POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di laurea magistrale

IBD Tool: applicazione web per il monitoraggio dei pazienti con malattie croniche intestinali

Relatore:
Prof. Carla Fabiana Chiasserini

Co-relatore:
Prof. Guido Pagana

Candidato
Marcello Figus

Luglio 2021

Abstract

Le *Inflammatory Bowel Diseases* (in sigla IBD) sono delle malattie infiammatorie croniche del tratto gastrointestinale. Tra queste le più diffuse sono la rettocolite ulcerosa e la malattia di Crohn. Le IBD spesso causano diarrea, dolore addominale, sanguinamento rettale ed un'altra ampia varietà di sintomi. In quanto malattie croniche, il paziente che ne soffre non raggiunge mai una guarigione completa, ma alterna periodi di remissione a periodi in cui i sintomi si presentano in forma acuta. La terapia medica, di conseguenza, varia dai periodi di remissione a quelli di recidiva. Per questo motivo, il paziente ha bisogno di essere monitorato nel tempo, in modo da permettere ai medici di agire in modo tempestivo onde evitare la degenerazione dello stato di salute del paziente. Particolarmente adatta a questo scopo è l'applicazione di servizi di telemedicina. Per telemedicina si intende l'erogazione di servizi di assistenza sanitaria in situazioni in cui il professionista della salute e il paziente (o due professionisti) non si trovino nello stesso luogo, utilizzando prevalentemente tecnologie ICT (*Informationand Communication Technologie*).

Il lavoro svolto durante questa tesi consiste nell'analisi e la conseguente realizzazione di possibili soluzioni a supporto di medici e pazienti nel monitoraggio delle IBD. La tesi è stata sviluppata in smart working presso *LINKS Foundation*, in collaborazione con il *Dipartimento di Gastroenterologia dell'Azienda Ospedaliera Ordine Mauriziano di Torino*. Questo progetto nasce come proseguimento di un precedente lavoro di tesi del Politecnico di Torino che ha costruito le basi di IBD Tool. IBD Tool è un'applicazione web che realizza un servizio di telemedicina attraverso la somministrazione periodica di questionari per il monitoraggio di pazienti con IBD. L'applicazione presentava le funzioni base per svolgere le seguenti azioni:

- registrazione differenziata per pazienti e medici;
- possibilità di compilare vari tipi di questionari;
- capacità di inviare periodicamente i questionari in base alla patologia del paziente.

L'obiettivo di questa tesi è perfezionare le funzionalità già disponibili e incrementare le operazioni possibili sulla piattaforma. I risultati ottenuti sono prevalentemente

focalizzati nel produrre delle funzionalità che aumentino le possibilità di comunicazione tra utenti e che permettano ai medici una più facile gestione del lavoro. Per ottenere questi risultati sono state realizzate diverse pagine per il controllo dei questionari e dei pazienti. Inoltre è stata creata una chat integrata nell'applicazione, in modo da permettere la comunicazione e lo scambio di informazioni e documenti tra medici e pazienti. Le tecnologie utilizzate sono *Angular* e *Spring* per quanto riguarda lo sviluppo del front-end e del back-end, mentre per il database si è scelto *MongoDB*.

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 Telemedicina: un valido alleato del sistema sanitario

Per telemedicina si intende una modalità di erogazione di servizi di assistenza sanitaria, tramite il ricorso a tecnologie innovative, in particolare alle Information and Communication Technologies (ICT), in situazioni in cui il professionista della salute e il paziente (o due professionisti) non si trovino nella stessa località [1]. La sua effettiva applicazione può essere basata su molteplici tecnologie, tra cui e-mail, consultazioni tramite chiamate vocali o video-chiamate, scambio di documentazioni digitali, monitoraggio attraverso sensori oppure applicazioni accessibili tramite computer o smartphone. Durante la pandemia di COVID-19, l'uso della telemedicina ha subito una forte accelerazione, infatti ad una diminuzione delle visite in presenza corrisponde una riduzione della possibilità di diffusione del virus durante le visite ospedaliere [3]. Secondo i dati aggiornati al 2 Maggio gli attuali positivi sono 430.906, di questi 20.869 sono ricoverati presso una struttura ospedaliera [4].

La telemedicina può essere molto utile come supporto dei medici di base, consentendo ai pazienti di ricevere prescrizioni elettroniche per farmaci o per visite specialistiche. È anche altrettanto interessante la sua applicazione nella gestione di disturbi cronici (es. malattie IBD, diabete, l'ipertensione e alcune malattie polmonari a lungo termine) o nei pazienti che subiscono un trattamento palliativo. Per quanto riguarda i disturbi cronici, dopo aver effettuato una diagnosi e impostato una terapia, qualsiasi modifica del regime di trattamento potrà essere agevolmente apportata anche da remoto [5]. Nel caso in cui invece non sia stata ancora effettuata una diagnosi, la telemedicina può mostrarsi utile in alcune applicazioni in cui è possibile identificare problemi visionando il corpo del paziente attraverso videochiamate o fotografie, come ad esempio per le consulenze dermatologiche. Questa modalità però potrebbe avere dei problemi nel caso in cui le immagini fornite fossero di bassa qualità [6].

Le visite in presenza rimangono un punto insostituibile della medicina perché consentono al medico di individuare con maggiore precisione dei dettagli che possono essere fondamentali per una corretta diagnosi. Nonostante la telemedicina non possa sostituire in maniera completa le visite tradizionali, rimane uno strumento molto valido per ridurre il numero di visite ambulatoriali, evitare il sovraffollamento dei posti letto e ridurre le spese sanitarie permettendo una maggiore concentrazione di fondi in settori cardine [3].

1.1.1 Le strutture ospedaliere in Italia

Le strutture ospedaliere negli ultimi anni hanno subito una serie di tagli finanziari. Questo fenomeno può essere spiegato dal tentativo di cambiare l'organizzazione della sanità pubblica nazionale, al fine di ridurre i costi ospedalieri [7]. La tendenza è quella di spostare sempre più l'assistenza dalle strutture sanitarie (ad esempio gli Ospedali) verso procedure fornite principalmente su base day hospital o attraverso una forma a domicilio (es. la telemedicina), riducendo in questo modo l'esigenza di avere un alto numero di posti letto per il trattamento dei pazienti. Analizzando i dati più recenti resi disponibili dal Ministero della Salute italiano (dal 2010 al 2017) si può notare che c'è stata un'importante riduzione dei posti letto nella maggior parte dei reparti ospedalieri [8], mentre si è riscontrata una tendenza inversa per le unità di terapia intensiva [9]. Generalmente i posti letto ospedalieri sono gestiti in modo efficiente in tutto il paese, soprattutto nelle regioni settentrionali italiane dove le pratiche vengono gestite con turni rapidi e senza lasciare letti vuoti durante l'anno [10].

1.1.2 Classificazione dei servizi di telemedicina

La telemedicina può essere suddivisa nelle seguenti categorie:

Telemedicina specialistica

Con telemedicina specialistica si intende il servizio medico a distanza all'interno di una specifica disciplina medica. Può avvenire tra medico e paziente oppure tra medici e altri operatori sanitari [1]. Può essere svolta in diverse modalità:

- *Televisita*: atto sanitario in cui il medico interagisce a distanza con un paziente assistito in presenza da un secondo operatore sanitario. Durante la Televisita possono essere prescritti farmaci o di cure.

- *Teleconsulto*: è una consulenza a distanza fra medici che permette di chiedere consiglio per l'indicazione di diagnosi o la scelta di una terapia relativa ad un paziente

- *Telecooperazione sanitaria*: è un'assistenza fornita tra medici o operatori sanitari impegnati in un atto sanitario (es, consulenza fornita per un soccorso d'urgenza).

Telesalute

Per Telesalute si intende il servizio di assistenza che mette in collegamento diretto pazienti e medici al fine di assistere nella diagnosi, monitroaggio e gestione della malattia. I dati necessari al Telemonitoraggio del paziente, possono essere registrati in maniera automatizzata dal paziente oppure da un operatore sanitario. Richiede un ruolo attivo del medico nella presa in carico del paziente e anche un ruolo attivo del paziente nell'autocura [1].

Teleassistenza

La Teleassistenza è un sistema socio-assistenziale per la presa in carico di una persona anziana o fragile a domicilio. L'assistenza viene gestita tramite allarmi, sistemi di emergenza, chiamate di supporto da parte di un centro servizi [1].

<table>
<tr><th colspan="5">TELEMEDICINA</th></tr>
<tr><th colspan="2">CLASSIFICAZIONE</th><th>AMBITO</th><th colspan="2">PAZIENTI</th></tr>
<tr><td rowspan="3">TELEMEDICINA SPECIALISTICA</td><td>TELE VISITA</td><td rowspan="3">Sanitario</td><td rowspan="3">Può essere rivolta a patologie acute, croniche, a situazioni di post-acuzie</td><td>Presenza attiva del paziente</td></tr>
<tr><td>TELE CONSULTO</td><td>Assenza del paziente</td></tr>
<tr><td>TELE COOPERAZIONE SANITARIA</td><td>Presenza del paziente in tempo reale</td></tr>
<tr><td>TELE SALUTE</td><td></td><td>Sanitario</td><td>È prevalentemente rivolta a patologie croniche</td><td>Presenza attiva del paziente</td></tr>
<tr><td>TELE ASSISTENZA</td><td></td><td>Socio-assistenziale</td><td>Può essere rivolta ad anziani e fragili e diversamente abili</td><td></td></tr>
</table>

Figura 1.1. Classificazione dei servizi di telemedicina [1]

1.2 IBD

Le IBD (Inflammatory Bowel Diseases) sono delle malattie infiammatorie croniche del tratto gastrointestinale. I pazienti con IBD spesso presentano dolore addominale, diarrea e sanguinamento rettale, ma possono anche avere un'ampia varietà di altri sintomi come perdita di peso, febbre, nausea, vomito. Di queste malattie, le più conosciute sono:

- *malattia di Crohn;*
- *rettocolite ulcerosa.*

Da uno studio condotto nei paesi del Nord America si stima che la prevalenza di pazienti con la malattia di Crohn sia di 241 casi ogni 100.000 negli adulti, mentre nei bambini di 58 ogni 100.000 [11]. Per i pazienti con la rettocolite ulcerosa è stato stimato ci siano 263 casi ogni 100.000 adulti, invece nei bambini si contano 34 casi ogni 100.000 e probabilmente questo dato aumenterà nel tempo [12].

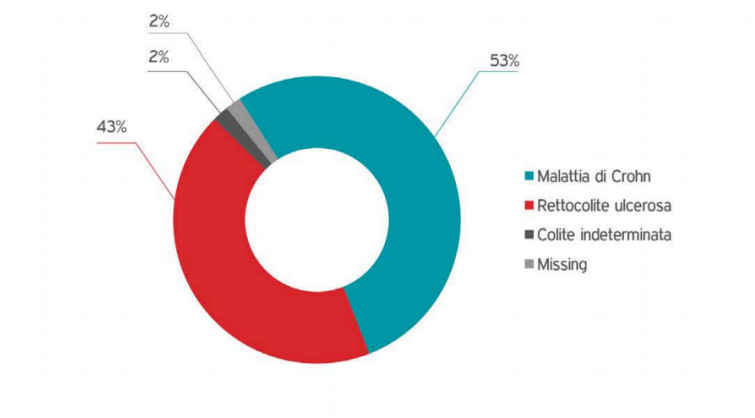


Figura 1.2. Percentuali delle malattie IBD[2]

1.2.1 Malattia di Crohn

La malattia di Crohn è una malattia infiammatoria cornica intestinale che colpisce delle parti del tratto gastrointestinale. Le zone interessate vanno dalla bocca all'ano, anche se può colpire con distribuzione segmentaria. Infatti la maggior parte delle volte colpisce l'ultima parte dell'intestino (l'ileo) e il colon [12]. Si presenta sotto forma di ulcere intestinali che se non vengono curate possono portare ad ulteriori complicazioni. La malattia di Crohn è definita cronica, in quanto presenta sintomi che non si risolvono nel tempo con una totale guarigione, ma ha un andamento altalenante tra periodi di benessere (remissione) e periodi in cui i sintomi causano maggiori disagi.

Cause e sintomi

Le cause che scatenano questa malattia non sono ancora conosciute. Per ora si suppone che una combinazione di fattori possano innescare l'infiammazione [13]. Tra i fattori troviamo: fumo di sigaretta, predisposizione genetica, inquinamento ambientale, età e alimentazione.

I sintomi dipendono dalla zona interessata dall'infiammazione. Di seguito vengono elencati i più comuni [14]:

- *diarrea cronica*
- *febbre e stanchezza*
- *dolori addominali*
- *sangue nelle feci*
- *ulcere nella bocca*
- *riduzione appetito e perdita di peso*
- *malattia perianale*

1.2.2 Rettocolite ulcerosa

La rettocolite ulcerosa è una malattia infiammatoria cornica intestinale, che colpisce in primo luogo la mucosa del retto e eventualmente può estendersi a parte o tutto il colon. È una patologia autoimmune ed è caratterizzata da un infiammazione che causa lesioni ulcerose. Come per la malattia di Crohn, anche la rettocolite ulcerosa ha un andamento caratterizzato dall'alternarsi di periodi di remissione clinica ad altri di attività acuta [15].

