ritorna un PDF contenente i risultati dell'analisi, solamente in forma sintetica, relative alle regole del gruppo peso, riferite all'intervallo di date passato e al paziente identificato mediante patientID.

- *POST /api/patients/{patientID}/recommendations/food* – Riceve un oggetto JSON contenente un intervallo di date. Ritorna un JSON contenente i risultati dell'analisi, relative alle regole del gruppo cibo, riferite all'intervallo di date passato e al paziente identificato mediante patientID. Se il metodo è invocato da un account di tipo paziente ritorna le informazioni in modo dettagliato, altrimenti in forma sintetica.

- *POST /api/patients/{patientID}/recommendations/foodPdf* - Riceve un oggetto JSON contenente un intervallo di date. Ritorna un PDF contenente i risultati dell'analisi, solamente in forma sintetica, relative alle regole del gruppo cibo, riferite all'intervallo di date passato e al paziente identificato mediante patientID.

- *POST /api/patients/{patientID}/recommendations/bloodAnalysis* – Riceve un oggetto JSON contenente un intervallo di date. Ritorna un JSON contenente i risultati dell'analisi, relative alle regole del gruppo analisi mediche, riferite all'intervallo di date passato e al paziente identificato mediante patientID. Se il metodo è invocato da un account di tipo paziente ritorna le informazioni in modo dettagliato, altrimenti in forma sintetica.

- *POST /api/patients/{patientID}/recommendations/bloodAnalysisPdf* - Riceve un oggetto JSON contenente un intervallo di date. Ritorna un PDF contenente i risultati dell'analisi, solamente in forma sintetica, relative alle regole del gruppo analisi mediche, riferite all'intervallo di date passato e al paziente identificato mediante patientID.

- *POST /api/patients/{patientID}/recommendations/completeAnalysis* – Riceve un oggetto JSON contenente un intervallo di date. Ritorna un JSON contenente i risultati dell'analisi, relative alle regole di tutti i gruppi individuati, riferite all'intervallo di date passato e al paziente identificato mediante patientID. Se il metodo è invocato da un account di tipo paziente ritorna le informazioni in modo dettagliato, altrimenti in forma sintetica.

- *POST /api/patients/{patientID}/recommendations/completeAnalysisPdf* - Riceve un oggetto JSON contenente un intervallo di date. Ritorna un PDF contenente i risultati dell'analisi, solamente in forma sintetica, relative alle regole di tutti i gruppi individuati, riferite all'intervallo di date passato e al paziente identificato mediante patientID.

L'ultimo gruppo di API è invece relativo alla consultazione dei report prodotti automaticamente dal recommender system. Le metodologie di produzione verranno spiegate approfonditamente nel paragrafo successivo. I punti salienti riguardano il fatto che i report vengono memorizzati e indicizzati nella base dati e sono consultabili in formato PDF.

È possibile accedere agli indici ed ai report per tipologia di paziente e per categorie di analisi. Gli endpoint disponibili sono:

- *GET /api/patients/{patientID}/recommendations/files/activity* – Ritorna un JSON contenente un vettore di oggetti relativi alle analisi sul gruppo attività per il paziente identificato mediante il contenuto della variabile patientID. Ogni oggetto contiene i metadati relativi ad un file: nome, url, tipo (in genere sempre application/pdf), dimensione in byte e data di creazione.

- *GET /api/patients/{patientID}/recommendations/files/activity/{fileID}* – Ritorna il file identificato mediante fileID, tipicamente in formato PDF relativo al report periodico prodotto automaticamente sul gruppo attività prodotto per il paziente identificato mediante il contenuto della variabile patientID.

- *GET /api/patients/{patientID}/recommendations/files/bloodAnalysis* – Ritorna un JSON contenente un vettore di oggetti relativi alle analisi sul gruppo analisi mediche per il paziente identificato mediante il contenuto della variabile patientID. Ogni oggetto contiene i metadati relativi ad un file: nome, url, tipo (in genere sempre application/pdf), dimensione in byte e data di creazione.

- *GET /api/patients/{patientID}/recommendations/files/bloodAnalysis/{fileID}* – Ritorna il file identificato mediante fileID, tipicamente in formato PDF relativo al report periodico prodotto automaticamente sul gruppo analisi mediche prodotto per il paziente identificato mediante il contenuto della variabile patientID.

- *GET /api/patients/{patientID}/recommendations/files/food* – Ritorna un JSON contenente un vettore di oggetti relativi alle analisi sul gruppo cibo per il paziente identificato mediante il contenuto della variabile patientID. Ogni oggetto contiene i metadati relativi ad un file: nome, url, tipo (in genere sempre application/pdf), dimensione in byte e data di creazione.

- *GET /api/patients/{patientID}/recommendations/files/food/{fileID}* – Ritorna il file identificato mediante fileID, tipicamente in formato PDF relativo al report periodico prodotto automaticamente sul gruppo cibo prodotto per il paziente identificato mediante il contenuto della variabile patientID.

- *GET /api/patients/{patientID}/recommendations/files/hr* – Ritorna un JSON contenente un vettore di oggetti relativi alle analisi sul gruppo frequenza cardiaca per il paziente identificato mediante il contenuto della variabile patientID. Ogni oggetto contiene i metadati relativi ad un file: nome, url, tipo (in genere sempre application/pdf), dimensione in byte e data di creazione.

- *GET /api/patients/{patientID}/recommendations/files/hr/{fileID}* – Ritorna il file identificato mediante fileID, tipicamente in formato PDF relativo al report periodico prodotto

automaticamente sul gruppo frequenza cardiaca prodotto per il paziente identificato mediante il contenuto della variabile patientID.

- *GET /api/patients/{patientID}/recommendations/files/sleep* – Ritorna un JSON contenente un vettore di oggetti relativi alle analisi sul gruppo sonno per il paziente identificato mediante il contenuto della variabile patientID. Ogni oggetto contiene i metadati relativi ad un file: nome, url, tipo (in genere sempre application/pdf), dimensione in byte e data di creazione.

- *GET /api/patients/{patientID}/recommendations/files/sleep/{fileID}* – Ritorna il file identificato mediante fileID, tipicamente in formato PDF relativo al report periodico prodotto automaticamente sul gruppo sonno prodotto per il paziente identificato mediante il contenuto della variabile patientID.

- *GET /api/patients/{patientID}/recommendations/files/weight* – Ritorna un JSON contenente un vettore di oggetti relativi alle analisi sul gruppo peso per il paziente identificato mediante il contenuto della variabile patientID. Ogni oggetto contiene i metadati relativi ad un file: nome, url, tipo (in genere sempre application/pdf), dimensione in byte e data di creazione.

- *GET /api/patients/{patientID}/recommendations/files/weight/{fileID}* – Ritorna il file identificato mediante fileID, tipicamente in formato PDF relativo al report periodico prodotto automaticamente sul gruppo peso prodotto per il paziente identificato mediante il contenuto della variabile patientID.

- *GET /api/patients/{patientID}/recommendations/files/completeAnalysis* – Ritorna un JSON contenente un vettore di oggetti relativi alle analisi su tutti i gruppi di regole per il paziente identificato mediante il contenuto della variabile patientID. Ogni oggetto contiene i metadati relativi ad un file: nome, url, tipo (in genere sempre application/pdf), dimensione in byte e data di creazione.

- *GET /api/patients/{patientID}/recommendations/files/completeAnalysis/{fileID}* – Ritorna il file identificato mediante fileID, tipicamente in formato PDF relativo al report periodico prodotto automaticamente su tutti i gruppi di regole prodotto per il paziente identificato mediante il contenuto della variabile patientID.

Grazie alle API appartenenti ai tre macrogruppi appena illustrati è possibile, da parte dei client, avere piena interazione con il recommender system. Tale interazione risulta piuttosto semplice ed immediata, nonostante il sistema sia così complesso da dover essere trattato come un modulo a sé, orizzontale alle quattro macroaree di HealthApp. I paragrafi successivi approfondiranno ulteriori aspetti del sistema che produce i consigli per gli assistiti medici.

5.3 Esecuzione periodica e generazione report

Al fine di mantenere uno storico dei progressi di ogni paziente, all'interno di HealthApp viene implementata, lato backend, una routine periodica a cadenza settimanale. Essa è incaricata di eseguire le tipologie di analisi sintetiche, relative a tutti i gruppi, per ogni paziente e di memorizzare i report prodotti sotto forma di file PDF.

Per supportare la creazione dei report viene definita un'apposita classe in grado di produrre, a partire dal risultato delle analisi, un file PDF organizzato in differenti tabelle. Il file è denominato seguendo lo schema: *tipologiaAnalisi_idPaziente_dataProduzioneReport.pdf* (esempio *completeAnalysis_1_2023-01-20.pdf*). Il titolo richiama il nome del file; seguono poi le tabelle. Per ogni tipologia di analisi (*attività, analisi mediche, cibo, frequenza cardiaca, sonno o peso*) si ha il titolo della tipologia e i gruppi di tabelle, ognuna preceduta dal sottotitolo relativo al sottogruppo di regole eseguite. Ogni tabella è composta da cinque colonne nel caso di regole a parametro singolo: *data* (a cui fa riferimento l'informazione analizzata), *risultato dell'analisi*, *risultato settimanale*, *risultato mensile*, *risultato positivo*. Nel caso di regole a parametri multipli o a macroarea multipla non si hanno le colonne relative ai risultati settimanali e mensili. La colonna "*risultato positivo*" serve a dare enfasi ai risultati negativi, che saranno evidenziati anche da un font di colore rosso.

La figura seguente mostra un estratto di un report pdf per rendere meglio l'idea:

Complete Analysis Result - 2023-01-28

Activity Analysis Result

Steps Analysis Result

Date	Result	Result Week	Result Month	Positive Result
2023-01-21	Steps low	Steps low	Steps low	false
2023-01-22	Steps low	Steps low	Steps low	false
2023-01-23	Steps low	Steps low	Steps low	false
2023-01-24	Steps low	Steps low	Steps low	false
2023-01-25	Steps low	Steps low	Steps low	false
2023-01-26	Steps low	Steps low	Steps low	false
2023-01-27	Steps low	Steps low	Steps low	false
2023-01-28	Steps low	Steps low	Steps low	false

Calories Analysis Result

Date	Result	Result Week	Result Month	Positive Result
2023-01-21	Calories are ok	Calories are ok	Calories are ok	true
2023-01-22	Calories are ok	Calories are ok	Calories are ok	true
2023-01-23	Calories are ok	Calories are ok	Calories are ok	true

Figura 14 - Estratto report pdf settimanale

Per il salvataggio dei file così prodotti, occorre predisporre opportunamente la base dati. Per ogni tipologia di analisi è necessaria una tabella (sette in totale se si considera l'analisi completa) che funge da indice e

contiene i metadati ed il contenuto del file. Attraverso le API precedentemente illustrate, paziente e medico possono consultare i risultati dei report.