Cause e sintomi

La causa esatta è ancora sconosciuta. Si sospetta che i fattori scatenanti possano essere la predisposizione genetica innescata da fattori ambientali presenti nel tubo digerente [16]. Nonostante non si sappiano le cause, ci sono diverse fattori che sono comuni a più pazienti:

- *Età*: la colite ulcerosa tendenzialmente inizia prima dei 30 anni, ma può manifestarsi in qualsiasi età.
- *Razza o etnia*: i bianchi, ma soprattutto se di origine ebrea ashkenazita, hanno maggiori probabilità di contrarre la malattia.
- *Storia famigliare*: le probabilità sono più alte nel caso in cui qualche parente di primo grado abbia la malattia.

I sintomi dipendono dalla fase in cui si trova la malattia e dalla zona dell'infiammazione. Il paziente può avere i seguenti segni e sintomi [15]:

- *diarrea ricorrente*: potrebbe essere accompagnata da sanguinamento o muco.
- *febbre e stanchezza*
- *dolori addominali*: spesso accompagnati da un miglioramento dopo l'evacuazione
- *sanguinamento rettale*
- *urgenza nel defecare*: nonostante l'urgenza, sono spesso caratterizzate da piccolo volume
- *incapacità di defecare nonostante l'urgenza*
- *perdita di peso*
- *artralgie*
- *rallentamento di crescita nei bambini*

1.2.3 Diagnosi e terapia

Per la formulazione di una diagnosi, il medico pone una serie di domande al fine di escludere altre possibili cause dei sintomi del paziente. Le domande consentono di investigare sui sintomi e dalla loro entità si può constatare se la malattia è in remissione o in fase acuta. Gli esami che il medico potrebbe prescrivere sono vari (es. esami del sangue, colonscopia, ecografia dell'addome) e permettono di confermare con maggiore precisione la diagnosi.

La terapia medica per le IBD varia in base alla fase in cui si trova la malattia che potrebbe essere in remissione o in fase acuta. Tra i possibili farmaci ci sono: immunosoppressori, antibiotici, corticosteroidi, antinfiammatori [17]. Nei casi in cui la terapia farmacologica non dovesse alleviare i sintomi, il medico potrebbe richiedere l'intervento chirurgico. In questi casi si opera cercando di eliminare le cause dei sintomi, come ad esempio chiudere fistole o drenare ascessi. Comunque l'intervento è un trattamento che non cura definitivamente la malattia, ma risolve temporaneamente le complicanze.

1.3 La telemedicina applicata alle IBD

Possiamo dedurre da quanto detto nei paragrafi precedenti quanto l'incontro tra telemedicina e le IBD possa essere utile e conveniente. Essendo le IBD delle malattie croniche, è possibile attraverso l'uso di un'applicazione, monitorare in ogni paziente

lo stato della malattia e consentire l'interazione medico-paziente nel caso ci fosse bisogno di cambiare la terapia. L'assistenza a distanza rimane anche nel caso delle IBD solo uno strumento aggiuntivo in mano a pazienti e medici, ma non può sostituire in maniera completa il lavoro svolto all'interno delle strutture sanitarie. Nei prossimi capitoli parleremo di **IBD Tool**, un'applicazione nata per soddisfare questo tipo di esigenze.

Capitolo 2

Stato dell'Arte: IBD Tool

2.1 Introduzione

IBD Tool è un applicazione nata dalla collaborazione tra l'Ospedale Mauriziano, la Fondazione LINKS e il Politecnico di Torino con l'intento di supportare la cura alle IBD. Nello specifico IBD Tool è una web-application, innovativa e utilizzabile facilmente tramite tecnologie come smartphone o computer.

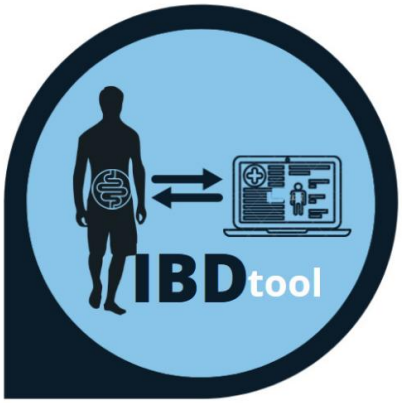


Figura 2.1. Logo IBD Tool

Per poter usufruire di questo servizio i pazienti dovranno essere invitati dal medico che li segue, che preventivamente chiederà loro se vogliono utilizzare questo strumento aggiuntivo. I pazienti che accetteranno, verranno iscritti all'applicazione e da quel momento avranno a disposizione un sistema di questionari periodici che permetterà di monitorare la loro malattia a distanza. Oltre questo avranno a

disposizione un canale diretto con i medici attraverso una chat, che risulta molto più comoda e diretta rispetto alla comunicazione via e-mail. I medici invece saranno supportati nel monitoraggio dei pazienti da avvisi periodici, che in maniera semplice e compatta, forniscono informazioni sull'attività di malattia dei pazienti.

Vedremo nei prossimi paragrafi le funzionalità disponibili su **IBD Tool** prima delle modifiche apportate da questo lavoro di tesi.

2.2 Accesso e registrazione

La prima pagina che qualsiasi utente della piattaforma visita è rappresentata in Fig. 2.2. Da qui è possibile effettuare la procedura di accesso per gli utenti già registrati, mentre coloro che non hanno ancora delle credenziali dovranno prima registrarsi. Per i medici è possibile registrarsi direttamente da questa pagina, mentre la registrazione dei pazienti è consentita solo dalla pagina personale di un medico.

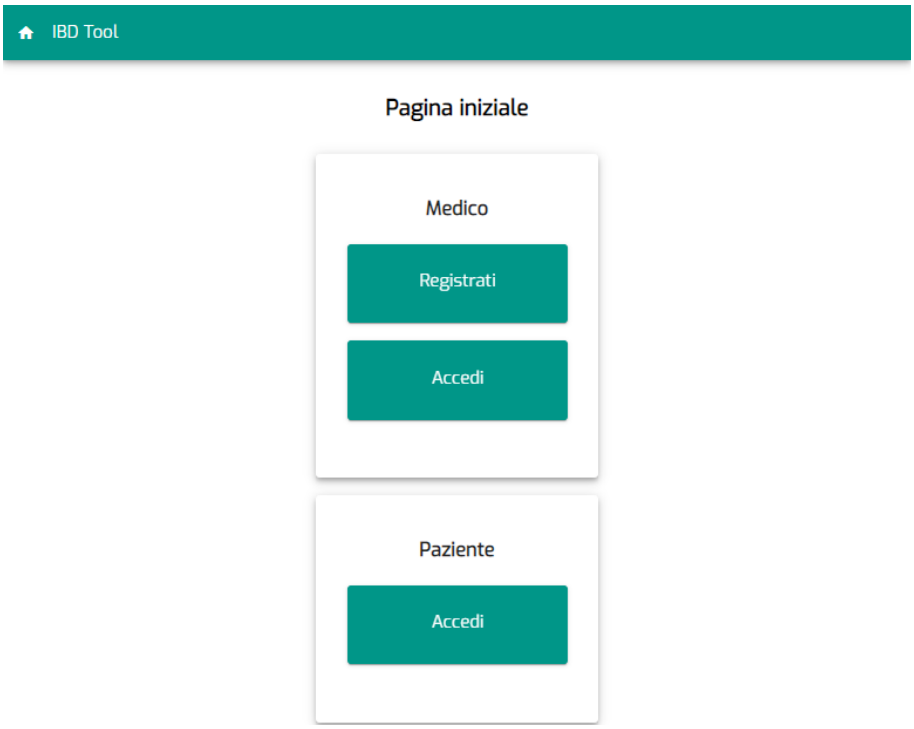


Figura 2.2. Pagina principale

Per la registrazione di un utente vengono richiesti l'indirizzo e-mail, nome, cognome, codice fiscale e la data di nascita. Nel caso della registrazione di un medico inoltre viene chiesto un numero di telefono e gli viene data la possibilità di scegliere una password. Invece per un la registrazione di un paziente viene richiesto,

oltre ai campi già elencati, la patologia. Per i pazienti la password verrà generata automaticamente dal sistema, ma viene consigliato di cambiarla con una più personale.

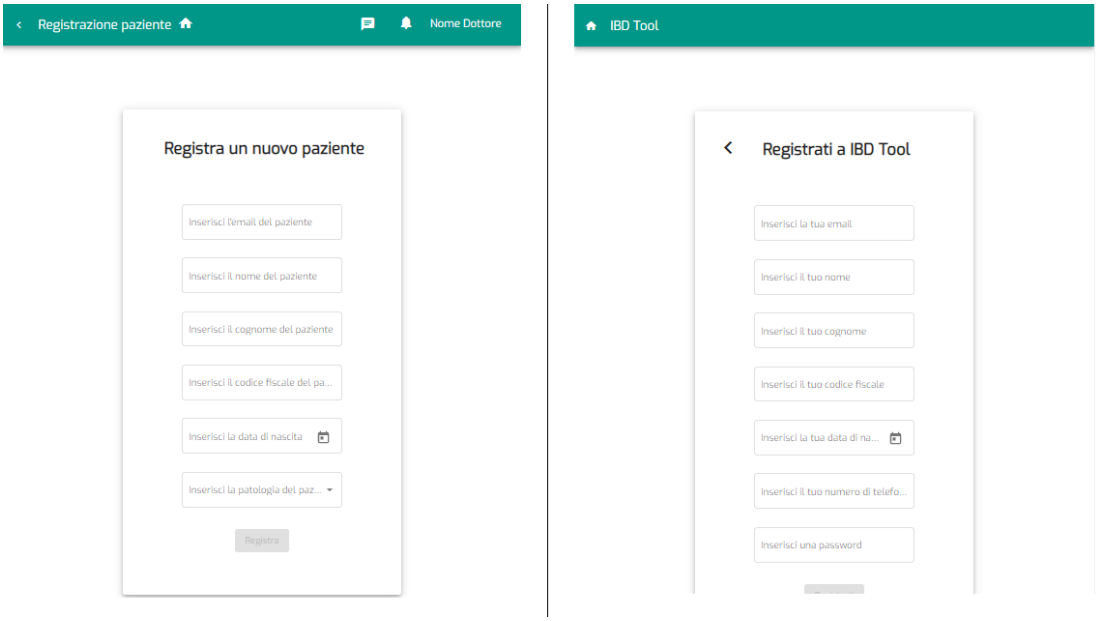


Figura 2.3. Pagine per la registrazione di utenti: a sinistra per i pazienti, a destra per i medici

2.3 Home page

Per i soli utenti registrati, dopo aver eseguito l'operazione di accesso, è visibile una home page con funzionalità differenti in base al tipo di utente che ha effettuato l'autenticazione.

Per i medici sono disponibili due pulsanti che servono per la gestione dei pazienti. I pazienti sono differenziati in due categorie: *personali* e *globali*. I primi sono quelli che il medico ha registrato dalla propria pagina personale, mentre i secondi comprendono sia i personali, sia quelli registrati da altri medici. Il medico a differenza di quanto possa fare con i pazienti globali, per i pazienti personali ha a disposizione un bottone per la loro rimozione dalla piattaforma ed, inoltre, riceve gli avvisi sull'andamento della malattia tramite *alert* e *e-mail*. Il paziente invece nella sua home page ha a disposizione altre due icone che rimandano ai questionari. La prima porta all'elenco dei questionari da completare, mentre la seconda mostra quali questionari sono stati compilati in passato. L'utente a questo punto ha piena libertà di compilare un nuovo questionario oppure controllare l'andamento della sua malattia attraverso i questionari compilati in precedenza.

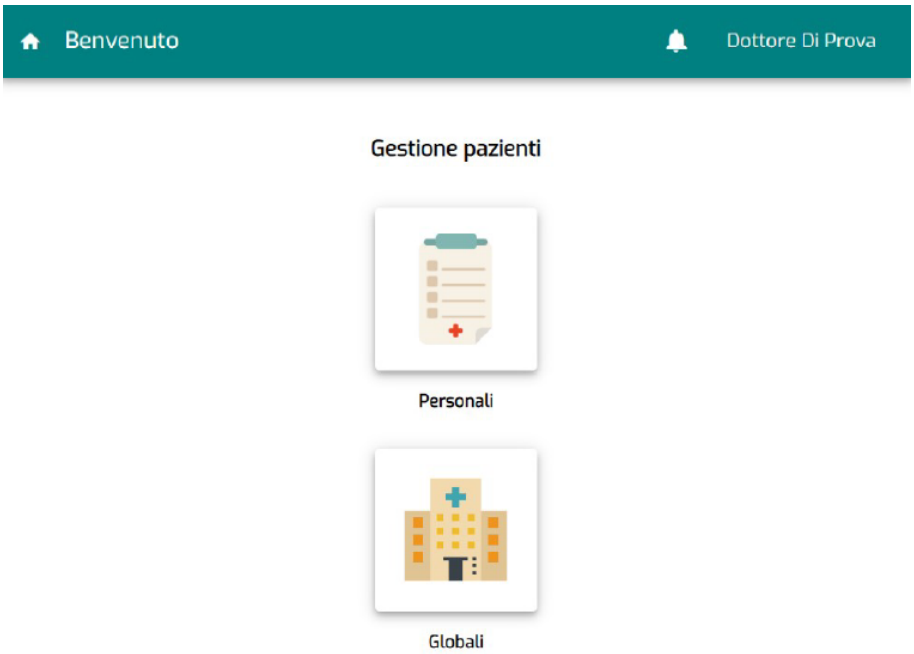


Figura 2.4. Home page per i medici

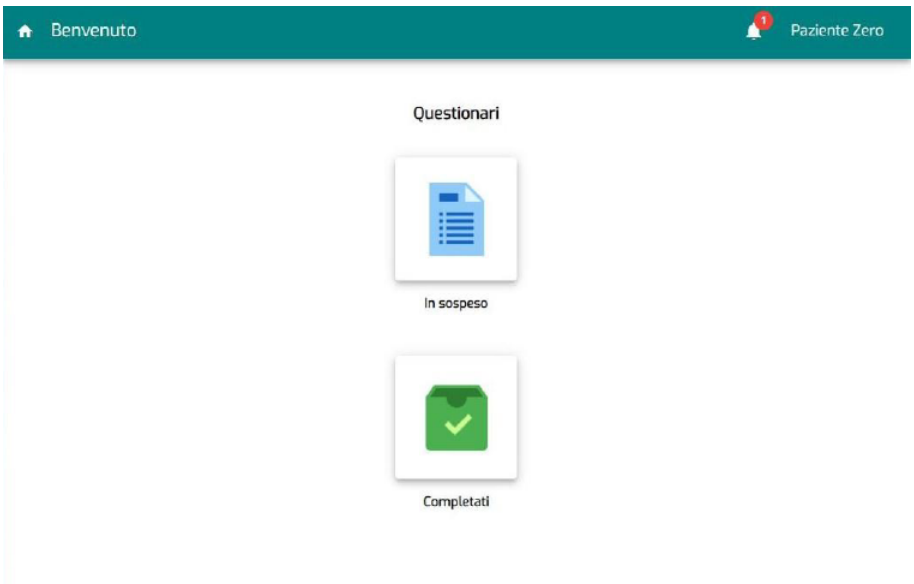


Figura 2.5. Home page per i pazienti

2.4 Questionari

Per tutti gli utenti la gestione dei questionari risulta molto facile e intuitiva. I pazienti direttamente dalla loro home page potranno visitare la pagina dei questionari da compilare. Dentro questa pagina il paziente visualizza in formato di elenco il nome e la data d'invio del questionario da completare. Da questa finestra l'utente potrà cliccare su uno dei questionari per procedere con la compilazione.

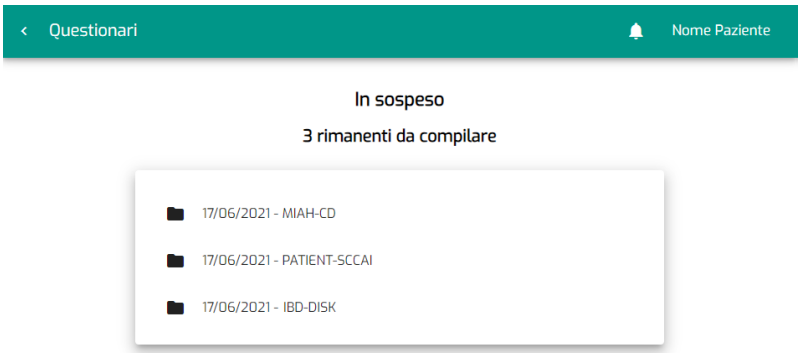


Figura 2.6. Pagina dei questionari da compilare

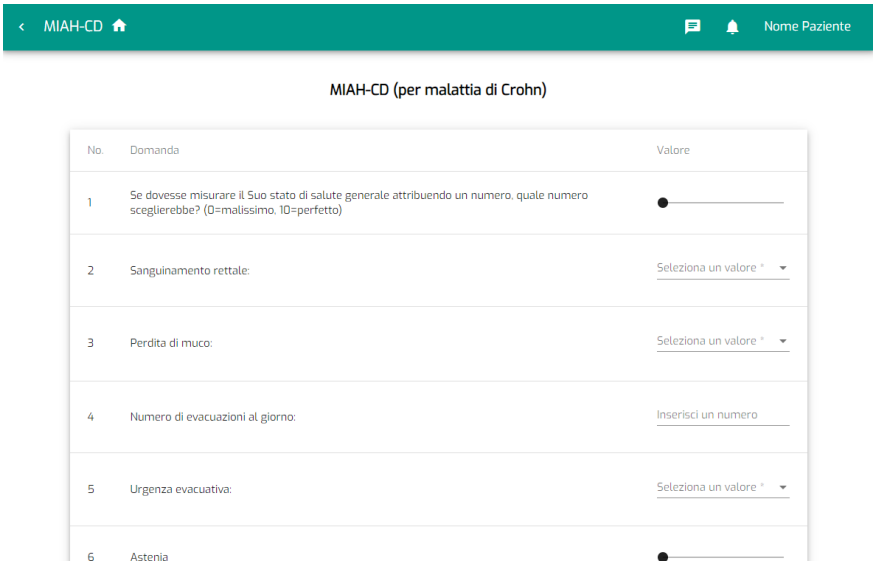


Figura 2.7. Esempio di questionario

Dopo aver compilato i questionari, i medici vengono informati delle informazioni ricavate dal questionario tramite e-mail o attraverso degli alert. Inoltre, i medici dalla propria pagina personale possono visualizzare i singoli questionari oppure visualizzare un grafico che riassume l'andamento della malattia nel tempo (Fig. 2.8).

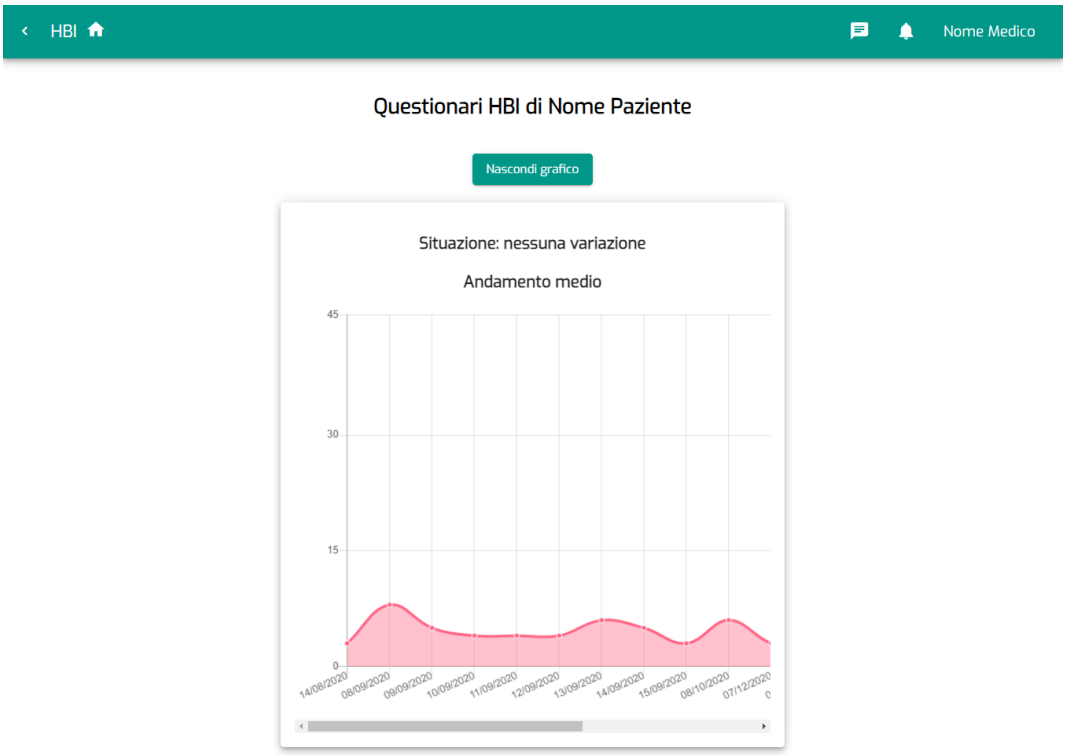


Figura 2.8. Esempio di grafico con evoluzione temporale della malattia

2.5 Gestione pazienti

I medici possono visualizzare tutti i pazienti personali presenti nell'applicazione direttamente dalla finestra rappresentata in Fig. 2.10. Per ciascuno di essi sono disponibili quattro bottoni, che permettono di:

- visualizzare la scheda del paziente (Fig. 2.9);
- eseguire una serie di operazioni sulla gestione dei questionari del paziente (descritte nel paragrafo 2.6);

- eliminare l'account del paziente dall'applicazione.

Tutte queste operazioni, tranne l'eliminazione dell'account, possono essere eseguite anche nella sezione dei pazienti globali.