Tuttavia, per evitare che l'occupazione di spazio cresca a dismisura, i report vengono mantenuti per un tempo limitato all'interno della base dati. Infatti, un'ulteriore routine periodica a cadenza trisettimanale provvede a rimuovere i file più vecchi di un anno. Tale scelta non rappresenta un problema di perdita di informazioni. Infatti, i dati su cui è stata effettuata l'analisi rimangono disponibili ed è possibile recuperarne i report utilizzando le API che permettono di generare un resoconto su un intervallo di date esplicito.

Come ribadito in precedenza, lo scopo dell'esecuzione periodica è quello di fornire un ulteriore supporto storico, anche se limitato nel tempo, a medico e assistito per la visualizzazione dei progressi effettuati da quest'ultimo. Il report funge inoltre da resoconto settimanale e può essere utilizzato come strumento di rapida consultazione dei risultati, andando poi ad approfondire quelli in cui vengono riscontrate maggiori criticità.

5.4 Performance e conclusioni

Il modulo illustrato nel capitolo precedente e in questo stesso si basa sull'integrazione di un Business Rule Engine, Drools e sulla definizione di regole da applicare. L'intero sottosistema aggiunge sicuramente un overhead spaziale e temporale all'applicazione su cui è necessario fare delle considerazioni.

Nel caso spaziale, al fine di limitare il crescere dell'occupazione di memoria è stata introdotta la routine di pulizia illustrata precedentemente. Le considerazioni seguenti riguardano l'overhead temporale. Numerosi riscontri dimostrano come l'impatto dell'esecuzione delle regole sul tempo di risposta sia da considerarsi minimo rispetto ai tempi della rete. Infatti, prendendo come riferimento tre API, una delle quali (GET) ritorna semplicemente la lista dei pazienti, l'altra (POST) esegue un'analisi di tipo attività su un arco di cinque giorni e l'ultima (POST) esegue un'analisi di tipo completo sul medesimo intervallo temporale, è possibile notare come il tempo di risposta sia piuttosto simile e pari a circa 150ms. Addirittura, nel caso dell'analisi completa (API più "pesante" in quanto esegue tutte le regole) il tempo di risposta è pari a 141ms, mentre la lista dei pazienti viene ritornata in 142ms. Le considerazioni espresse si basano su un numero maggiore di test eseguiti e ne viene riportato solamente uno ai fini sintetici.

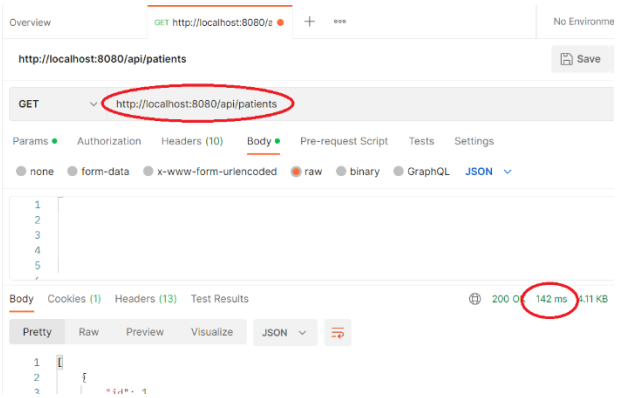


Figura 15 - Lista pazienti, tempo di risposta

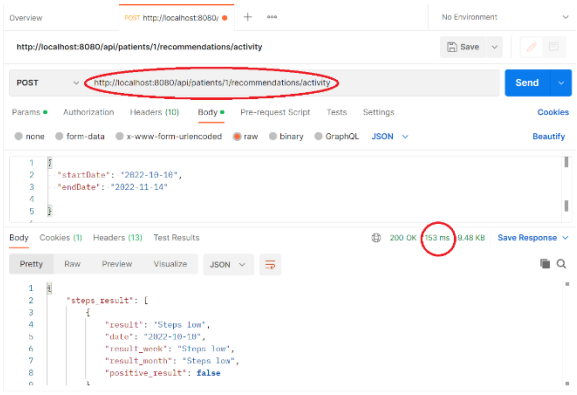


Figura 16 - Analisi di tipo attività, tempo di risposta

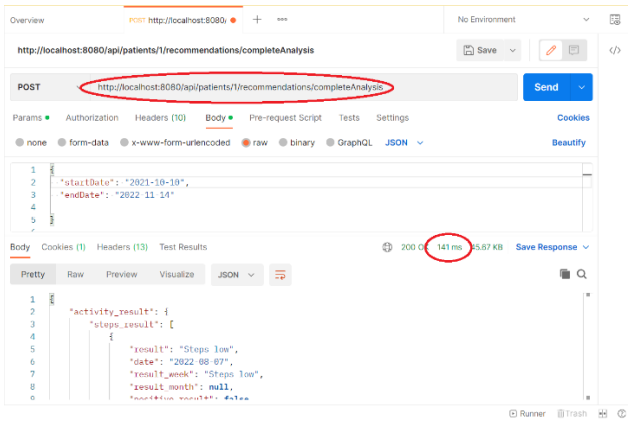


Figura 17 - Analisi completa, tempo di risposta

È possibile quindi affermare che il modulo delle recommendations è assolutamente gestibile dal punto di vista spaziale e temporale e non aggiunge overhead significativi all'applicazione. Ovviamente vi saranno dei casi limite in cui la risposta verrà elaborata in un tempo di risposta relativamente maggiore, dell'ordine dei secondi, come ad esempio nel caso di un'analisi completa su un arco temporale di sei o più mesi.

In conclusione, il recommender system aggiunge sicuramente delle funzionalità innovative ad HealthApp, difficili da trovare in altre applicazioni del settore. La maggior parte delle regole, come visto, sono a parametro singolo; sarebbe molto interessante espandere l'insieme di regole inserendone nuove a parametri multipli ed a macroarea multipla in quanto potrebbe risultare non banale avere il supporto di un sistema in grado di analizzare e segnalare eventuali anomalie relative alle correlazioni tra differenti parametri. Per fare ciò, è necessaria una continua collaborazione con i medici, al fine di individuare i parametri da analizzare e le relazioni esistenti.

Lo scopo finale del sottosistema e dell'intera applicazione è quello di provare a migliorare il benessere degli assistiti medici, ma in nessuna parte di questo progetto si è pensato che il recommender system possa sostituire il parere di un medico o prendere decisioni al posto di un medico. Il caso d'uso ideale è appunto quello di supportare il medico nelle decisioni o di far scattare dei campanelli d'allarme nel paziente, qualora vengano rilevati dati anomali, al fine di invogliarlo a tenere sotto controllo il proprio stile di vita.

6. Test su dataset esistenti

Lo scopo del capitolo è quello di validare le regole precedentemente illustrate. Il processo consisterà nell'importare in HealthApp alcuni dataset già esistenti e applicarvi le regole, osservando che i risultati prodotti siano in linea con le aspettative. L'utilità del testing non è da sottovalutare, in quanto consente sempre di far emergere problemi e criticità che possono sfuggire in fase di sviluppo. Tuttavia, essendo già stati effettuati numerosi test prima di quelli descritti in questo capitolo, ci si aspetta che le correzioni da effettuare in questo caso siano minime.

6.1 Ricerca dataset esistenti

Il primo passo consiste nella ricerca di dataset esistenti contenenti informazioni utili al processo di testing, in particolare quelli prodotti da dispositivi Fitbit. È fondamentale che le informazioni ricercate siano valide e accurate, al fine di simulare un comportamento realistico del recommender system.

Tre dataset trovati risultano di particolare interesse: *FitBit Fitness Tracker Data* offerto da *Kaggle* [52], *PMData - A lifelogging dataset of 16 persons during 5 months using Fitbit, Google Forms and PMSys*, offerto da *Simula* [53] e *Dietary-Behavior-Dataset* offerto dal *CSSEJH (Center for Systems Science and Engineering at John Hopkins University)* [54].

Per semplicità, si parlerà rispettivamente di *Kaggle dataset, Simula dataset e Dietary dataset*.

Il dataset *Kaggle* raccoglie il risultato di alcuni sondaggi in cui 32 utenti hanno condiviso i propri dati Fitbit. Essi sono relativi al periodo che va dal 12 aprile al 12 maggio 2016. Le informazioni d'interesse per i test nell'applicazione riguardano: massa e BMI per il sottogruppo peso, passi e calorie per il sottogruppo attività, tempo di sonno per il sottogruppo sonno. Le informazioni relative al sottogruppo frequenza cardiaca non sono d'interesse in quanto viene riportato il battito (non a riposo) al secondo. Non sono invece presenti informazioni riguardanti cibo e analisi mediche.

Il dataset Simula contiene informazioni prodotte nell'arco di cinque mesi (dal 1° novembre 2019 al 31 marzo 2020) utilizzando lo smartwatch *Fitbit Versa 2* e l'app di logging *PMSys*. A questi si aggiunge l'utilizzo di *Google Form* per permettere ai 16 utenti partecipanti di registrare la propria massa misurata in seguito ad un pasto, la quantità di fluidi ingerita e il consumo di alcol. Alcuni utenti hanno inoltre scattato delle foto di tutto ciò che hanno mangiato. Vista l'abbondanza di informazioni, è necessario estrarre quelle d'interesse per i test ossia: passi e calorie per il sottogruppo attività, tempo di sonno per il sottogruppo sonno, battito a riposo per il sottogruppo frequenza cardiaca, massa e BMI (calcolato a partire da massa e altezza) per il sottogruppo peso. Non si hanno a disposizione dati su analisi mediche e cibo.

Infine, il dietary dataset contiene i risultati del NHANES (National Health and Nutrition Examination Survey) del periodo 2015-2016 condotto dal governo degli USA e relativo al diario alimentare registrato da centinaia di utenti. Ai fini del test verrà utilizzato un sotto insieme ridotto di utenti e un periodo temporale di due giorni. Le informazioni di interesse riguardano i cibi consumati, la quantità e i valori nutrizionali associati.

Per ovviare all'assenza di dataset contenenti informazioni sull'analisi mediche, saranno utilizzati i valori reali provenienti dalle mie analisi mediche. Nonostante sia noto dalla letteratura informatica che utilizzare le informazioni prodotte dagli sviluppatori per il testing sia una cattiva pratica, e che questi non debbano impersonificate gli utenti, pena utilizzo dell'applicazione con bias e assunzioni non sempre valide, in questo caso si fa riferimento a dati derivanti da misurazioni di laboratorio e quindi non direttamente prodotti da me.

Si utilizzeranno infine i dati prodotti tramite il tracker Fitbit personale dal professor Morisio e di un collega tesista come ulteriore strumento di validazione per le recommedations. Essi sono relativi ai sottogruppi: attività, frequenza cardiaca e sonno; si parlerà in questo caso e in quello delle analisi mediche di dati autoprodotti.

I dataset citati, ad esclusione di quelli autoprodotti, vanno integrati all'interno dell'applicazione mediante adattatori in grado di leggerne il formato, interpretarlo, estrarne i dati e memorizzarli all'interno della base dati di HealthApp. Verranno illustrati nel paragrafo successivo.

6.2 Implementazione adattatori

I dataset sono disponibili in formato JSON o CSV. Occorrerà quindi definire delle classi in grado di fungere da adattatori e leggere tali formati, salvandone il contenuto nella base dati di HealthApp. I file di partenza faranno parte delle risorse dell'applicazione.

La prima famiglia di adattatori riguarda il Dataset *Simula*. È costituita da tre metodi:

- *getActivitySimulaDataset()* : Effettua la lettura del file *steps.json* andando a salvare in un buffer di memoria i dati relativi ai passi, mantenendo la separazione temporale. Effettua successivamente la lettura del file *calories.json* recuperando le informazioni relative alle calorie bruciate. Effettua il merge con i dati dei passi e li salva nella base dati associandoli ad un paziente fittizio. Tale operazione è necessaria perché i dati si trovano su due file separati. La memorizzazione fa riferimento ad un singolo utente del campione. Per ogni utente del campione è necessario aggiornare i file JSON nelle risorse presenti nell'applicazione.

- *getSleepSimulaDataset()* : Effettua la lettura del file *sleep.json* andando a salvare i tempi di sonno per l'utente nella base dati di HealthApp, associandoli ad un paziente fittizio. È necessario chiamare il metodo per ogni utente del campione, sovrascrivendo ogni volta il file JSON.

- *getRestHrSimulaDataset()* : Effettua la lettura del file *resting_ heart_ rate.json* andando a salvare le frequenze cardiache a riposo per l'utente nella base dati di HealthApp, associandoli ad un paziente fittizio. È necessario chiamare il metodo per ogni utente del campione, sovrascrivendo ogni volta il file JSON.

- getWeightSimulaDataset(height: Double) : Effettua la lettura del file *reporting.csv* andando ad estrapolare la massa per ogni data. Inoltre, calcola il BMI a partire dal valore letto e dall'altezza ricevuta come parametro. Salva massa e BMI nella base dati dell'applicazione, associandoli ad un

paziente fittizio. Anche in questo caso è necessario chiamare il metodo per ogni utente del campione, sovrascrivendo opportunamente il file CSV.

I test verranno eseguiti utente per utente, andando ogni volta a ripulire il database e aggiornare i file di riferimento.

Gli adattatori del dataset *Kaggle* hanno un funzionamento simile, ma devono gestire solamente dei file in formato CSV:

- *getActivityKaggleDataset()* : Effettua la lettura di passi e calorie bruciate dal file *dailyActivity_ merged.csv*, filtrando le informazioni per l'id dell'utente passato. Effettua successivamente la memorizzazione all'interno della base dati, associando i dati ad un paziente fittizio.

- *getSleepKaggleDataset()* : Effettua la lettura dei tempi di sonno in minuti dal file *sleepDay_ merged.*csv, filtrando le informazioni per l'id dell'utente passato. Converte le informazioni in millisecondi e le salva nella base dati di HealthApp, associandole ad un paziente fittizio.

- *getWeightKaggleDataset()* : Effettua la lettura del file *weightLogInfo_ merged.csv*, filtrando le informazioni per l'id dell'utente passato. Salva successivamente i valori di massa e BMI nella base dati di HealthApp, associandoli ad un paziente fittizio.

L'insieme dei metodi per il dataset *Kaggle* memorizza le informazioni per un singolo utente del campione. Tra un test ed un altro è necessario aggiornare l'identificativo dell'utente in modo tale da aggiornare i dati da processare.

Per quanto riguarda invece il *dietary* dataset, i dati ed i valori nutrizionali associati non fanno riferimento al provider scelto da HealthApp, ossia *FatSecret*. In più, non sono presenti le informazioni relative ai minerali e alle vitamine. Per tale motivo si prendono manualmente i cibi con i rispettivi quantitativi e si recuperano i valori nutrizionali dal corrispondente alimento su FatSecret, andando infine a popolare il database. Non è quindi necessaria l'implementazione di alcun adattatore per questo dataset.

I dati autoprodotti invece vengono nativamente salvati all'interno della base dati di HealthApp. Per tale motivo anche in questo caso non sono necessari adattatori.

In seguito all'importazione dei dataset, è possibile procedere con i test. È importante sottolineare che in tutti i test eseguiti sono state utilizzate, per le regole, le soglie statiche definite dalla letteratura, senza alcuna personalizzazione.

6.3 Risultati test su simula dataset

I test sono stati condotti su sedici utenti, nell'arco di cinque mesi, effettuando l'analisi di tipo completo. Si riporta una sintesi dei risultati (il termine Rec. Indica recommendations):

Tabella 5 - Risultati Simula Dataset

Utente	Informazioni	Rec. Positive Prodotte	Rec. Negative Prodotte	Rec. Errate Prodotte	Rec. Corrette Prodotte
P01	Età: 48 anni Altezza: 195cm Genere: Maschio	427	357	0	784
P02	Età: 60 anni Altezza: 180cm Genere: Maschio	577	224	0	801
P03	Età: 25 anni Altezza: 184cm Genere: Maschio	462	272	0	734
P04	Età: 26 anni Altezza: 163cm Genere: Femmina	568	157	0	725
P05	Età: 35 anni Altezza: 176cm Genere: Maschio	382	351	0	733
P06	Età: 42 anni Altezza: 179cm Genere: Maschio	524	308	0	832
P07	Età: 26 anni Altezza: 177cm Genere: Maschio	731	83	0	814
P08	Età: 27 anni Altezza: 186cm Genere: Maschio	552	124	0	676

P09	Età: 26 anni Altezza: 180cm Genere: Maschio	568	198	0	766
P10	Età: 38 anni Altezza: 179cm Genere: Femmina	567	136	0	703
P11	Età: 25 anni Altezza: 171cm Genere: Femmina	482	165	0	647
P12	Età: 27 anni Altezza: 178cm Genere: Maschio Note: nessun dato su frequenza cardiaca	379	95	0	474
P13	Età: 31 anni Altezza: 183cm Genere: Maschio Note: nessun dato su frequenza cardiaca	348	145	0	493
P14	Età: 45 anni Altezza: 181cm Genere: Maschio	604	177	0	781

P15	Età: 54 anni Altezza: 180cm Genere: Maschio Note: Infortunato; nessun dato su massa	397	181	0	578
P16	Età: 23 anni Altezza: 182cm Genere: Maschio	493	240	0	733

Si osserva che le regole sono state eseguite in ogni test senza errori, a testimonianza che, anche nei casi in cui il valore si trova all'estremo di un intervallo, il recommender system funziona correttamente. Inoltre, la maggioranza delle recommendations negative prodotte per i vari utenti si riferisce alle regole *"Rest Hr low"* e *"Sleep time low"*. Ciò dimostra che un approccio puramente *theory-based* risulterebbe efficiente in alcuni sottogruppi del dataset ed inefficiente su altri. Di conseguenza, si conclude che è molto importante l'utilizzo delle soglie personalizzate per paziente.

6.4 Risultati test su kaggle dataset

I test sono stati condotti su trentadue utenti, nell'arco di un mese, effettuando l'analisi di tipo completo. Si riporta una sintesi dei risultati (il termine Rec. Indica recommendations):

Tabella 6 - Risultati Kaggle Dataset

Utente	Informazioni	Rec. Positive Prodotte	Rec. Negative Prodotte	Rec. Errate Prodotte	Rec. Corrette Prodotte
1	ID Kaggle: 1503960366	63	28	0	91
2	ID Kaggle: 1624580081 Disponibili solo dati attività	32	30	0	62

3	ID Kaggle: 1844505072 Disponibili solo dati attività e sonno	33	32	0	65
4	ID Kaggle: 1927972279	33	36	0	69
5	ID Kaggle: 2022484408 Disponibili solo dati attività	58	4	0	62
6	ID Kaggle: 2026352035 Disponibili solo dati attività e sonno	60	30	0	90
7	ID Kaggle: 2320127002 Disponibili solo dati attività e sonno	31	32	0	63
8	ID Kaggle: 2347167796 Disponibili solo dati attività e sonno	35	16	0	51
9	ID Kaggle: 2873212765 Disponibili solo dati attività e peso	35	31	0	66
10	ID Kaggle: 3372868164 Disponibili solo dati attività	20	20	0	40

11	ID Kaggle: 3977333714 Disponibili solo dati attività e sonno	62	26	0	88
12	ID Kaggle: 4020332650 Disponibili solo dati attività e sonno	35	35	0	70
13	ID Kaggle: 4057192912 Disponibili solo dati attività	4	4	0	8
14	ID Kaggle: 4319703577	63	29	0	92
15	ID Kaggle: 4388161847 Disponibili solo dati attività e sonno	64	21	0	85
16	ID Kaggle: 4445114986	38	52	0	90
17	ID Kaggle: 4558609924	45	32	0	77
18	ID Kaggle: 4702921684 Disponibili solo dati attività e sonno	44	45	0	89
19	ID Kaggle: 5553957443 Disponibili solo dati attività e sonno	57	36	0	93
20	ID Kaggle: 5577150313	51	37	0	88