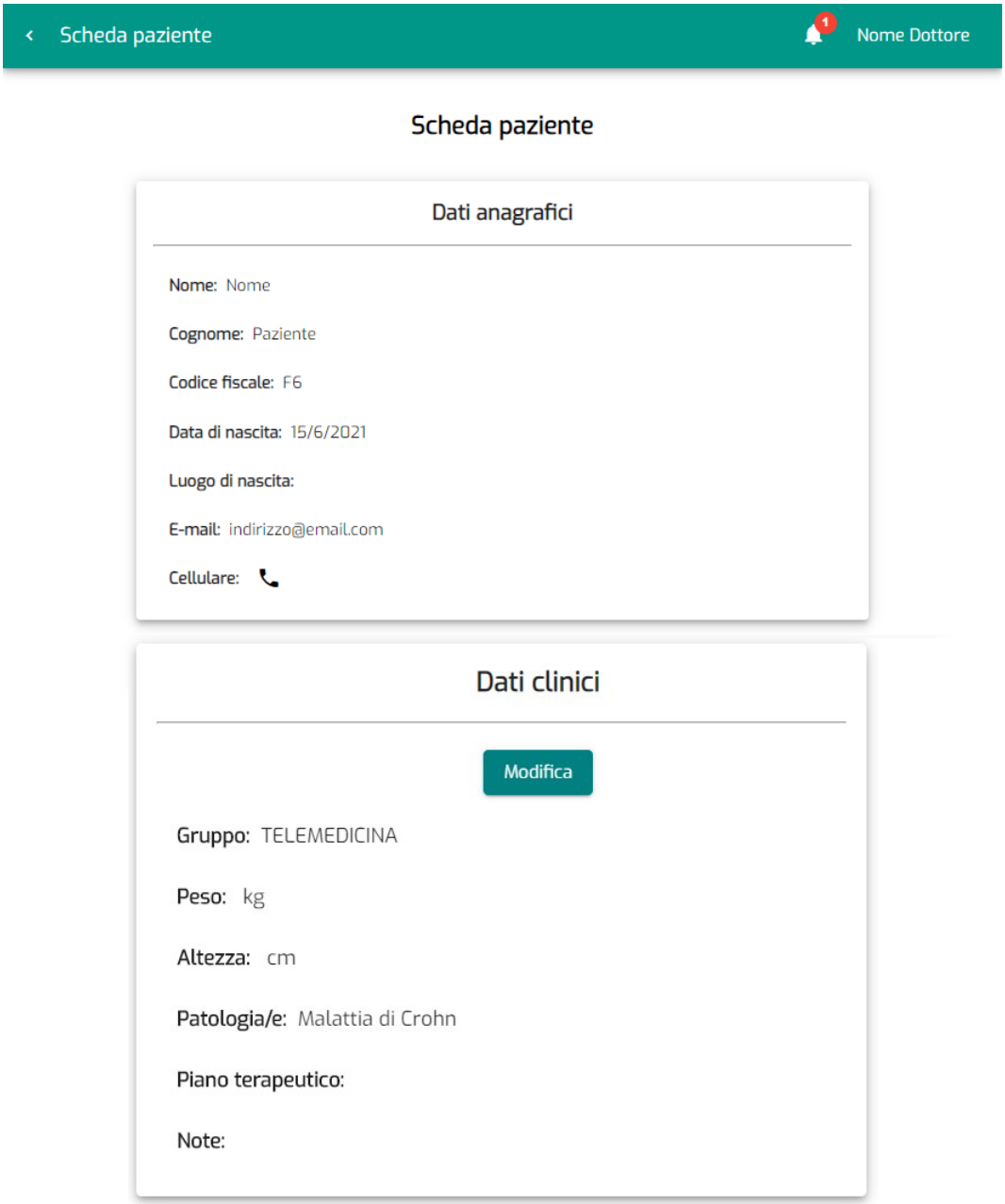


Figura 2.9. Scheda paziente

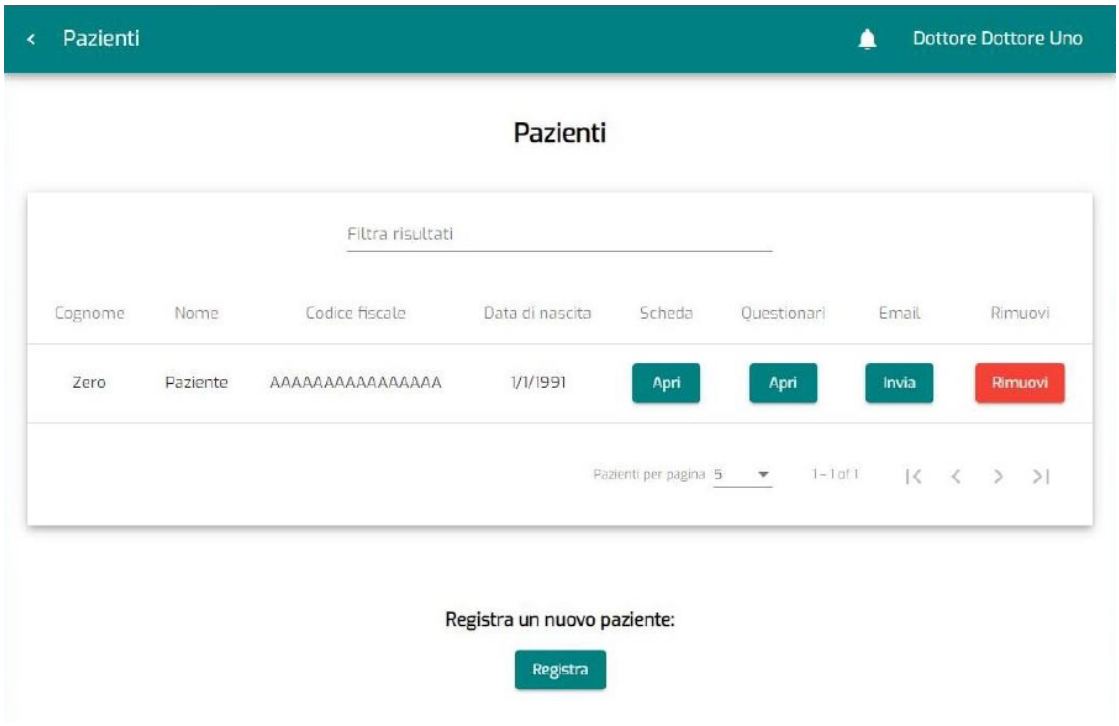


Figura 2.10. Finestra con l'elenco dei pazienti personali.

2.6 Gestione questionari

I medici hanno a disposizione una pagina che per ogni paziente permette di eseguire una serie di operazioni. Tra queste è possibile:

- consultare i questionari compilati: attraverso un bottone presente vicino al nome di ogni questionario è possibile visitare la pagina contenente i questionari compilati. Inoltre su questo bottone viene visualizzato un numero che indica quanti sono i questionari non letti;
- cambiare il tempo che intercorre tra l'invio automatico di due questionari;
- inviare manualmente un questionario;
- eliminare un questionario per il paziente selezionato;
- aggiungere un nuovo tipo di questionario per il paziente selezionato.

Inoltre nel caso il paziente venga visitato in ambulatorio, il medico può compilare dei questionari direttamente da questa pagina.

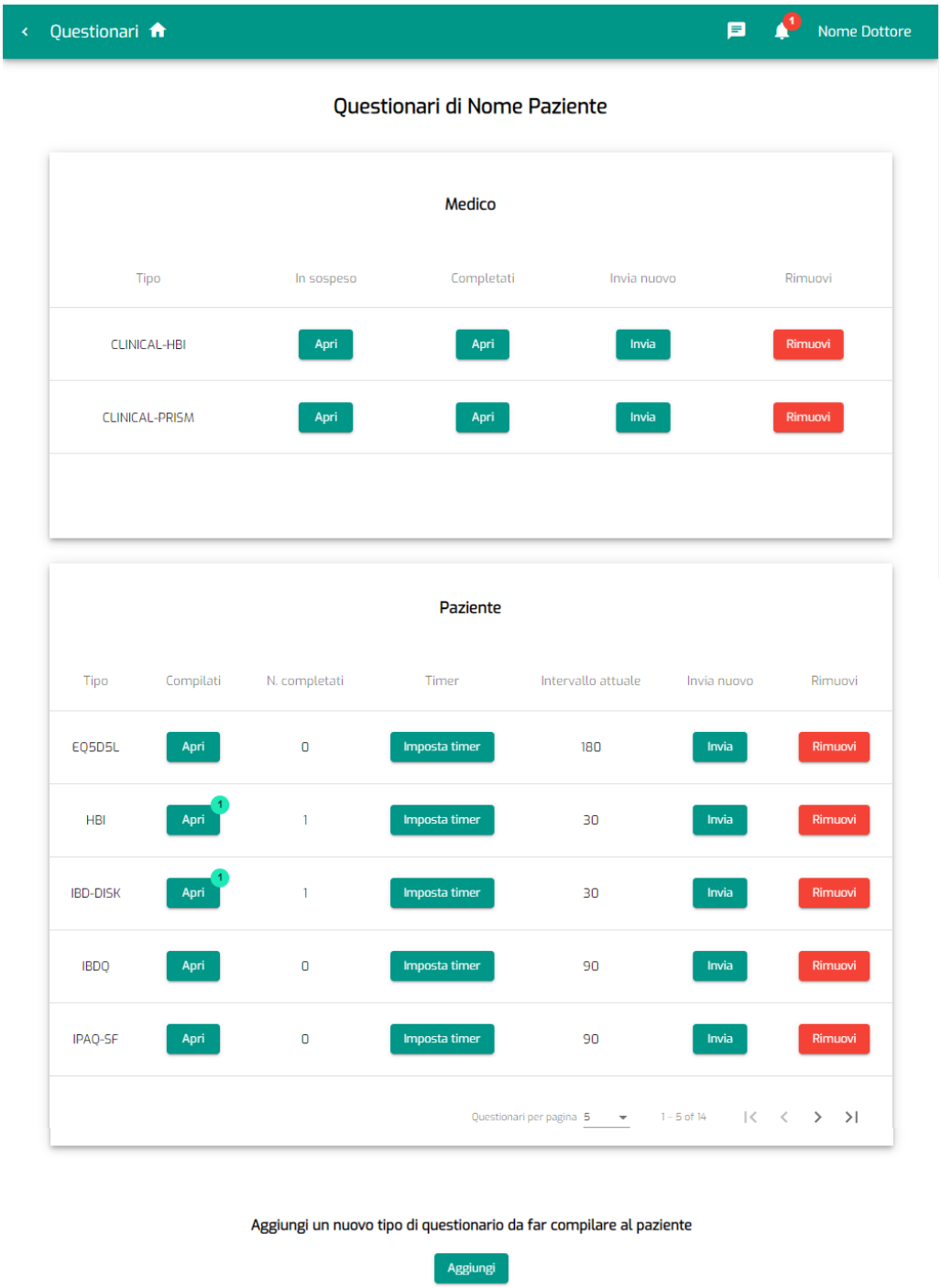


Figura 2.11. Finestra per l'interazione con i questionari compilati dal paziente selezionato

2.7 Notifiche

Le notifiche (o *alert*) servono per informare l'utente di specifici eventi. Per il medico vengono inviate delle notifiche nel caso vengano modificate le informazioni anagrafiche da un paziente oppure nel caso un paziente segnali di aver compilato un questionario in maniera errata. Invece le notifiche indirizzate ai pazienti servono prevalentemente per avvisarlo su tutte le attività riguardo i propri questionari, come ad esempio l'aggiunta o la rimozione di un tipo di questionario.

Graficamente è possibile vedere le notifiche dall'icona di una campanella presente nella toolbar. Questa verrà evidenziata da un badge rosso contenente il numero di notifiche non lette. Interagendo con questa campanella si viene reindirizzati nel centro notifiche (Fig. 2.12), in cui sono presenti tutte le notifiche. Per distinguere le notifiche non lette, vengono rappresentate in grassetto fino al momento della visualizzazione.

Figura 2.12. Finestra contenente la lista delle notifiche

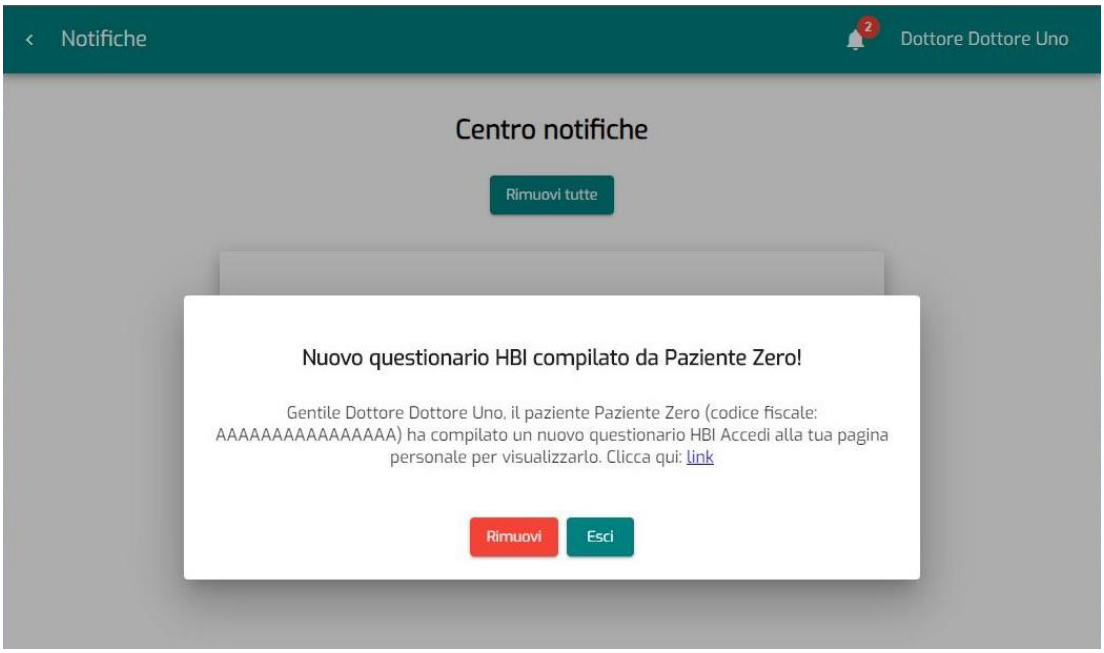


Figura 2.13. Visualizzazione di una notifica

Capitolo 3

Materiali

Il lavoro svolto in questa tesi ha come obiettivo finale lo sviluppo di nuove funzionalità per **IBD Tool**. Per raggiungere questo scopo si è diviso il lavoro in passi preliminari:

1. *studio dello stato dell'arte*: partendo dalla documentazione e dal codice dell'applicazione già esistente, sono state studiate le **tecnologie** utilizzate ed i **questionari** su cui è basato il servizio di monitoraggio;

2. *analisi e risoluzione di problemi*: inizialmente il progetto aveva alcune funzionalità che non si comportavano come era stato pianificato. Si è dunque analizzato il codice, approfondito quali possibili cause potevano scatenare dei problemi ed infine adattato il codice ad una nuova soluzione, ma cercando comunque di non stravolgere la logica del programma;

3. *analisi e sviluppo di nuove funzionalità*: in collaborazione con i medici del reparto di gastroenterologia dell'Azienda Ospedaliera Ordine Mauriziano di Torino, si è investigato su quali nuove funzionalità integrare all'interno della piattaforma e su come renderle intuitive e di facile utilizzo per tutti gli utenti.

Di seguito vedremo quali tecnologie informatiche e quali questionari sono stati scelti per l'applicazione.

3.1 Tecnologie

Per la creazione di IBD Tool sono state usate le tecnologie più recenti e comunemente usate da tutta la comunità di sviluppatori. L'obiettivo era quello di costruire un prodotto che sia valido negli anni e che all'occorrenza può facilmente essere ampliato per aggiungere nuove funzionalità. IBD Tool strutturalmente può essere divisa in due parti:

- *Front-end*

- *Back-end*

3.1.1 Front-end

Nelle applicazioni web, per front-end (in sigla FE) si intende l'interfaccia grafica del programma, ovvero la parte visibile all'utente che nel caso di questa applicazione viene eseguita da un qualsiasi browser (es. Google Chrome, Mozilla Firefox, Microsoft Edge). In **IBD Tool** si è scelto di costruire questa parte utilizzando Angular come piattaforma di sviluppo.

Angular

Angular è un framework open source per la creazione di applicazioni client a pagina singola basato sull'utilizzo di HTML e TypeScript. Le applicazioni a pagina singola (SPA) sono più rapide rispetto alle applicazioni multi-pagina (MPA), perché richiedono un singolo caricamento al momento in cui si entra nel sito, e a differenza delle MPA, durante la navigazione tra le varie aree o quando si eseguono delle operazioni, i dati vengono recuperati in background sostituendo solo le parti necessarie [18].