21	ID Kaggle: 6117666160 Disponibili solo dati attività e sonno	47	27	0	74
22	ID Kaggle: 6290855005 Disponibili solo dati attività	28	30	0	58
23	ID Kaggle: 6775888955 Disponibili solo dati attività e sonno	26	29	0	55
24	ID Kaggle: 6962181067	123	30	0	153
25	ID Kaggle: 7007744171 Disponibili solo dati attività e sonno	44	10	0	54
26	ID Kaggle: 7086361926 Disponibili solo dati attività e sonno	55	31	0	86
27	ID Kaggle: 8053475328 Disponibili solo dati attività e sonno	59	6	0	65
28	ID Kaggle: 8253242879 Disponibili solo dati attività e sonno	21	17	0	38

29	ID Kaggle: 8378563200 Disponibili solo dati attività e sonno	57	36	0	93
30	ID Kaggle: 8583815059 Disponibili solo dati attività	36	26	0	62
31	ID Kaggle: 8792009665 Disponibili solo dati attività e sonno	34	39	0	73
32	ID Kaggle: 8877689391 Disponibili solo dati attività e peso	107	3	0	110

Per il *Kaggle Dataset* si hanno un numero maggiore di utenti da testare ma su un arco di tempo minore. Le recommendations negative prodotte si riferiscono principalmente alla regola *"Steps low"*. La motivazione è da ricercare nel fatto che la popolazione presa in esame presenta un pattern comune, ossia quello di avere un quantitativo di passi giornaliero inferiore ai diecimila (soglia statica stabilita dalla letteratura). Si osserva invece che non si hanno problemi per quanto riguarda il tempo di sonno, regola che aveva prodotto numerose recommendations negative nei test relativi al *Simula Dataset*. Si conclude che per le macroaree movimento, salute e sonno si ha una forte dipendenza delle soglie dalla popolazione scelta. È impossibile stabilire a priori un insieme di soglie "universali" ed è molto importante quindi l'utilizzo delle soglie personalizzate. Il test sui dataset *Kaggle* e *Simula* conferma quanto dichiarato nel capitolo 5 relativamente all'approccio di un sistema puramente *theory-based*, ossia è inefficiente.

6.5 Risultati test su dietary dataset

Il dietary dataset serve per effettuare test relativi alla macroarea alimentazione. Si sceglie una popolazione ridotta di tre utenti che registrano i propri pasti nell'arco di due giorni. Si riporta una sintesi dei risultati (il termine Rec. Indica recommendations):

Tabella 7 - Risultati Dietary Dataset

Utente	Informazioni	Rec. Positive Prodotte	Rec. Negative Prodotte	Rec. Errate Prodotte	Rec. Corrette Prodotte
1	ID Dietary: 83732	29	17	0	46
2	ID Dietary: 83733	31	15	0	46
3	ID Dietary: 83734	29	17	0	46

Anche le regole del sottogruppo cibo si comportano egregiamente, non presentando alcun errore in fase di esecuzione. In questo caso, le recommendations negative prodotte fanno principalmente riferimento allo sforamento delle soglie nel caso di zuccheri, grassi di ogni tipo e colesterolo. Un risultato del genere era atteso in quanto il dataset fa riferimento ad utenti americani che seguono una dieta differente da quella mediterranea, non bilanciata correttamente. Lo stesso discorso si applica anche sulle regole a parametri multipli che coinvolgono i grassi, in cui troviamo prevalentemente recommendations di tipo negativo, mentre le altre risultano generare in maggioranza recommendations positive.

Complessivamente, il sistema a soglie statiche sembra funzionare bene in questo sottogruppo, e si potrebbe pensare di non dover intervenire. Tuttavia, non sono note le informazioni sugli utenti oggetto dei test; di conseguenza, non si può concludere che un approccio interamente *theory-based* funzioni in quanto sicuramente servirà intervenire sulle soglie in casi di diete, obesità, diabete o altre circostanze legate all'alimentazione.

6.6 Risultati test su dati autoprodotti

La procedura di test sui dati autoprodotti è finalizzata a verificare il corretto funzionamento, oltre che delle regole, anche dei meccanismi di interazione tra il backend di HealthApp e quello Fitbit. Inoltre, si propone di coprire il sottogruppo relativo alle analisi mediche, finora non testato in quanto nessun dataset rilevante è disponibile online.

Si utilizzeranno i dati prodotti:

- Dal professor Morisio mediante dispositivo Fitbit, relativi ai sottogruppi attività, frequenza cardiaca e sonno nell'arco di tempo che va dal 23/12/2022 al 09/02/2023.

- Da un collega tesista mediante dispositivo Fitbit, relativi ai sottogruppi attività, frequenza cardiaca e sonno nell'intervallo temporale che va dal 02/11/2022 al 04/11/2022

- Da me, relativi al sottogruppo analisi mediche, relativi a tre esami datati marzo 2014, settembre 2020 e gennaio 2022.

Relativamente ai test condotti si ha:

Tabella 8 - Risultati dati autoprodotti

Utente	Informazioni	Rec. Positive Prodotte	Rec. Negative Prodotte	Rec. Errate Prodotte	Rec. Corrette Prodotte
Prof. Morisio	Solo dati attività, frequenza cardiaca e sonno	129	63	0	192
Collega Tesista	Solo dati attività, frequenza cardiaca e sonno	7	4	0	11
Miei dati	Solo dati analisi mediche	14	31	0	45

Relativamente ai risultati del professore e del collega, la maggior parte di recommendations di tipo negativo deriva da un numero di passi e di ore di sonno inferiori alle soglie individuate (rispettivamente 10000 passi e 8 ore di sonno). Può sembrare eccessivo il fatto che circa un terzo delle recommendations totali sia di tipo negativo, ma l'obiettivo di HealthApp è proprio quello di invogliare gli utenti a migliorare il proprio benessere, presentando le informazioni in una modalità che non faccia sentire in colpa l'utilizzatore, ma che lo spinga a curare il proprio stile di vita.

Per quanto riguarda invece il test sulle analisi mediche, la maggior parte di recommendations negative sono dovute ai dati non presenti (*null*), mentre le poche altre sono relative a valori di poco fuori soglia. La motivazione va ricercata nel fatto che differenti laboratori possono avere soglie leggermente variabili. Per completezza, sono stati effettuati diversi test, non riportati, su dati inventati con il risultato di errore pari a zero sia sulle regole a parametro singolo che su quelle a parametri multipli.

6.7 Conclusioni

L'insieme di test condotti ha permesso di far funzionare a pieno regime il recommender system, che si è dimostrato solido di fronte ad analisi su brevi, medi e lunghi periodi. Il risultato importante da sottolineare è che, nella varietà di dataset utilizzati, non sono mai state riscontrate eccezioni in fase di esecuzione. È infatti molto importante che il sistema non vada in crash, altrimenti il rischio è di provocare l'arresto di tutto il server e la conseguente perdita di disponibilità.

L'intero backend risulta adesso collaudato e pronto ad entrare in produzione, ossia essere accessibile mediante internet dai client esterni. Il capitolo successivo tratterà i vari step del deployment, insieme alle considerazioni finali.

7. Deployment, spunti di miglioramento e conclusioni

7.1 Deployment

Per mettere online e quindi rendere accessibile ai client web e mobile backend e recommender system è necessario passare dalla fase di sviluppo (development) a quella di distribuzione (deployment).

Il codice fin qui prodotto e testato deve essere "impacchettato, caricato ed eseguito" su una macchina che esponga un indirizzo IP pubblico, ossia raggiungibile da Internet. È possibile quindi illustrare la procedura di deployment in due fasi: impacchettatura e caricamento (containerizing) ed esecuzione (execution).

Si farà riferimento all'insieme di backend e recommender system sotto la denominazione di "*core*". Una procedura di deployment simile a quella che verrà descritta sarà eseguita dai colleghi tesisti anche per quanto riguarda i frontend web e mobile, in modo tale da rendere disponibili la web application e la mobile application agli utilizzatori finali. Il capitolo tratterà esclusivamente il deployment del core.

7.1.1 Containerizing

Il primo step consiste nel creare un'immagine del core che contenga tutte gli elementi necessari per permettere un'esecuzione autonoma del sistema. A tal proposito si utilizza Docker, un software libero pensato per creare ed eseguire immagini in un ambiente isolato, chiamato container. [55] Due o più container possono comunicare attraverso un *network bridge*, in grado di collegare le reti dei singoli container tra loro. Il *bridge* verrà utilizzato nella comunicazione tra il container del core e quello del frontend web.

Un container permette di eseguire al suo interno un'immagine, costituita da tutte le risorse necessarie (librerie, codice eseguibile, volumi ecc.) per l'esecuzione autonoma di un processo. Un'immagine viene costruita mediante un insieme di direttive raccolte all'interno di un file noto come *dockerfile*.

Ogni immagine deve avere il proprio container; per eseguirne una o più è possibile utilizzare un'interfaccia a linea di comando, un'interfaccia grafica o il *docker compose tool*, in grado di eseguire applicazioni multi-container a partire da un file di configurazione con estensione YAML.

Nel caso del *core* di HealthApp, è necessario definire il *dockerfile* che permetta di creare l'immagine da eseguire. Tale file è così costituito:

```
1. FROM adoptopenjdk/openjdk11:alpine-jre
2. VOLUME /data
3. ADD /data/db_data.mv.db /data/db_data.mv.db
4. ARG JAR_FILE=target/*.jar
5. COPY ${JAR_FILE} app.jar
6. EXPOSE 8080
7. ENTRYPOINT ["java","-jar","/app.jar"]
```

La direttiva 1 indica l'immagine di base utilizzata, contenente tutte le librerie e gli strumenti di contorno necessari.

Le direttive 2 e 3 definiscono il volume che fungerà da base dati per l'applicazione ed il suo percorso. In questo caso, il volume si troverà nella cartella *data*, mentre la base dati sarà costituita da un semplice file, copiato dal sistema locale (percorso */data/db_ data.mv.db*) all'immagine Docker (stesso percorso), contenente tabelle e dati.

La direttiva 4 individua come variabile denominata *JAR_ FILE* il file con estensione JAR contenuto nella cartella target mentre la 5 copia tale file nel percorso root dell'immagine Docker, assegnandogli il nome *app.jar*

Infine, la direttiva 6 espone, rendendo accessibile, la porta 8080 sulla rete del container che conterrà l'immagine mentre la 7 indica il comando da eseguire per avviare l'esecuzione.

Definito il *dockerfile*, il passaggio successivo è relativo alla produzione del file JAR (Java Archive) [56], ossia un archivio contenente tutto il codice sorgente del core, utilizzato in fase d'esecuzione. Per generare tale file è sufficiente eseguire, nella cartella root del progetto, il comando *maven package*, lasciando al tool il compito di impacchettare le varie classi.

Una volta definiti *dockerfile* e JAR non resta che creare l'immagine da eseguire all'interno del container Docker. Per fare ciò è possibile utilizzare l'interfaccia grafica messa a disposizione dall'IDE (IntelliJ in questo caso) o il comando *docker build*. Il processo di creazione dura qualche manciata di secondi e al termine genera l'immagine e il volume associato, pronti da eseguire all'interno di un container.

Non resta che caricare l'immagine sul repository Docker del progetto e iniziare la seconda fase, ossia quella di esecuzione.

7.1.2 Execution

Per eseguire l'immagine è necessaria una macchina (host) connessa alla rete e raggiungibile da altre macchine mediante un indirizzo IP pubblico. Nel caso di HealthApp esistono due soluzioni valide:

- una macchina interna alla rete del Politecnico e accessibile tramite un percorso relativo al dominio softeng.polito.it
- una macchina offerta gratuitamente per 12 mesi da Amazon Web Services (AWS)

Inizialmente si effettuano le prime esecuzioni su Softeng per poi sportarsi su AWS. La scelta è motivata dal fatto di avere un host direttamente controllabile dagli sviluppatori senza la necessità di ricorrere ad intermediari. Poiché il Politecnico offre tale possibilità solo al termine di una lunga procedura di richiesta, si ricorre ad un intermediario solamente per verificare il funzionamento dell'immagine su Softeng, mentre si utilizza AWS per l'esecuzione vera e propria, in modo tale da poter correggere qualsiasi anomalia nel minor tempo possibile, offrire la maggiore disponibilità possibile del server e garantire sempre una corretta comunicazione tra client e core.

In ogni caso, i passaggi da eseguire per effettuare l'esecuzione sono comuni ad entrambe le soluzioni:

1. Installare l'applicativo Docker sulla macchina scelta.

2. Scaricare l'immagine precedentemente caricata sul repository.

3. Eseguire il comando *docker run* per eseguire l'immagine all'interno del container, mappando la porta 8080 (precedentemente esposta sul *dockerfile*) sulla porta scelta. Inoltre, aggiungere il flag *--restart always* per permettere il riavvio automatico del container in caso di crash, eccezioni o interruzioni anomale.

4. Raggiungere il core mediante il percorso *http(s)://BASE_URL:PORT* in cui *BASE_URL* rappresenta il percorso o l'IP dell'host, mentre *PORT* è la porta scelta in fase di mapping.

Il deployment è così terminato, non resta che far puntare i client web e mobile al percorso del core per avere l'intero sistema funzionante.

Con la procedura appena descritta termina la discussione riguardo il progetto i tesi. I paragrafi seguenti riguarderanno gli spunti di miglioramento e le conclusioni.

7.2 Spunti di miglioramento

Quanto illustrato nei capitoli precedenti è un core adatto a supportare le diverse funzionalità delle quattro macroaree in maniera abbastanza soddisfacente. È tuttavia possibile pensare a dei miglioramenti, in termini architetturali, di supporto, di evoluzione e di performance e ottimizzazione.

Di seguito alcune idee:

- Pensare ad un recommender system ibrido, sia *theory-based* ma anche *data-driven*, introducendo delle tecniche di data science in grado di analizzare i dati prodotti dagli assistiti medici, estrarne i pattern significativi e produrre recommendations non unicamente basate sulle soglie, ma anche sulle relazioni rilevate tra i dati. Un interessante idea di studio riguarda la correlazione tra determinati valori delle macroaree e le informazioni anagrafiche e mediche del paziente (sesso, età, patologie, massa etc.). Tale modifica richiederebbe una rivisitazione radicale di tutto il modulo.

- Espandere il recommender system, aggiungendo un numero maggiore di regole a parametri multipli o a macroaree multiple. Tali regole potrebbero essere ricavate non solo dalla letteratura, ma anche in seguito ad interviste condotte con medici di differenti ospedali, in modo tale da estrarre una tendenza comune a diversi presidi ospedalieri, in differenti regioni italiane o europee.

- Estendere il supporto a nuovi tracker e smartwatch proprietari appartenenti a differenti brand, pensando di far interagire i differenti server di terze parti con quello di HealthApp, facendo confluire in quest'ultimo tutti i dati prodotti e memorizzati in maniera consistente. In questo modo, nonostante la diversità degli apparati, il paziente ed il medico avranno a che fare sempre con dati omogenei.

- Estendere il supporto a tracker open-source (es. PineTime), eliminando totalmente l'interazione con i server di terze parti, facendo confluire i dati prodotti dagli assistiti direttamente sul backend di HealthApp.

- Estendere il supporto a nuove tipologie di dispositivi come misuratori di pressione, saturatori di ossigeno, sensori di glicemia ecc., espandendo la macroarea salute e rilevando ulteriori parametri, oltre alla frequenza cardiaca.

- Modificare il modello del database, passando da uno relazionale ed SQL ad uno document-based e non-SQL, riducendo l'occupazione di spazio per le differenti tipologie di dati che possono assumere il valore *null* (si pensi alle analisi mediche o al cibo) e che non sarebbero memorizzati in un database a documenti.

- Introdurre un sistema per la gestione del database e non più un semplice file, eseguito in un container indipendente ed in grado di comunicare esclusivamente con il backend di HealthApp.

È chiaro che il progetto è in continua evoluzione e parte di queste idee saranno messe in pratica dai colleghi che si occuperanno di continuare lo sviluppo. Ad ogni modo, il core attuale è ben funzionante e può fornire da punto di partenza per la sperimentazione in reale di tutto l'applicativo.

7.3 Conclusioni

A differenza di tutti gli altri paragrafi, ho scelto di adottare in quest'ultimo un linguaggio informale. Parlerò in breve della mia esperienza, di ciò che ho imparato e soprattutto delle emozioni che ho provato.

Nel momento in cui mi misi a cercare un progetto di tesi non avevo le idee ben precise su cosa volessi fare ma un obiettivo: trovare qualcosa che non fosse banale, che non fosse troppo astratto ma che potesse, un giorno, offrire un aiuto concreto alla società. I buoni rapporti e i contatti frequenti con il Professor Morisio, uniti al periodo di pandemia trascorso mi spinsero ad abbracciare un progetto vicino, oltre che all'informatica, al mondo della medicina: HealthApp in collaborazione con l' Azienda Ospedaliera di Verona.

Immediatamente mi sentii estasiato dalla sfida della progettazione di un applicativo che sotto certi aspetti poteva sembrare simile a decine di applicazioni già esistenti, ma che è arrivato a presentare delle novità assolute, come il recommender system utilizzato per analizzare e monitorare il benessere degli assistiti medici.

I mesi di sviluppo mi hanno insegnato molto in quanto, salvo alcuni progetti affrontati nel corso degli anni universitari, non avevo avuto esperienze significative di teamworking. Ho imparato a lavorare insieme a persone con caratteri, esigenze e tempistiche totalmente differenti dalle mie. A volte attendere i risultati di qualcun altro è risultato snervante, altre volte rilassante, in quanto mi ha permesso di dedicare maggior tempo alla mia vita, alle relazioni interpersonali, ai miei hobby, ai viaggi e a tutto ciò che era passato in secondo piano durante quasi sei anni di intensi studi universitari.

Alcune volte mi sono sentito in ansia di produrre per tempo un risultato, di rispettare le scadenze, di presentare qualcosa durante i meeting settimanali, ma con i mesi ho imparato a gestire al meglio il mio

tempo, la mia produttività e l'ansia è sempre di più diminuita in favore di una migliore gestione dello stress.

Ho imparato che dietro la corretta riuscita di un progetto è fondamentale la comunicazione e l'ascolto. È stato importante ascoltare i punti di vista ed i consigli altrui, anche se a volte andavano in una direzione differente rispetto a quella da me immaginata. È stato fondamentale evitare i conflitti, favorendo sempre la collaborazione, la ricerca dei compromessi e delle soluzioni valide. È stato altresì importante esporre sempre il proprio punto di vista, chiedere quando si avevano dei dubbi e non dare mai nulla per scontato. Più la comunicazione era efficace, maggiore è stata la qualità dei risultati.

Ho trovato un ambiente decisamente amichevole e collaborativo, in cui il professore ed i colleghi si sono sempre mostrati disponibili in caso di difficoltà. Inoltre, la totale flessibilità nell'organizzazione dei luoghi e degli orari di lavoro ha reso il progetto coinvolgente e totalmente gestibile, senza mai risultare noioso.

Per me HealthApp rappresenta la conclusione di un periodo intenso della mia vita, fatto di gioie, dolori, sorrisi, lacrime, felicità e tristezza che definiscono la mia esperienza universitaria. Esperienza che porterò sempre dentro di me e a cui sarò eternamente grato per la maturità e le conoscenze che mi ha conferito.

Bibliografia

- Inchauspé J., La rivoluzione del glucosio. Come controllare i livelli di zucchero nel sangue per perdere peso, abbattere la fame e avere più energia. Con autotest e 10 sorprendenti trucchi nutrizionali, Milano, Antonio Vallardi Editore, 13 settembre 2022, I Edizione, 288 pagine, Italiano, titolo originale Glucose Revolution, traduzione Ferloni N.

- 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

Sitografia

[1] PrepInsta, 3 Tier Architecture,
https://www.prepinsta.com/dbms/3-tier-architecture/,
consultato il 28/10/2022.

[2] IBM Cloud Education, Three-Tier Architecture,
https://www.ibm.com/cloud/learn/three-tier-architecture,
consultato il 28/10/2022.

[3] Javarevisited, Difference between @Component, @Service, @Controller, and @Repository in Spring ,
https://javarevisited.blogspot.com/2017/11/difference-between-component-service.html ,
consultato il 29/10/2022.

[4] Collings Tom, Medium , Controller-Service-Repository,
https://tom-collings.medium.com/controller-service-repository-16e29a4684e5,
consultato il 29/10/2022.