Inoltre, Angular è uno strumento facile e veloce per sviluppare applicazioni multi-piattaforma. Attraverso dei toolkit è facilmente possibile rendere le applicazioni *responsive*, ovvero far sì che il design del sito web si adatti alla piattaforma utilizzata (es. laptop, smartphone, tablet). In IBD Tool è stato fatto un grosso utilizzo di Angular Material, un toolkit che mette a disposizione una serie di componenti che permettono facilmente di costruire oggetti all'interno della pagina (es. tabelle, liste, barre di ricerca). Per esempio, per costruire una toolbar con all'interno il testo "My Application" il codice da scrivere è molto semplice, infatti basta copiare i seguenti tag all'interno del file HTML.

```
<mat-toolbar>
    <span>My Application</span>
</mat-toolbar>
```

In un'applicazione Angular è necessario un modulo radice che permetta il bootstrap dell'intera applicazione. Oltre al modulo radice esistono una serie di moduli che si dividono principalmente in due tipi [19]:

- *Components*
- *Services*

I *components* definiscono le *viste*, e cioè l'insieme degli elementi che Angular può visualizzare in base alla logica e ai dati del programma. Ogni *component* viene etichettato con l'annotazione **@Component()** ed è composto tipicamente da tre file: uno HTML, uno foglio di stile (CSS) e una classe TypeScript. Il file HTML

permette di definire gli elementi all'interno della pagina, mentre attraverso il file CSS si definisce la loro collocazione e la loro proprietà. Tutte le proprietà definite nel file CSS possono essere dichiarate anche nel foglio HTML, ma questa divisione rende il foglio HTML più leggibile e comprensibile. TypeScript è un linguaggio di programmazione Object-Oriented, e viene utilizzato all'interno del componente per parte logica, ad esempio la gestione i dati, l'ordinamento oppure le funzionalità per contattare i servizi che si occupano di inviare o ricevere dati. All'interno si trovano i metodi che gestiscono le operazioni dell'utente. I tre metodi che si trovano più frequentemente sono:

- *constuructor(...)*: è responsabile della costruzione del componente. All'interno delle parentesi vengono passati come parametri tutti i *services* che successivamente il componente avrà bisogno di utilizzare.

- *ngOnInit()*: inizializza il componente. Al suo interno si trovano tutte quelle operazioni preliminari che sono fondamentali per il corretto funzionamento del componente.

- *ngOnDestroy()*: è l'ultimo metodo che viene eseguito prima della distruzione del componente. Al suo interno tipicamente vengono disabilitate alcune funzioni asincrone, in modo da evitare che si attivino durante la navigazione in altri componenti.

Nel seguente esempio possiamo vedere la creazione di un component. Il codice riportato è quello presente nella classe TypeScript e richiama tramite le parole chiave *templateURL* e *styleUrls* i file HTML e CSS, mentre la parola chiave *selector* indica con quale nome il componente dovrà essere invocato dagli altri componenti [20].

```
@Component({
  selector: 'app-example',
  templateUrl: './example.component.html',
  styleUrls: ['./example.component.css']
})
export class ExampleComponent {
    //Sezione per il codice TypeScript
}
```

I *services* invece sono utilizzati per funzionalità specifiche non direttamente correlate alla visualizzazione, come ad esempio la gestione delle richieste HTTP verso il server. Per essere *iniettati* all'interno dei componenti devono possedere l'annotazione **@Injectable**, cosicché qualsiasi componente potrà utilizzare questo *service* passandolo all'interno del costruttore. Questo modello di progettazione consente di scrivere codice più testabile e flessibile [20]. Per illustrare come viene costruito un

service, successivamente è presente un esempio in cui è definito un solo metodo che permette ad ogni invocazione di stampare un numero sulla console.

```
@Injectable({providedIn: 'root'})
export class Logger {
  writeCount(count: number) {
    console.warn(count);
  }
}
```

3.1.2 Back-end

Con il termine back-end (in sigla BE) si indica la parte dell'applicazione che viene elaborata su un server gestito direttamente o indirettamente dagli amministratori. Dato che il server non dovrebbe essere modificabile senza determinate autorizzazioni, gli vengono delegate le operazioni che necessitano di un alto livello di sicurezza. Tra le funzionalità più ricorrenti ci sono: autenticazione e gestione delle autorizzazioni degli utenti, manutenzione dei dati, verifica correttezza ed elaborazione di dati inseriti da un utente.

Java e Spring

Per realizzare la parte back-end di **IBD Tool**, è stato scelto come linguaggio di programmazione Java. Questo linguaggio è uno dei più popolari e può essere utilizzato sopra Spring Boot, uno dei più famosi e moderni framework per la creazione di applicazioni web basate su *microservizi*. L'architettura a *microservizi* è nata in contrapposizione a quella monolitica. Nella monolitica il programma può essere rappresentato come un grosso blocco di codice in cui ogni funzionalità è estremamente collegata alle altre. Con questo approccio il codice diventa più complesso e meno scalabile all'aumentare della dimensione. Con l'architettura a *microservizi* invece ogni servizio può essere sviluppato in maniera indipendente dagli altri. Il risultato è un prodotto più semplice e con una maggiore tolleranza ai guasti [21].

Spring ha come obiettivo principale quello di semplificare la costruzione delle applicazioni Java. I due modelli di programmazione che rendono Spring estremamente comodo sono:

- *Aspect Oriented Programming (AOP)*
- *Inversion of Control (IoC)*

L'Aspect-Oriented programming mira ad aumentare la modularità consentendo al programmatore di non doversi preoccupare di funzionalità ricorrenti, concentrandosi solo sulla logica del programma [22]. Questo paradigma permette di aggiungere

funzionalità al codice, senza modificarlo, ma specificando separatamente su quali parti deve agire la nuova funzionalità.

L'Inversion of Control si occupa di aiutare lo sviluppatore nella gestione delle dipendenze. Infatti, durante la scrittura di un grosso programma, la gestione delle dipendenze è un compito che richiede attenzione. Un'incorrettezza nel flusso delle dipendenze può generare problemi come quello delle dipendenze cicliche. Queste nascono quando una classe ne include altre, che a loro volta includono le prima, portando i componenti ad essere dipendenti da loro stessi e quindi il programma a non funzionare correttamente. Spring dà al programmatore la libertà di non doversi preoccupare delle dipendenze, utilizzando l'IoC. Diversamente dalla programmazione tradizionale dove lo sviluppatore specifica le operazioni di creazione, inizializzazione e invocazione degli oggetti nei metodi, nell'IoC lo sviluppatore delegherà tutti questi aspetti al framework che reagendo a qualche stimolo si occuperà di iniettare le dipendenze. Per lo sviluppatore uno dei modi più comuni per esplicitare di quali dipendenze ha bisogno è dichiarare la classe da importare come attributo ed annotandolo con **@Autowiring**. Spring implementa la IoC tramite Dependency Injection (DI). Questa prevede che gli oggetti all'interno della nostra applicazione accettino le dipendenze iniettate dall'esterno.

Abbiamo visto come l'utilizzo di Spring permetta di tralasciare delle complessità per concentrarsi sulla logica di business. Il back-end di **IBD Tool** è organizzato seguendo la *multitier architecture*, che si abbina perfettamente con gli strumenti messi a disposizione da Spring. La *multitier architecture* è composta principalmente da 4 livelli:

- *Presentation Layer*: è lo strato più esterno, incaricato di ricevere i dati dal front-end e passarli al Service Layer. Quando il Service Layer avrà completato le sue operazioni, il Presentation Layer avrà il compito di generare una risposta e spedirla al front-end. Utilizzando Spring è possibile creare una classe che svolga questi compiti annotandola con **@Controller**. In IBD Tool per la comunicazione tra back-end e front-end viene utilizzata l'architettura REST, quindi per questo motivo sulle classi incaricate del Presentation Layer si utilizza l'annotazione **@RestController**. All'interno della classe si avranno diversi metodi, ognuno responsabile di rispondere nel caso si arrivi una richiesta HTTP specifica (es. GET, PUT, POST, DELETE) all'URL a cui il metodo risponde. Ad esempio nel caso si voglia creare un metodo responsabile di rispondere ad una GET all'URL *"esempioUrl"*, sarà possibile utilizzare l'annotazione **@GetMapping("*esempioURL*")**, delegando a Spring il compito di invocare il metodo al momento giusto. Oltre a occuparsi di gestire le richieste, il Presentation Layer deve occuparsi di fornire le risposte sia nel caso in cui la richiesta vada bene, sia nel caso venga scatenata un'eccezione. Nel secondo caso è bene distinguere attraverso un messaggio personalizzato i vari motivi che l'hanno scatenata. Per questo si utilizzano i vari stati messi a disposizione

dal protocollo HTTP. Ad esempio nel caso venga ricercato un dato inesistente si può restituire un messaggio 404, che identifica una risorsa non trovata.

- *Service Layer*: è lo strato più personalizzabile che definisce la business logic dell'applicazione. Viene contattato dal Presentation Layer e utilizza le classi fornitegli dal Data Domain Layer. Le classi di questo strato sono etichettate con l'annotazione **@Service**. All'interno di questo livello è molto importante validare i dati ricevuti in modo da evitare di compromettere l'integrità del database. Infatti nonostante si possano fare dei controlli sui dati direttamente sul front-end, non si può far affidamento su di essi in quanto possono essere facilmente evitati dall'utente. Il Service Layer è molto legato a tutte le altre classi del back-end, per questo motivo è necessaria una procedura per importarle con semplicità. Spring ci mette a disposizione l'annotazione **@Autowired**, che permette di importare un'altra classe come se fosse un oggetto del *service*.

- *Data Domain Layer*: è lo strato che si occupa di astrarre i dati in oggetti, che il Service Layer utilizzerà per applicare la business logic. Su Spring il Data Domain Layer è rappresentato attraverso una serie di classi che modellano i dati presenti sul database. Nel caso si utilizzi un database a documenti (come MongoDB su IBD Tool) si etichettano le classi di questo strato con **@Document()**. Questo serve per far capire a Spring che l'oggetto rappresentato nella classe è l'astrazione di un dato del database. Tra gli attributi dell'oggetto deve esserci uno etichettato con **@Id**, in modo da indicare quale attributo è univoco all'interno della collezione. Se non si dovesse indicare con @Id nessun attributo, verrà generato un campo con un id generato automaticamente.

- *Data Access Layer*: è lo strato che offre i metodi CRUD (Create, Read, Update, Delete) per l'accesso alla base dati. Utilizzando Spring risulta molto semplice creare una classe che mette a disposizione tutti i metodi per interrogare il database. Se utilizziamo un database a documenti come *MongoDB* possiamo creare un interfaccia che estenda **MongoRepository <NomeClasse, Chiave>**. Tra parentesi angolari vengono indicati come primo campo il nome della classe del Data Domain Layer per cui reperisce le informazioni, e come seconda il tipo della chiave primaria della suddetta classe. Nel caso venga utilizzato MongoRepository, successivamente si potranno creare dei metodi per interrogare il database senza conoscere le query dello specifico database. Ad esempio, scrivendo un metodo con nome *findAllByNome(String nome)* e passando come parametro una stringa che corrisponda al nome ricercato, il database restituirà una lista di oggetti di tipo *NomeClasse*, che hanno il valore dell'attributo *nome* uguale alla stringa passata come parametro. Nel caso la ricerca fosse particolarmente complicata e non si potesse creare un metodo nel

modo precedentemente illustrato, Spring mette a disposizione l'annotazione **@Query()**, in cui è possibile inserire la query tra le parentesi.

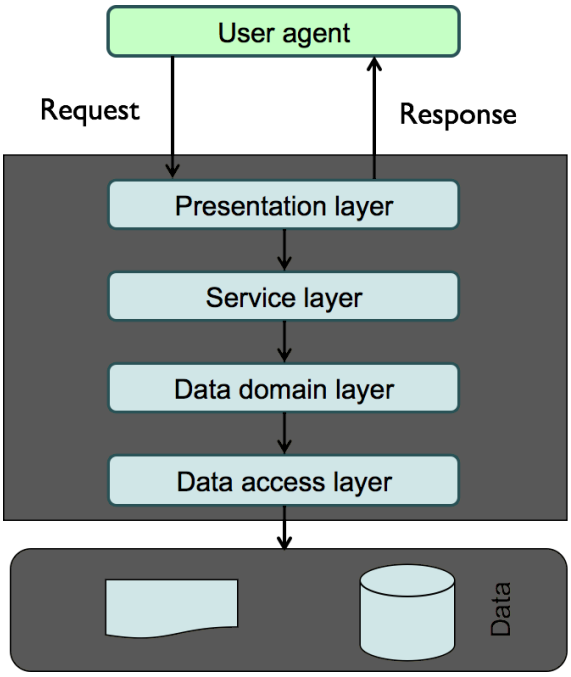


Figura 3.1. Rappresentazione della Multi-Tier Architecture

Nello sviluppo di un'applicazione un aspetto che non può essere assolutamente tralasciato è la gestione della sicurezza. Spring ha un modulo chiamato *Spring Security* che permette di gestire in maniera efficace ed estremamente semplice le funzioni di sicurezza dell'applicazione. Per avere degli alti standard di sicurezza è importante controllare tutti gli accessi. Per creare un sistema di controllo degli accessi bisogna innanzi tutto creare delle classi che rappresentino gli utenti registrati con *username*, *password* ed eventualmente il loro ruolo per un controllo più fine sugli accessi. Per il salvataggio della password è fondamentale utilizzare un sistema che la renda indecifrabile agli amministratori del database. A tal proposito, prima del salvataggio della nuova password sul database, viene fatta un operazione di *hash* che non permette di risalire alla password iniziale. Per evitare attacchi di tipo dizionario, prima di fare l'hash si concatena la password con un ulteriore stringa chiamata *sale*. Gli attacchi di tipo dizionario basandosi sul precalcolo degli hash di un elenco di password comuni, non possono essere utilizzati nel caso le password siano state modificate con un sale specifico per ogni password.