[5] Ministero della salute, Calcolo indice di massa corporea,
https://www.salute.gov.it/portale/nutrizione/dettaglioIMCNutrizione.jsp?lingua=italiano&id=5479&area=nutrizione&menu=vuoto,
consultato il 09/01/2023.

[6] U.S. Department of Health & Human Services, Calculating BMI Using the Metric System,
https://www.cdc.gov/nccdphp/dnpao/growthcharts/training/bmiage/page5_1.html,
consultato il 09/01/2023.

[7] U.S. Department of Health & Human Services, Calculating BMI Using the Metric System, https://www.cdc.gov/nccdphp/dnpao/growthcharts/training/bmiage/page5_1.html, consultato il 09/01/2023.

[8] Fitbit Developer Console, https://dev.fitbit.com/, consultato il 12/01/2023.

[9] Fitbit, Application design, https://dev.fitbit.com/build/reference/web-api/developer-guide/application-design/, consultato il 12/01/2023.

[10] T. Enrico, Teranet, Che cos'è e come funziona il protocollo OAuth 2.0? , https://www.teranet.it/introduzione-ad-oauth-2, consultato il 12/01/2023.

[11] FatSecret, The FatSecret platform API, https://platform.fatsecret.com/api/, consultato il 12/01/2022.

[12] Jason Kincaid, TechCrunch, https://techcrunch.com/2009/08/14/fatsecret-looks-to-become-a-central-hub-for-nutrition-data-with-new-api/, consultato il 12/01/2022.

[13] FatSecret, Authentication with OAuth 1.0 , https://platform.fatsecret.com/api/Default.aspx?screen=rapiauth1 , consultato il 12/01/2022.

[14] Internet Standard, Rfc Editor, Uniform Resource Identifier (URI): Section 2.1, https://www.rfc-editor.org/rfc/rfc3986#section-2.1, consultato il 12/01/2022

[15] Internet Standard, Rfc Editor, Multipurpose Internet Mail Extensions (MIME): Section 6.8, https://www.rfc-editor.org/rfc/rfc2045#section-6.8, consultato il 12/01/2022

[16] Nowak M., Hyperion, What is a rules engine, https://www.hyperon.io/blog/what-is-a-rules-engine, consultato il 05/12/2022.

[17] Wikipedia, Event Condition Action, https://en.wikipedia.org/wiki/Event_condition_action, consultato il 05/12/2022

[18] DecisionRules, Decision Rules Pricing,
https://www.decisionrules.io/pricing/public-cloud,
consultato il 05/12/2022

[19] Hyperion, Hyperion Pricing,
https://www.hyperon.io/pricing,
consultato il 05/12/2022

[20] Drools, Drools Documentation,
https://docs.drools.org/7.73.0.Final/drools-docs/html_single/index.html,
consultato il 05/12/2022

[21] StackOverflow, Drools stateless vs stateful session ,
https://stackoverflow.com/questions/17175037/droolsstateless-vs-stateful-knowledge-session ,
consultato il 05/12/2022

[22] Hawes K., Kansas University Medical Center, 10,000 steps might really be the 'magic pill' everyone is seeking,
https://www.kumc.edu/about/news/news-archive/jama-study-ten-thousand-steps.html,
consultato il 21/01/2023.

[23] SINU (Società Italiana di nutrizione umana), Fabbisogno energetico medio (AR) in età geriatrica,
https://sinu.it/2019/07/09/fabbisogno-energetico-medio-ar-in-eta-geriatrica/,
consultato il 21/01/2023.

[24] Censani S., Blog Salugea, Rapporto colesterolo totale e HDL: l'indice di rischio cardiovascolare,
https://blog.salugea.com/colesterolo-e-trigliceridi/rapporto-colesterolo-totale-e-hdl,
consultato il 21/01/2023.

[25] Wikipedia, Rapporto BUN-creatinina,
https://it.wikipedia.org/wiki/Rapporto_BUN-creatinina,
consultato il 21/01/2023.

[26] Tomasi C., COLESTEROLO HDL E TRIGLICERIDI: occhio al rapporto,
https://www.cristinatomasi.com/colesterolo-hdl-e-trigliceridi-occhio-al-rapporto/,
consultato il 21/01/2023

[27] Fattorie Girau, Calcolo calorie,
https://www.fattoriegirau.com/magazine/filosofia-alimentare/educazione-alimentare/calcolo-calorie/,
consultato il 22/01/2023

[28] Project Invictus, Quanti carboidrati al giorno,
https://www.projectinvictus.it/quanti-carboidrati-al-giorno/,
consultato il 22/01/2023

[29] Project Invictus, Quante proteine assumere,
https://www.projectinvictus.it/quante-proteine-assumere/,
consultato il 22/01/2023

[30] Project Invictus, Quanti grassi al giorno,
https://www.projectinvictus.it/quanti-grassi-al-giorno/,
consultato il 22/01/2023

[31] My personal trainer,
Percentuali di Grassi Saturi e Insaturi nella Dieta ,
https://www.my-personaltrainer.it/nutrizione/grassi-saturi-insaturi1.html, consultato il
22/01/2023

[32] Fondazione Veronesi, Grassi trans: addio per tutti entro il 2023? ,
https://www.fondazioneveronesi.it/magazine/articoli/alimentazione/grassi-trans-lorganizzazione-mondiale-della-sanita-vuole-eliminarli-entro-il-2023,
consultato il 22/01/2023

[33] Armolipid, Cibi che abbassano il colesterolo,
https://www.armolipid.it/it-it/cambiamenti-nello-stile-di-vita/mangiare-sano/colesterolo-alimentare,
consultato il 22/01/2023

[34] Istituto superiore di sanità, Prevenzione delle malattie cardiovascolari,
https://www.cuore.iss.it/prevenzione/pdf/sale_broch2pag.pdf,
consultato il 22/01/2023.

[35] Istituto superiore di sanità, Il consumo di sodio e potassio in Italia,
https://www.cuore.iss.it/prevenzione/ProgettoMinisal,
consultato il 22/01/2023.

[36] Istituto superiore di sanità, Sali Minerali,
https://www.epicentro.iss.it/sali/macroelementi,
consultato il 22/01/2023.

[37] Humanitas, Ferro,
https://www.humanitas.it/enciclopedia/sali-minerali/ferro/,
consultato il 22/01/2023.

[38] Project Invictus, Fibre alimentari,
https://www.projectinvictus.it/fibre-alimentari/,
consultato il 22/01/2023

[39] Rita V., Quotidiano Sanità,
https://www.quotidianosanita.it/scienza-e-farmaci/articolo.php?articolo_id=26470,
consultato il 22/01/2023

[40] Project Invictus, Zucchero al giorno,
https://www.projectinvictus.it/zucchero-al-giorno/,
consultato il 22/01/2023

[41] Humanitas, Vitamina A (retinolo),
https://www.humanitas.it/enciclopedia/vitamine/vitamina-a-retinolo/,
consultato il 22/01/2023

[42] Farmacia del Leone, La vitamina C alleata per le difese immunitarie,
https://www.farmaciadelleonesalerno.it/la-vitamina-c-alleata-per-le-difese-immunitarie/,
consultato il 22/01/2023

[43] Humanitas, Vitamina D,
https://www.humanitas.it/enciclopedia/vitamine/vitamina-d/,
consultato il 22/01/2023

[44] Ministero della salute, SODIO, POTASSIO E IODIO NELLA DIETA DEGLI ITALIANI,
https://sinu.it/wp-content/uploads/2019/07/MINISAL-sintesi-risultati.pdf,
consultato il 22/01/2023

[45] My personal trainer, Fabbisogno di grassi,
https://www.my-personaltrainer.it/grassi-fabbisogno1.html,
consultato il 22/01/2023

[46] Meli E., Corriere Della Sera, Grassi, zuccheri e fibre: qual è il giusto rapporto? Detti e contraddetti,
https://www.corriere.it/salute/nutrizione/17_novembre_29/grassi-zuccheri-fibre-giusto-rapporto-ff8617d4-d524-11e7-85e2-6290f9ff2b20.shtml,
consultato il 22/01/2023

[47] Diete Facili, L'IMPORTANZA DEL BILANCIO CALORICO NEL DIMAGRIMENTO,
https://www.dietefacili.com/dimagrire/importanza-bilancio-calorico.html,
consultato il 22/01/2023

[48] Gindro R., The Wom Healthy, Frequenza cardiaca normale, a riposo e massima,
https://healthy.thewom.it/esami-e-analisi/frequenza-cardiaca/,
consultato il 22/01/2023

[49] My personal trainer, Quante Ore Bisogna Dormire in Base all'Età,
https://www.my-personaltrainer.it/benessere/quante-ore-dormire-eta.html,
consultato il 22/01/2023

[50] Dica 33, Tabelle del peso ideale di donne e uomini,
https://www.dica33.it/peso-forma/tabelle-peso-ideale/tabelle-peso-ideale-donne-uomini.asp,
consultato il 22/01/2023

[51] Humanitas Gavezzani, BMI – Body Mass Index,
https://www.gavazzeni.it/enciclopedia/prevenzione/obesita-calcola-il-tuo-bmi/,
consultato il 22/01/2023

[52] MÖBIUS, Kaggle, FitBit Fitness Tracker Data,
https://www.kaggle.com/datasets/arashnic/fitbit,
consultato il 06/02/2023.

[53] Simula, PMData - A lifelogging dataset of 16 persons during 5 months using Fitbit, Google Forms and PMSys.,
https://datasets.simula.no/pmdata/,
consultato il 06/02/2023

[54] CSSE JHU, An Open-Source Dataset on Dietary Behaviors and DASH Eating Plan Optimization Constraints,
https://github.com/CSSEHealthcare/Dietary-Behavior-Dataset,
consultato il 06/02/2023

[55] Wikipedia, Docker,
https://it.wikipedia.org/wiki/Docker,
consultato il 13/02/2023

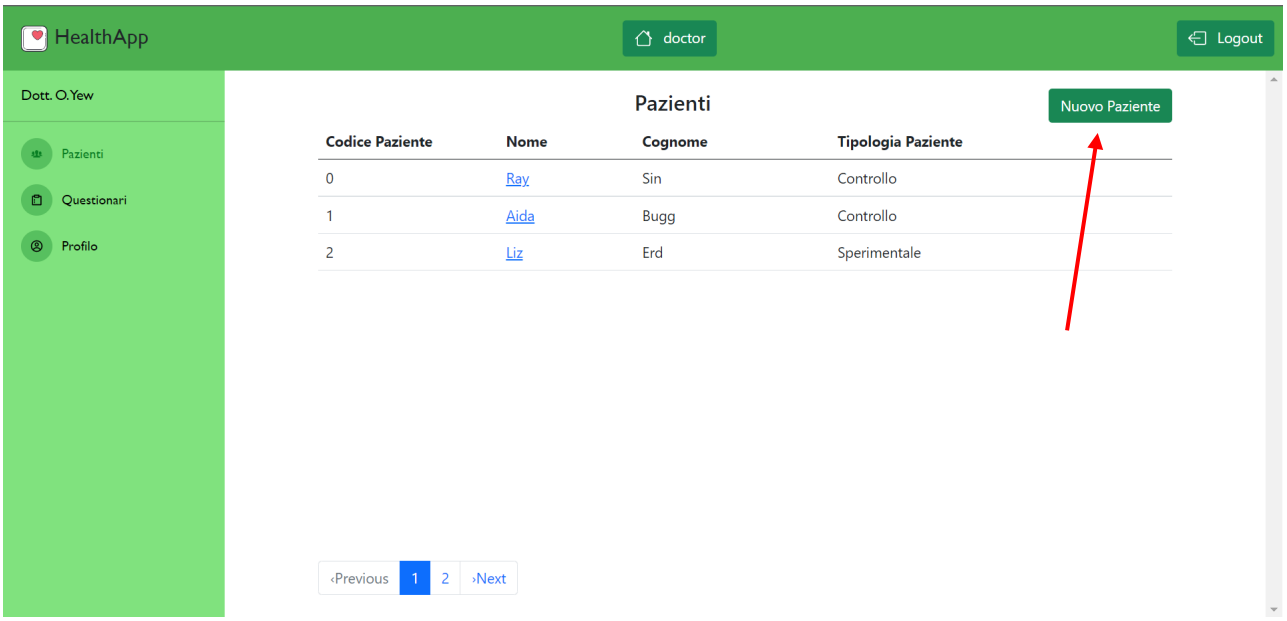
[56] Wikipedia, JAR (formato di file),
https://it.wikipedia.org/wiki/JAR_(formato_di_file),
consultato il 13/02/2023

Allegato 1. Guida registrazione nuovo paziente su HealthApp

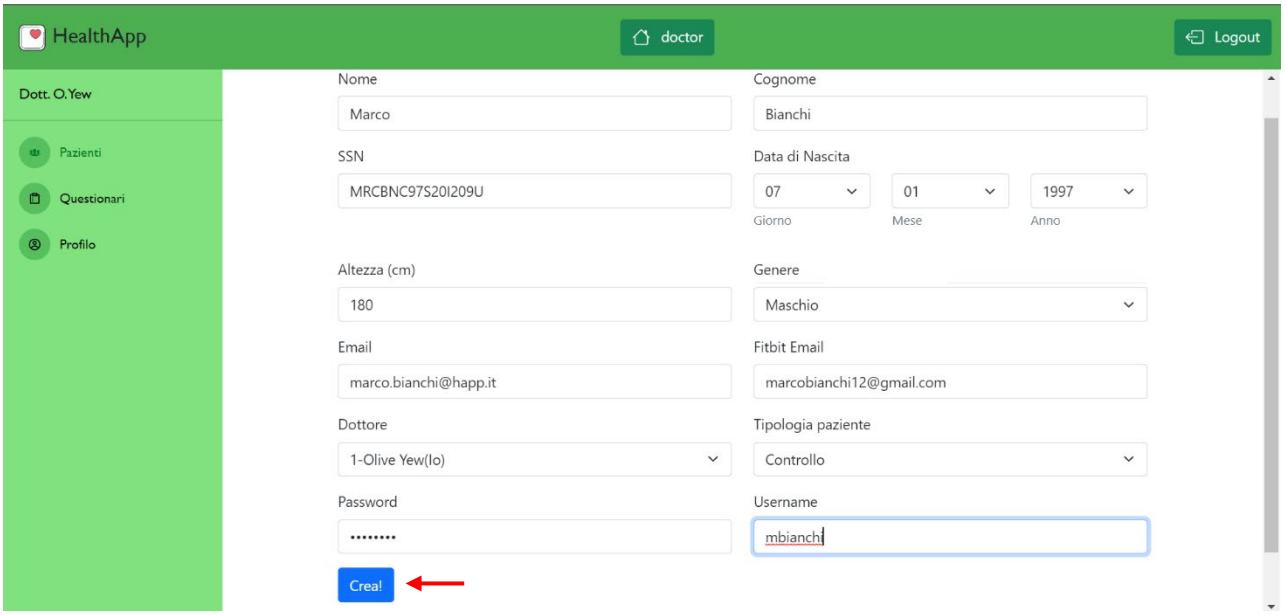
1. Raggiungere il sito e accedere con le seguenti credenziali:

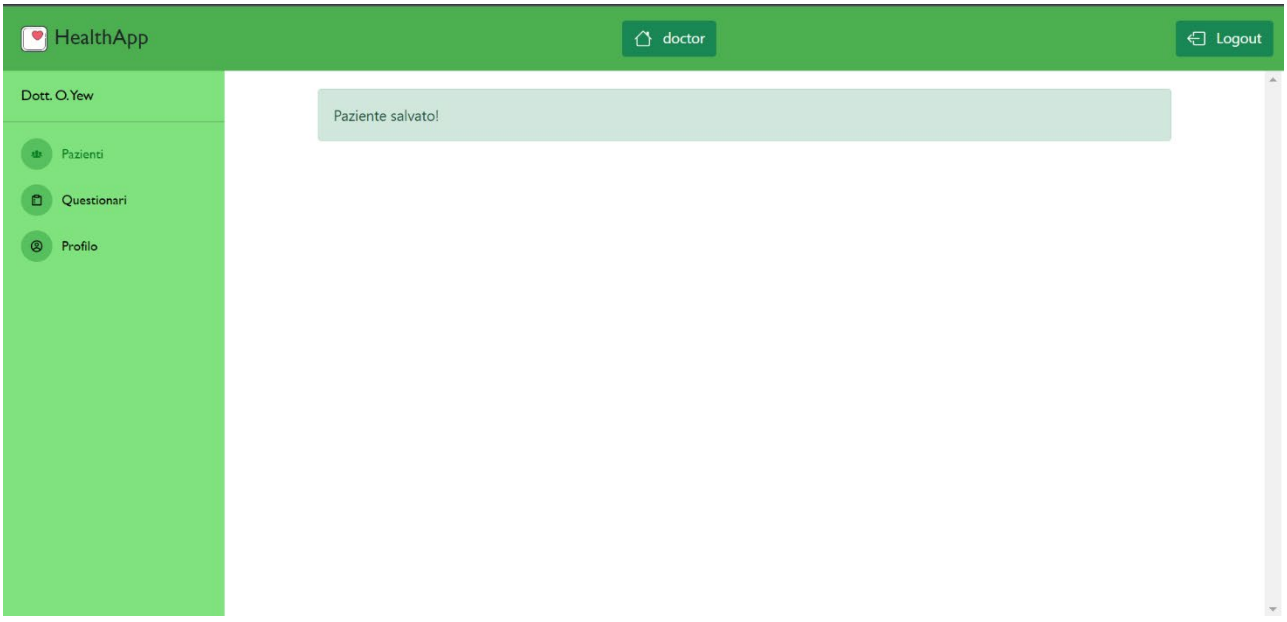
   Username: doctor
   Password: password

2. Cliccare il pulsante *"Nuovo Paziente"*



3. Compilare il modulo inserendo i dati del paziente e la mail dell'account Fitbit dello stesso. Scegliere username e password e cliccare il pulsante *"Crea"*.