Dopo aver costruito un sistema che permetta di memorizzare gli utenti registrati, si ha la necessita di controllare se gli utenti che richiedono le risorse ne hanno effettivamente il diritto. Spring Security permette di applicare il controllo in due punti,

che spesso vengono usati in sequenza in modo da creare due livelli di sicurezza:

1. *controllo sugli end-points*: viene definito per quali URL è necessario essere autorizzato e quali invece sono visitabili da tutti (es. pagina di login o registrazione). Inoltre si può fare un controllo più fine sugli autenticati e controllare per quali URL hanno un'autorizzazione e per quali no;

2. *controllo sui metodi*: applicando l'annotazione **@PreAuthorize("hasAny Authority('ROLE')")** è possibile fare un controllo su chi accede al metodo, indicando nel campo *ROLE* il tipo di autorizzazione necessaria.

In **IBD Tool** il processo di autenticazione inizia in un form dove viene richiesto di inserire *username* e *password*. Queste informazioni saranno ricevute dal back-end che in caso di corrispondenza rilascerà un token che identifica l'utente. Questo token verrà quindi conservato nel front-end, che si occuperà di allegarlo a tutte le successive richieste HTTP. Nello specifico, il token viene chiamato **JSON-Web-Token** o **JWT** ed è una stringa composta da tre parti:

- *Header*: indica la tipologia del token e quella dell'algoritmo utilizzato per la cifratura;

- *Payload*: contiene una lista di informazioni sul token che possono essere predefinite (es. chi ha rilasciato il token, per quanto sarà valido, quando è stato generato) oppure personalizzate dallo sviluppatore dell'applicazione;

- *Signature*: è la concatenazione della codifica in *BASE64* dell'*Header*, del *Payload* e del *BASE64(HMAC-SHA256("Header" + "Payload", SecretKey)*, dove la *SecretKey* è la chiave segreta usata per l'*HMAC-SHA256*.

In **IBD Tool** il token ha una validità di 15 minuti, ma è possibile rinnovarlo attraverso delle richieste specifiche. Il front-end è configurato per avanzare queste richieste ogni 10 minuti, in questo modo l'utente non è costretto a inserire ogni 15 minuti la password.

MongoDB

MongoDB è un sistema open-source per la gestione di database non relazionali. I database non relazionali vengono definiti *NoSQL*, in quanto nati in contrapposizione ai database relazionali che utilizzano il linguaggio SQL per la costruzione delle interrogazioni. I database relazionali hanno una struttura rigida formata da tabelle. Ognuna di queste rappresenta una struttura dati dove ogni riga rappresenta un singolo oggetto e le colonne i suoi vari attributi. È molto frequente che ci sia la necessità di abbinare i dati da diverse tabelle che hanno una colonna in comune, con l'operazione nota come *join*.

Dalla necessità di eliminare la rigidità della struttura tabellare e di evitare la pesantezza e la lentezza derivanti dal frequente utilizzo di join, si è cercato un sistema che abbattesse queste criticità. Il sistema è stato trovato nei database NoSQL. Di database di questo tipo ne esistono di varia natura, ma noi analizzeremo solo quelli a *documenti* in quanto è il tipo utilizzato da **MongoDB** e quindi su **IBD Tool**. Il termine *documento* indica la struttura che ospita i dati relativi a un singolo oggetto. I dati sono memorizzati in formato *BSON*, un'estensione del formato JSON che permette di conservare i dati in formato binario in modo da incrementare la velocità di ricerca. Logicamente i dati sono organizzati secondo il pattern *Chiave-Valore*. All'interno di un documento possono essere raccolti non solo i dati propri di quell'entità come avveniva con le tabelle, ma anche tutti quelli associati. In questo modo attraverso una singola ricerca è possibile ottenere tutte le informazioni correlate.


```
{
  "_id": "03/01/2021-IBD-DISK-7dcdh",
  "type": "IBD-DISK",
  "doctorID": "XXXXX",
  "patientSSN": "XXXX",
  "compiled": true,
  "date": {
    "$date": "2021-01-03T11:31:31.160Z"
  },
  "results": [3,2,2,5,0,0,0,0,0],
  "finalScore": 12,
  "read": true,
  "evaluation": false,
  "warning": false
}
```


Figura 3.2. Esempio di un documento

Un ulteriore problema dei database relazionali è che per rispettare le proprietà **ACID** (atomicità, consistenza, isolamento, durabilità) non è possibile distribuire il database su più nodi. Questo rende il sistema poco scalabile e quindi problematico quando numerosi utenti cercano di accedervi contemporaneamente. Per questo motivo in contrapposizione alle proprietà ACID, per i database NoSQL si utilizza un approccio chiamato **BASE**. Prima di parlare dell'approccio BASE è essenziale introdurre il **teorema CAP**. Questo teorema afferma che in un sistema informatico distribuito è impossibile che le seguenti tre proprietà siano contemporaneamente garantite:

- *Consistency*: un sistema rispetta questa proprietà quando è in grado di garantire che una volta memorizzato un dato, questo sarà sempre lo stesso ad ogni richiesta fino ad una sua eventuale ulteriore modifica;
- *Availability*: proprietà che indica un sistema sempre disponibile a soddisfare le richieste;
- *Partition Tolerance*: questa proprietà è soddisfatta quando il sistema distribuito continua a funzionare nonostante parte di esso non risponda più a causa di malfunzionamenti della rete.

Non volendo rinunciare alla *Partition Tolerance* che permette di rimanere performanti anche con un grande numero di utenti, nei database non relazionali è stato introdotto l'approccio BASE che rappresenta una soluzione al limite del teorema CAP. L'acronimo indica quali principi sono alla base di questo approccio e sono:

- ***B**asically **A**vailable*: il sistema deve essere sempre in grado di rendere disponibili i dati;
- ***S**oft State*: il sistema può evolvere il suo stato nel tempo, anche in momenti in cui non sono state effettuate delle operazioni di scrittura o lettura;
- ***E**ventual consistency*: la consistenza non è garantita in ogni momento, ma il sistema nel tempo cerca di convergere ad uno stato consistente.

Si può notare come la proprietà che viene rilassata maggiormente è quella della consistenza. Alcuni siti di e-commerce che possono risentire di una grossa mole di richieste al database, i problemi causati dall'inconsistenza vengono risolti a posteriori, ad esempio tramite il rimborso di un bene che effettivamente non poteva essere venduto in quanto non più disponibile.

IBD Tool utilizza un database MongoDB collocato in cloud, accessibile utilizzando un browser e collegandosi alla piattaforma *MongoDB Atlas* oppure installando sulla macchina di sviluppo il software *MongoDBCompass*.

3.1.3 Architettura di comunicazione tra BE e FE

Come abbiamo appena visto, **IBD Tool** è divisa in più elementi che comunicano attraverso la rete. Per comprendere meglio come le varie parti collaborano, descriveremo l'architettura REST e il protocollo applicativo HTTP su cui REST si basa.

HTTP

La sigla HTTP sta per Hypertext Transfer Protocol ed è un protocollo che secondo il modello OSI può essere collocato al livello applicativo. Il suo ruolo è quello di

permettere lo scambio di informazioni relative all'applicazione in un'architettura *client-server*. Generalmente si appoggia sul protocollo di trasporto TCP. Un concetto fondamentale nell'HTTP è l'URL che identifica univocamente una risorsa sul web. Le URL nel caso del protocollo HTTP hanno tutte la seguente struttura:

http://hostname[:port]/path[?query][#fragment]

I campi hanno il seguente significato:

- *Hostname*: è il nome del dominio su cui risiede la risorsa ricercata. Potrebbe essere sostituito dall'indirizzo IP, ma essendo difficile che un utente si ricordi gli indirizzi IP, è molto più frequente inserire il nome del dominio e lasciare al servizio DNS il compito di tradurre i nomi in indirizzi IP;

- *Port*: identifica la porta sul quale è ospitato il servizio. Su un server possono essere attive più porte, a cui rispondono servizi diversi. Se non viene specificato verrà contattata la porta standard del protocollo indicato all'inizio dell'URL. Per lo standard, l'HTTP ha la porta numero 80, mentre per l'HTTPS la 443;

- *Path*: è un campo opzionale e indica il percorso della risorsa all'interno del server;

- *Query*: è un campo opzionale che permette di passare al server più di un parametro con il formato *nomeParametro=valoreParametro*. Nel caso si volessero inviare più di un parametro, questi vengono concatenati utilizzando il simbolo &;

- *Fragment*: è un campo opzionale che identifica una specifica parte del foglio HTML. Viene utilizzato spesso per indicare uno specifico paragrafo all'interno di una grossa pagina HTML.

Il protocollo HTTP è molto semplice ed è stato pensato per essere privo di stati. Ad una richiesta da parte del client, il server risponderà alla singola richiesta, non prendendo in considerazione quelle precedenti.

Le richieste sono composte da una serie di informazioni: l'*azione* da svolgere, la *URL* della risorsa, una serie di sotto-intestazioni e in alcuni casi un corpo della richiesta. Esistono una serie di possibili azioni, vediamo ora le più importanti:

- *GET*: richiede l'invio di una risorsa. Permette di effettuare delle richieste *condizionali* o *partizionali*. Le prime permettono di sfruttare le cache, infatti si chiede se la risorsa sul sito è cambiata rispetto a quella presente in locale e di inviarla solo se è stata aggiornata. Le seconde invece si chiamano partizionali perché richiedono solo una parte specifica della risorsa. Questo tipo di richiesta non ha mai un corpo;

- *POST*: invia nel corpo della richiesta una serie di informazioni alla risorsa indicata nell'URL. Questo metodo si utilizza per avere risultati non idempotenti, quindi se si invia la medesima richiesta POST per più volte alla stessa risorsa, la risorsa avrà stati sempre diversi ad ogni richiesta;

- *PUT*: questa azione è simile alla precedente, ma si usa per avere risultati idempotenti. Ciò vuol dire che il risultato di più richieste identiche è uguale a quello di una sola;

- *DELETE*: utilizzata per richiedere la distruzione della risorsa indicata dall'URL;

- *OPTIONS*: è usata per richiedere al server l'elenco delle azioni disponibili su una specifica risorsa.

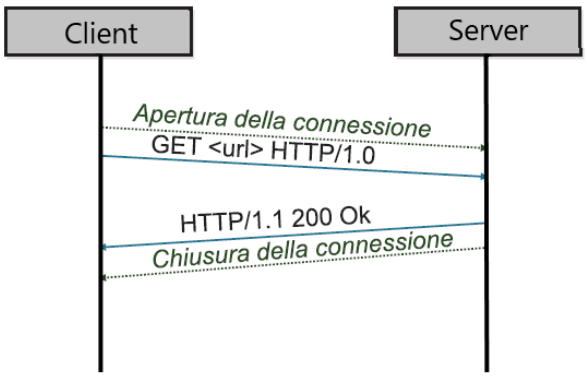


Figura 3.3. Esempio di interazione tra client e server

Le risposte da parte del server sono composte da: versione del protocollo, codice di stato, ulteriori sotto-intestazioni e opzionalmente il corpo della risposta. I codici di stato descrivono l'esito dell'operazione richiesta. Sono composti da tre cifre di cui la prima identifica la classe, mentre le altre due descrivono con maggiore dettaglio l'esito dell'operazione. Le classi sono cinque, vediamo di seguito come sono divise:

- **1xx** - *Informational*: sono dei messaggi che vengono inviati a titolo informativo nel mentre che la richiesta è in svolgimento. Un esempio è il codice 100 CONTINUE, che viene usato dal server per dire al client che ha accettato la parte di richiesta mandata e che può continuare con l'invio delle altre parti;

- **2xx** - *Successful*: la richiesta è stata accettata e compresa dal server.

- **3xx** - *Redirection*: sono dei messaggi che permettono di reindirizzare il client verso la risorsa cercata.
- **4xx** - *Client error*: la richiesta è stata rifiutata a causa di un errore nella richiesta del client. Se vuole che la richiesta vada a buon fine si deve riformulare. Un esempio è l'errore 404 NOT FOUND che viene rilanciato nel caso si richieda una risorsa inesistente.
- **5xx** - *Server error*: messaggi di questo tipo vengono inviati quando la richiesta è fallita a causa di un errore interno al server. Un esempio potrebbe essere quando il database non risponde per un malfunzionamento della rete. Conviene rispondere sempre con lo stato 500 perché dare troppi dettagli su un malfunzionamento permette a dei possibili attaccanti di trovare punti deboli nell'applicazione.

REST

REST sta per *Representational state transfer* ed è un paradigma di programmazione che definisce una serie di linee guida per il trasferimento dei dati in un'applicazione web. Utilizzando REST si vogliono garantire le seguenti proprietà:

- *Eterogeneità*: capacità di comunicazione tra dispositivi diversi;
- *Scalabilità*: capacità di avere prestazioni costanti nonostante l'aumento del carico sul server;
- *Possibilità di evolvere*: l'architettura non deve avere dei vincoli che ne impediscano la crescita;
- *Affidabilità*: capacità del sistema di mantenere funzionanti i propri servizi anche in caso di un malfunzionamento;
- *Efficienza*: capacità di offrire il servizio con prestazioni alte mantenendo il numero di risorse necessarie il più basso possibile;
- *Prestazioni*: capacità di erogare un servizio in meno tempo possibile.