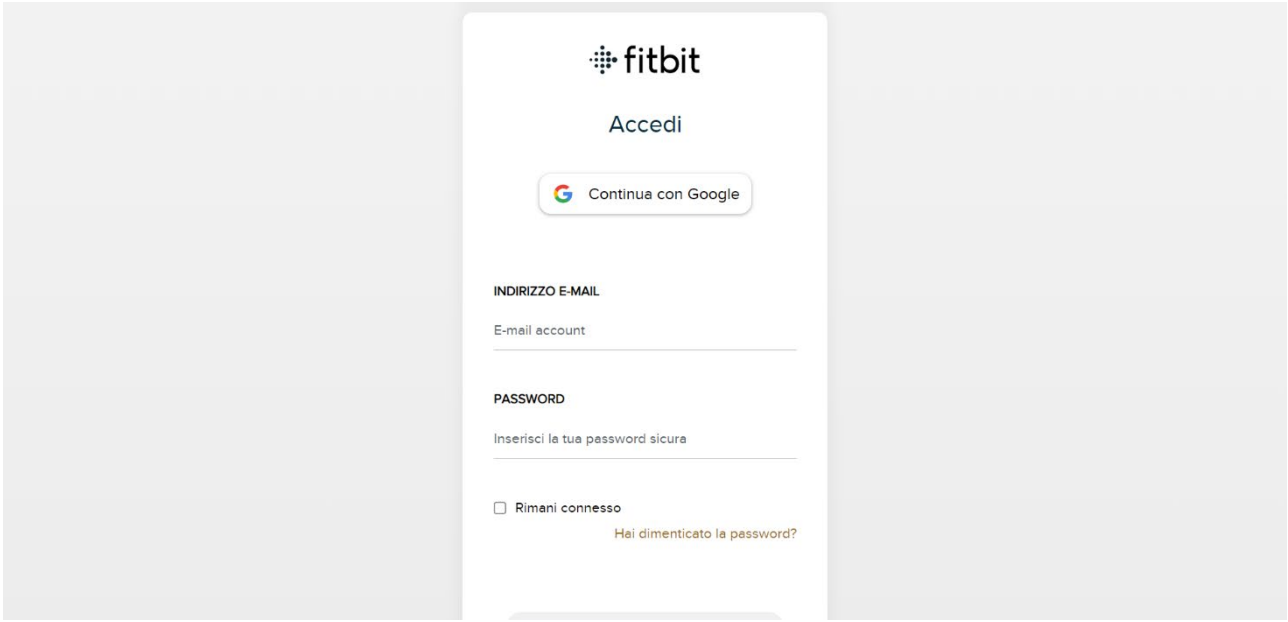




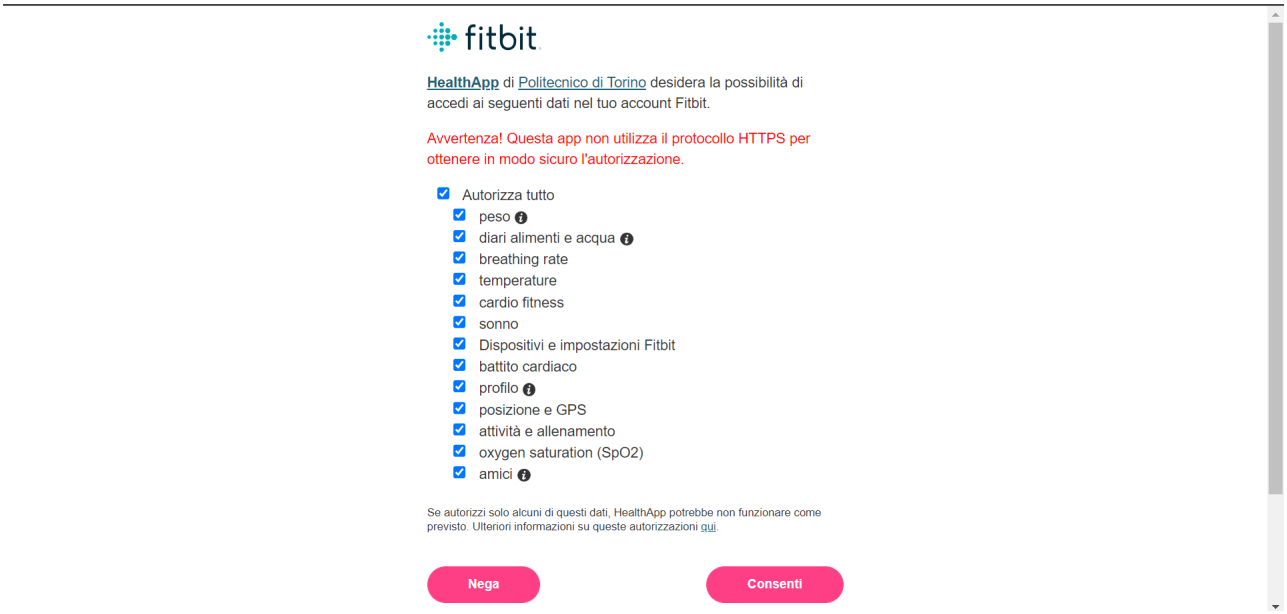
4. Fare logout dall'account del medico

5. Accedere con l'account appena creato. Per il primo login, si verrà reindirizzati al sito Fitbit.

6. Accedere con l'account Fitbit (del paziente)

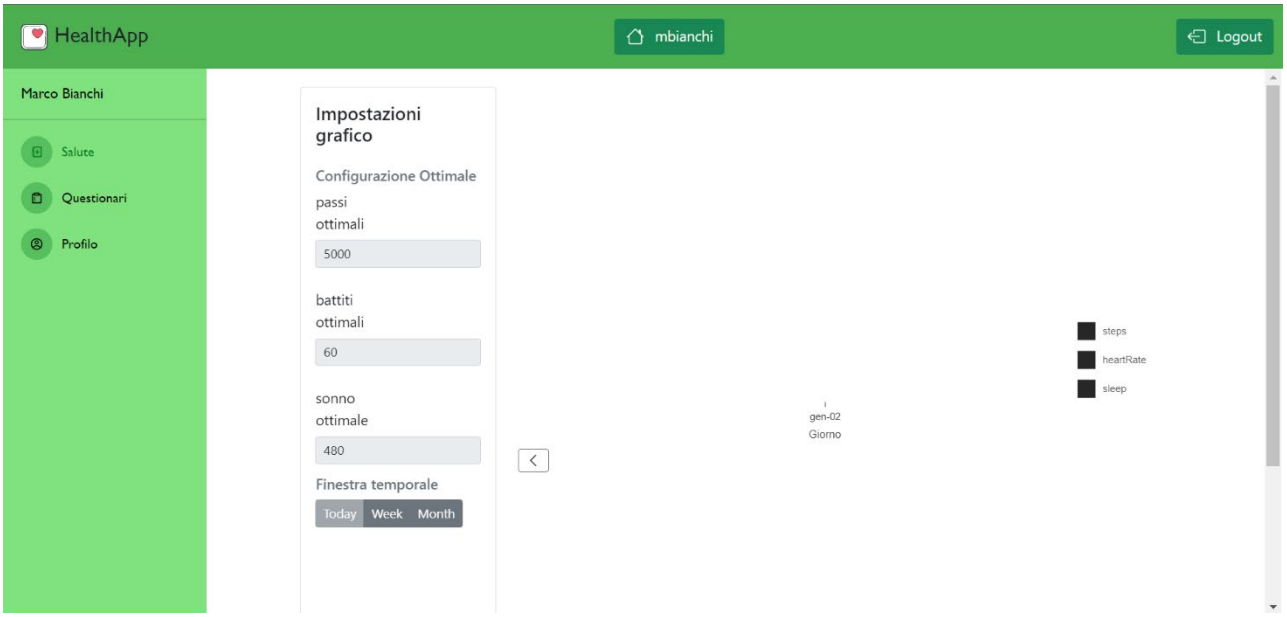


7. Concedere tutti i permessi e le autorizzazioni necessarie



8. Se il procedimento è andato a buon fine, si verrà reindirizzati nuovamente su HealthApp

9. Eseguire il logout e successivamente il login con l'account appena creato in modo tale da permettere l'aggiornamento di tutti i dati



10. È possibile utilizzare l'account paziente creato

Ringraziamenti

Ringrazio il Professore Maurizio Morisio per avermi permesso di contribuire a questo progetto favorendo la mia crescita in ambito professionale e umano. Lo ringrazio inoltre per la disponibilità sempre dimostrata e per aver creato un ambiente di lavoro sano, flessibile e di aiuto reciproco in cui ognuno ha potuto esprimere liberamente il proprio punto di vista.

Ringrazio Milena per avermi accompagnato in questi anni contribuendo a forgiare il mio carattere e a farmi crescere tanto dal punto di vista personale e professionale. La tua sensibilità e il tuo approccio alla vita mi hanno insegnato e fatto imparare molto, ma non ti sono grato solo per quello. Ti ringrazio per tutti i piccoli gesti quotidiani, per essere stata al mio fianco nei periodi più duri, per avermi dato la forza e il coraggio di fare scelte che da solo non avrei mai fatto. Ti ringrazio per darmi sempre il tuo meglio nei momenti di gioia e di dolore, per tutto quello che fai per me e semplicemente perché mi fai sentire completo, non potevo desiderare una persona migliore nella mia vita. Ti Amo.

Ringrazio Mamma per aver insistito affinché io seguissi la mia strada al Politecnico sostenendomi e dandomi forza fin dal primo giorno e non smettendo mai di credere in me. Nonostante la distanza, non mi hai mai fatto sentire solo e ci sei sempre stata, a qualsiasi ora del giorno e della notte. Adesso il traguardo è stato raggiunto ma è soprattutto grazie a te se sono riuscito a superare la maggior parte delle difficoltà senza mollare mai. Grazie per esserci stata sempre e per continuare a farlo. Ti voglio bene.

Ringrazio Papà per aver seguito sempre con interesse tutto il mio percorso universitario ma soprattutto per avermi insegnato tanto dal punto di vista umano. Grazie a te ho imparato che non bisogna sempre ambire alla perfezione assoluta per sentirsi realizzati. Mi hai insegnato a sognare, ad essere ottimista e ambizioso non solo dal punto di vista professionale ma anche personale. È in particolare grazie ai tuoi consigli se sono riuscito a vivere quest'esperienza a 360°. Grazie per non avermi fatto mancare mai niente e per aver reso possibile il raggiungimento di questo risultato. Ti voglio bene.

Ringrazio i Nonni tutti per l'affetto, l'incoraggiamento e l'interesse sempre dimostrato quando ci trovavamo a parlare di università, anche se alcuni concetti potevano sembrare loro astratti. In particolare, ringrazio Nonno Rocco per essere stato la colonna portante della mia esistenza e per avermi trasmesso i valori del lavoro duro e dell'inesauribile forza di volontà. Mi mancherà chiamarti subito dopo l'esito di un esame per sentire la tua gioia venir fuori come se fossi stato tu a sostenerlo. Ringrazio inoltre Nonno Peppe per avermi fatto capire l'importanza dello studio e dell'applicazione costante. Mi manca vedere uscire fuori la tua commozione e la tua sensibilità di fronte al tema dell'istruzione.

Ringrazio tutti i miei familiari per la vicinanza e la costante presenza dimostratami anche aldilà del percorso universitario. Avete contribuito allo sviluppo di importanti valori in me, come quelli dell'unità e della famiglia. Avrete sempre un posto nel mio cuore.

Ringrazio i miei coinquilini Marco, Edoardo ed Emanuele per essere stati i compagni e soprattutto gli amici con cui ho condiviso tante esperienze e semestri durante il mio percorso a Torino. Condividere la casa, i momenti di stacco, gli scherzi, lo studio, le sessioni, i pasti, aiutandosi e dandosi consigli reciprocamente in tutto ha sicuramente contribuito a rendere più semplici questi anni. Ringrazio inoltre Marco per le mille peripezie vissute a lezione e fuori. Non potevo e non posso immaginare un'esperienza con dei coinquilini migliori, mi porto dentro dei ricordi indimenticabili.

Ringrazio Arya per la sincerità, la disponibilità al confronto e la collaborazione che c'è stata in questi anni. Sono molto contento del rapporto di amicizia vero che si è instaurato tra di noi e che prosegue al di fuori dei confini del Politecnico. Il tuo impegno, la tua forza di volontà e la tua metodologia di lavoro hanno dato un contributo molto importante alla mia crescita professionale. I tanti discorsi affrontati insieme hanno invece contribuito ad aprire la mia mentalità e ad offrirmi una visione differente e spesso migliore dei problemi.

Ringrazio Andrea e Federico per essere stati presenti nel periodo difficile della pandemia vissuto lontano da Torino e in diverse altre occasioni. Sicuramente diverse giornate che sembravano interminabili sono passate molto più in fretta. Grazie per tutti i momenti passati insieme in questi anni.