Le applicazioni REST sono costruite su un'architettura client-server, in cui il server ospita uno o più servizi. Ogni servizio gestisce un insieme di informazioni, ognuna di esse ha un nome univoco (URI) che permette di agire sulla risorsa attraverso le operazioni **CRUD**. A questo scopo vengono utilizzati i seguenti metodi dell'HTTP:

- *GET*: ottiene un singola voce o una collezione;
- *POST*: crea una voce singola o una collezione;
- *PUT*: aggiorna una voce singola o sostituisce una collezione;

- *DELETE*: cancella la risorsa a cui si riferisce l'URL.

Per valutare quanto un'applicazione aderisce ai principi REST si può utilizzare il **Richardson Maturity Model** (in sigla **RMM**). Questo modello classifica le applicazioni in quattro livelli, dove ad un livello più alto corrisponde un maggiore livello di maturità dell'applicazione REST:

- *Livello 0*: viene utilizzato l'HTTP come protocollo applicativo. Tutte le richieste vengono rivolte ad una singola URL utilizzando prevalentemente il metodo POST, mettendo come parametro il metodo da utilizzare. Un'applicazione di livello zero non si può dire che sia conforme ai principi REST.
- *Livello 1*: si utilizza il protocollo HTTP, ma a differenza del livello zero ogni risorsa ha una sua URL. A questo livello si utilizza ancora il metodo POST mettendo come parametro il nome del metodo da eseguire.
- *Livello 2*: da questo livello ogni comando viene mappato sul giusto metodo HTTP.
- *Livello 3*: rispetto al livello inferiore per ogni oggetto trasferito vengono aggiunti una serie di collegamenti che indicano quali azioni possono essere eseguite sull'oggetto.

3.1.4 Strumenti a supporto della produzione

Finora abbiamo visto tutte le tecnologie che sono state utilizzate durante lo sviluppo del codice di **IBD Tool**. In questa sezione ci concentreremo su tutti i prodotti che vengono utilizzati per eseguire e ospitare il codice sviluppato.

GitHub

GitHub è una piattaforma per la gestione del codice durante le fasi di sviluppo e produzione. Ogni utente una volta iscritto nella piattaforma, può gratuitamente creare delle *repository* in cui mettere il codice relativo ad un'applicazione. Le repository possono essere sia private che pubbliche. Nel mondo *open source* si sfrutta la possibilità di pubblicare codice su repository pubbliche in modo da diffondere i progetti e attirare nuovi collaboratori. Nel caso di progetti come IBD Tool invece si utilizzano le repository private in modo da proteggere il codice da possibili soggetti malintenzionati. GitHub permette di condividere permessi per l'accesso ad una repository privata tramite degli inviti, consentendo a più utenti di collaborare ad un progetto. Le operazioni base messe a disposizione sono:

- *Fork*: permette di creare un nuovo ramo (*branch*) del codice. Da questo momento in poi l'utente verrà spostato nel nuovo ramo, cosicché tutte le modifiche fatte da quel momento non cambieranno il codice presente nel ramo di partenza.

- *Pull*: permette di unire le modifiche fatte in locale con le ultime modifiche caricate sulla repository remota. Questa operazione ha effetto solo sul branch in cui la si esegue.
- *Commit*: registra le modifiche fatte sul branch, dando la possibilità di aggiungere un commento che descriva le operazioni fatte.
- *Push*: viene eseguita solitamente dopo il commit e permette di aggiornare la repository remota con le modifiche registrate nel commit.
- *Merge*: unisce il codice di due branch. Nel caso il software trovi conflitti tra le versioni, si dà all'utente la possibilità di decidere come risolverli.
- *Revert*: permette di ritornare alla versione del codice indicata nell'operazione

Su **IBD Tool** la divisione in branch è stata usata per dividere il codice di produzione da quello di sviluppo. I due differiscono solo per quanto riguarda la configurazione dei servizi a cui accedono. Infatti nell'ambiente di sviluppo si utilizzano database locali, indirizzi email differenti da quelli usati dall'applicazione e sia il front-end che il back-end sono configurati per rispondere all'indirizzo *localhost*. Invece nell'ambiente di produzione si utilizzano gli indirizzi del server ospitato su *Heroku* (servizio di cloud computing utilizzato per IBD Tool), del client caricato su *Firebase Hosting* e del database raggiungibile sulla piattaforma *MongoDB Atlas*. Successivamente la divisione in branch è stata utilizzata anche per parallelizzare il lavoro tra più programmatori. Ogni programmatore avendo un compito da eseguire, partiva da una versione stabile del progetto, creava la nuova funzionalità su un branch dedicato e successivamente faceva il merge sul ramo principale. Durante tutte queste fasi intermedie ogni programmatore aveva il proprio branch così da non interferire con il lavoro degli altri.

Heroku

Heroku è una piattaforma che fornisce un servizio di *cloud computing*. Il cloud computing deresponsabilizza gli sviluppatori di software dalla manutenzione di un server fisico, delegando al cloud i compiti. Su Heroku sono eseguibili programmi scritti in diversi linguaggi di programmazione, nel caso di **IBD Tool** è stato utilizzato per eseguire il codice Java del back-end. Una funzionalità molto interessante di Heroku è la possibilità di collegare un account *GitHub* in modo di fare i *deploy* automatici nel caso venga aggiornato il codice presente su GitHub.

Firebase

Firebase è una piattaforma utilizzata per lo sviluppo di applicazioni e fornisce una serie di servizi utili. In **IBD Tool** sono stati utilizzati solo due dei servizi disponibili:

- *Firebase Hosting*
- *Firebase Cloud Messaging*

Firebase Hosting si occupa dell'hosting di applicazioni sviluppate con Angular. In **IBD Tool** utilizziamo questo servizio per ospitare la parte del front-end, visitabile all'indirizzo *https://ibd-tool-mauriziano.web.app/*.

Viene messa a disposizione una **command-line interface** (**CLI**) per rendere facile l'interazione con Firebase tramite dei semplici comandi. Ad esempio una volta installata la CLI, per fare il deploy dell'applicazione sviluppata, bisogna accedere all'account di Firebase e dalla cartella del progetto eseguire i seguenti comandi:

```
ng build --prod
firebase deploy -m "Titolo del commit"
```

Firebase Cloud Messaging è il modulo che fornisce gli strumenti per la comunicazione tra dispositivi in tempo reale. A differenza della classica comunicazione vista nel modello REST, con questo modulo un *user agent* può ricevere aggiornamenti anche senza chiederle tramite richieste HTTP. Firebase Cloud Messaging deve essere gestito sia nel front-end che nel back-end.

Sul front-end bisogna eseguire in sequenza i seguenti passi:

1. richiedere all'utente il permesso di inviargli delle notifiche. Tipicamente questo viene fatto attraverso un pop-up. Una volta registrato il permesso non verrà più richiesto ai successivi accessi;

2. generare un token univoco che identifichi il dispositivo che ha accettato le notifiche. Questo token verrà inviato al server e verrà utilizzato per raggiungere lo user agent;

3. gestire i messaggi ricevuti, differenziando il loro trattamento in base allo scopo per cui sono stati mandati.

Sul back-end sfruttando anche le funzionalità messe a disposizione da un plugin Firebase per Java, dobbiamo:

1. salvare il token ricevuto dal front-end, in modo da poterlo recuperare dal database ogni volta che c'è la necessita di mandare un messaggio;

2. inviare delle notifiche personalizzate in base alle esigenze ai dispositivi.

Nel progetto **IBD Tool** abbiamo sfruttato il campo *type* del messaggio per differenziare le notifiche pop dalle notifiche dei nuovi messaggi nella sezione della chat.

3.2 Questionari

Per la piattaforma **IBD Tool** sono stati scelti una serie di questionari che monitorano il paziente sotto diversi aspetti. Di seguito vedremo che alcuni questionari dopo la compilazione forniscono un valore numerico chiamato *score*. Possiamo dividere i questionari in tre macro categorie:

1. *Questionari che monitorano l'attività di malattia*:

   - **HBI**: questionario utilizzato per monitorare la salute del paziente con malattia di Crohn considerando parametri clinici con focus sull'attività e dolore addominale. Il questionario restituisce un valore numerico che classifica il paziente in una delle seguenti categorie: *remissione* (valori inferiori a 5), *attività lieve* (da 5 a 7), *attività moderata* (da 8 a 16), *grave* (valori superiori a 16). Sia il medico che il paziente verranno informati:
     - dello score finale;
     - del risultato della classificazione;
     - se c'è stato un cambiamento di classificazione rispetto all'ultimo questionario compilato;
     - se c'è stata una variazione di score superiore ai due punti rispetto all'ultimo questionario compilato;
     - nel caso in cui lo score sia maggiore o uguale a otto punti, verrà indicato che è stata rilevata attività di malattia.
   - **PATIENT-SCCAI**: questionario per il monitoraggio della salute del paziente con rettocolite ulcerosa, che tiene conto del numero ed urgenza delle evacuazioni. Il questionario classifica la malattia come in *remissione* se lo score è inferiore a 5, altrimenti come *recidiva*. Sia il medico che il paziente vengono informati:
     - dello score finale;
     - del risultato della classificazione;
     - se c'è stato un cambiamento di classificazione rispetto all'ultimo questionario compilato;
     - se c'è stata una variazione di score superiore ai due punti rispetto all'ultimo questionario compilato;
     - nel caso in cui lo score sia maggiore o uguale a cinque punti, verrà indicato che è stata rilevata attività di malattia.
   - **MIAH-UC**: questionario per il monitoraggio della salute del paziente con rettocolite ulcerosa che tiene conto del dolore addominale e dell'urgenza evacuativa. Nel caso in cui lo score sia maggiore o uguale a 3.6 punti verranno informati sia il medico che il paziente che è stata rilevata attività di malattia;

- **MIAH-CD**: questionario per il monitoraggio della salute del paziente con malattia di Crohn che tiene conto del numero ed urgenza delle evacuazioni. Nel caso in cui lo score sia maggiore o uguale a 3.5 punti verranno informati sia il medico che il paziente che è stata rilevata attività di malattia.

2. *Questionari che monitorano l'impatto della malattia su diversi ambiti della vita del paziente*:

   - **PHQ9**: questionario utilizzato per la diagnosi, il monitoraggio e la determinazione della gravità della depressione. Il completamento del questionario restituisce uno score che classifica il paziente in uno tra i seguenti stati: da 0 a 4 *depressione assente*, da 5 a 9 *depressione sottosoglia*, da 10 a 14 *depressione maggiore lieve*, per valori maggiori di 20 *depressione maggiore severa*. In questo caso solo il medico verrà informato del risultato del questionario;
   - **IBD-DISK**: questionario per valutare l'impatto della malattia sulla vita del paziente. Il questionario parte dalle interazioni interpersonali, educazione e lavoro fino allo stato di salute in generale. Il medico e il paziente verranno informati dello score prodotto dal questionario.

Figura 3.4. Esempio grafico prodotto dalla compilazione di IBD-Disk

   - **PRISM**: valutazione grafica dell'impatto della malattia sulla vita del paziente. Il questionario consiste nell'indicare avvicinando una sfere rossa ad una gialla, quanto è grossa l'influenza che ha la malattia sulla vita del paziente;
   - **IPAQ-SF**: questionario sull'attività fisica quotidiana svolta negli ultimi 7 giorni. Il questionario classifica il paziente in base ad uno score: per valori inferiori a 700 viene classificato come *inattivo*, tra 700 e 2519 come *sufficientemente attivo*, per valori maggiori di 2520 come *attivo o molto attivo*;

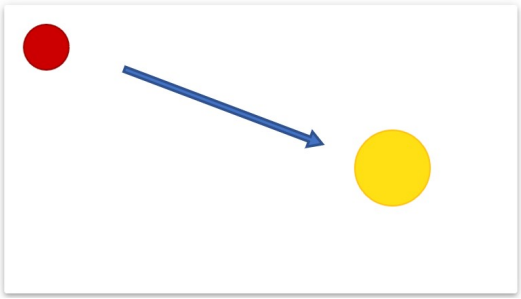


Figura 3.5. Esempio di un PRISM

- **PSQI**: questionario per la valutazione del sonno nell'ultimo mese. Il paziente verrà classificato in due possibili classi: *remissione* se lo score è minore di 5, altrimenti *recidivo*;
- **WPAI**: questionario utilizzato per il monitoraggio dell'influenza della malattia sull'attività lavorativa del paziente;
- **EQ5D5L**: questionario per il monitoraggio delle attività abituali del paziente quali: lavoro e cura della persona ma anche riguardante la salute fisica e psicologica.

3. *Questionari che misurano la compatibilità con la terapia*:

   - **MMAS8**: questionario sulla costanza nell'assunzione dei farmaci. Il questionario restituisce uno score che classifica se il paziente è *non aderente alla terapia* (score inferiore a 6), oppure *aderente alla terapia* (score tra 6 e 8). Sia il medico che il paziente verranno informati del risultato della classificazione;
   - **TSQM**: questionario che valuta il livello di soddisfazione del paziente riguardo alla terapia farmacologica in uso.

Ad ogni paziente iscritto alla piattaforma verrà assegnata in maniera casuale una categoria tra:

- *Telemedicina*;
- *Cure tradizionali*.

Al momento la differenza tra le due categorie è data dal numero e dalla frequenza dei questionari inviati automaticamente dal sistema. In Fig. 3.6 possiamo notare la cadenza con il quale ogni questionario viene inviato ai pazienti facenti parte delle varie categorie.

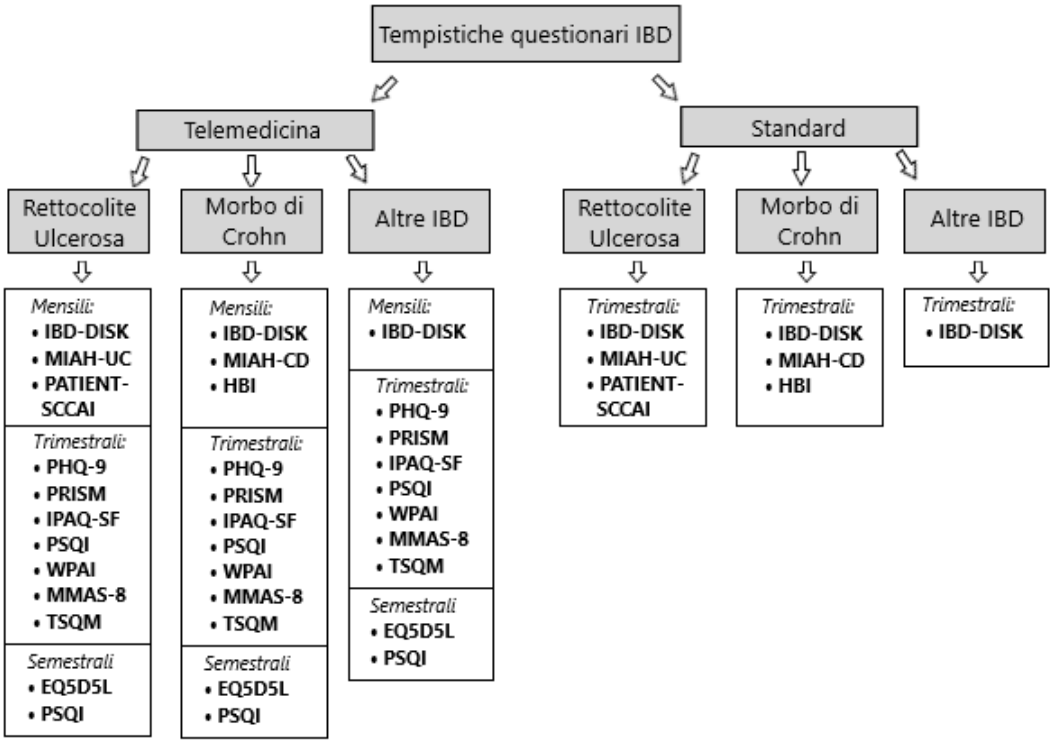


Figura 3.6. Tempistiche di invio dei questionari all'interno di IBD Tool

Capitolo 4

Risultati

Come abbiamo visto nei precedenti paragrafi, questa tesi ha come obiettivo l'ampliamento delle funzionalità disponibili all'interno di **IBD Tool**. Nei prossimi paragrafi tratteremo le modifiche fatte all'applicazione durante questa tesi. I primi paragrafi parleranno di come sono stati migliorati alcuni aspetti già presenti nell'applicazione. Successivamente invece vengono descritte le nuove funzionalità che sono state sviluppate per aumentare le operazioni possibili.

4.1 Miglioramento delle funzionalità esistenti

Durante una fase iniziale di studio e monitoraggio dei comportamenti dell'applicazione, sono stati individuati alcuni punti migliorabili. Di seguito analizzeremo i più importanti.

4.1.1 Invio anomalo di questionari

Il primo aspetto analizzato riguarda la calendarizzazione dei questionari spediti in automatico. È stato infatti notato che l'applicazione non svolgeva questo incarico con le tempistiche giuste. Dal normale funzionamento del programma ci si aspettava che i questionari venissero mandati solo nei seguenti casi:

- All'iscrizione di un nuovo paziente alla piattaforma, in questa fase il programma invia per i pazienti in *telemedicina* i questionari mensili, mentre per quelli sotto *cure tradizionali*, i questionari trimestrali.

- Ogni uno, tre o sei mesi, a secondo della frequenza programmata. La cadenza per ogni questionario la troviamo descritta in Fig. 3.6. Le frequenze di invio non sono fisse, ma al medico è consentito cambiarle specificando di quale determinato questionario di un determinato paziente si vuole effettuare la modifica, fino ad un massimo di un questionario al giorno.

- Quando il medico decide di inviare il questionario al paziente cliccando su un apposito bottone presente nella sezione medici. Questo bottone permette una volta selezionato un paziente, di inviargli un determinato questionario fino ad un massimo di un questionario al giorno.

Nonostante ci fossero solo queste tre possibilità, alcuni pazienti ricevevano questionari quotidianamente, sebbene il medico non avesse modificato le frequenze di invio.

Una volta compreso il problema, si è cercato quale potesse essere la causa. La si è trovata nell'invio automatico dei questionari a scadenze prestabilite. A questo punto è stata messa sotto analisi la logica che gestiva questa funzionalità. La risoluzione del problema è passata attraverso tre fasi:

1. studiare il codice che permetteva la funzionalità di invio automatizzato;
2. comprendere cosa scatenava l'effetto indesiderato;
3. correggere il funzionamento del programma nel più breve tempo possibile.

Dallo studio del codice si è scoperto che la logica si basava sull'utilizzo di processi asincroni. Quindi per ogni programmazione di un questionario di un singolo paziente, veniva generato uno *ScheduledFuture*. Lo ScheduledFuture è un oggetto messo a disposizione nel package *java.util.concurrent*, ed il suo scopo è quello di ritardare un'azione per un tempo inserito in input e in caso ci fosse bisogno, eseguirla in parallelo al flusso del programma principale. Il programma non viene arrestato in attesa che passi il tempo inserito, ma continua la sua normale esecuzione fino a quando una volta scaduto il tempo, verrà richiamato il metodo dichiarato. Quindi lo ScheduledFuture veniva utilizzato per pianificare l'invio di un questionario ad una certa data. Inoltre, sul database veniva gestita una collezione di documenti chiamati *Job*. Il loro scopo era di ricordare quali ScheduledFuture fossero stati lanciati e in quale momento ci si aspettava l'invio del questionario. Questo permetteva di prevenire che gli arresti dell'applicazione (es. per applicare un aggiornamento) cancellassero l'invio pianificato di tutti i questionari. Infatti, dopo un arresto dell'applicazione tutti gli ScheduledFuture vengono persi, ma utilizzando la data di consegna del questionario inserita nel *Job*, era possibile all'avvio dell'applicazione ricostruire tutti gli ScheduledFuture.

Non essendo riusciti a trovare un errore a livello logico, la colpa del malfunzionamento si è imputata alla configurazione della programmazione asincrona. La configurazione infatti risulta leggermente verbosa e poco intuitiva. Inoltre, il problema richiedeva di essere risolto nei tempi più brevi possibili, perché la quantità troppo elevata di questionari generava problemi di spam ai pazienti, che di conseguenza potevano decidere di abbandonare l'utilizzo dell'applicazione. Per questi motivi si è deciso di riscrivere la parte che gestiva questa funzionalità utilizzando un approccio differente. Al codice si è aggiunto in primo luogo un metodo annotato

con **@Scheduled**, che accetta come parametri tra parentesi la cadenza con il quale deve essere eseguito. Sfruttando questa possibilità abbiamo creato un metodo che giornalmente controlla quali pazienti devono ricevere dei questionari e infine invia loro una e-mail per notificare l'arrivo di un nuovo questionario. Il metodo controlla dei campi opportunamente aggiunti nella collezione *users*. Questi campi sono in totale quattro, ma i primi tre che vedremo sono utilizzati in mutua esclusione rispetto al quarto. I campi sono:

- **timestampLastMonthlyQuestionnaire**, **timestampLastThreeMonthly Questionnaire**, **timestampLastBiannualQuestionnaire**: sono tre campi di tipo *long* che servono per ricordarsi quando sono stati mandati rispettivamente gli ultimi questionari mensili, trimestrali e semestrali;
- **timerAndTimestampQuestionnaireMap**: è una mappa che contiene come chiave il nome di un questionario, e come valore una coppia di informazioni che indicano quando è stato mandato l'ultima volta il questionario specificato nella chiave e quanto tempo deve passare tra un invio e l'altro. Questa mappa viene interrogata solo nel caso in cui il valore di un flag chiamato *deadlineQuestionnaireModified* ha il valore *true*.

La mappa quindi viene interrogata solo quando un dottore modifica manualmente le ricorrenze dei questionari. Questa soluzione risulta molto semplice e permette di alleggerire il programma dalla gestione di migliaia di *ScheduledFuture*.

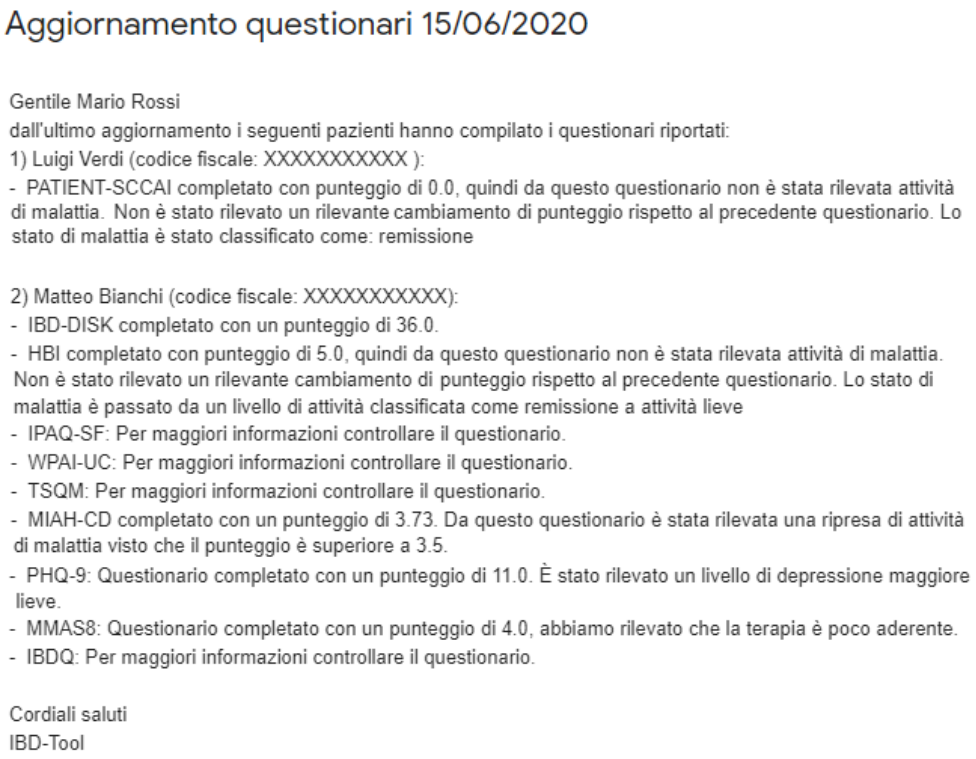
4.1.2 Riassunto quotidiano sui questionari compilati

L'applicazione permette di aggiornare i medici su quali questionari siano stati compilati e quali aggiornamenti importanti ne derivino (es. una ripresa di attività di malattia). Questo tipo di informazione viene data tramite e-mail alla casella di posta indicata dal medico. Inizialmente la logica dell'applicazione imponeva che per ogni questionario compilato, il sistema reagisse creando subito più di un'e-mail. Il contenuto di ogni e-mail informava il medico di una delle seguenti novità:

- nuovo questionario compilato;
- classificazione dell'attività di malattia risultante dal questionario. Nel caso in cui nel precedente questionario la classificazione risultasse diversa, viene inoltre notificato anche l'avvenuto cambiamento di classe indicando quale fosse la precedente;
- variazione di più o meno di due punti rispetto al questionario precedente.

Mentre il primo tipo di informazione viene fornita per tutti i questionari, invece la classificazione e l'analisi della variazione del punteggio si può fornire solo per un sottoinsieme.

Da quanto detto, si può dedurre che questo tipo di informazione risulta poco ordinata. Infatti per ogni questionario compilato vengono generate da una a tre e-mail, richiedendo al medico di dover aprire centinaia di e-mail al giorno contenenti informazioni che possono essere contratte in un unico resoconto. Da qui l'idea di creare un sistema che giornalmente organizzi una singola e-mail per notificare al medico quali questionari sono stati compilati da ogni singolo paziente. Per fare questo è stato necessario aggiungere alle collezioni del database un'altra che contenesse quali questionari erano stati compilati e quali informazioni estrapolare. Per la realizzazione è stato scritto un metodo annotato con *@Scheduled* che giornalmente si occupa di leggere da questa collezione, di scrivere in un'unica e-mail i dati raccolti e infine di spedirla.



Aggiornamento questionari 15/06/2020

Gentile Mario Rossi
dall'ultimo aggiornamento i seguenti pazienti hanno compilato i questionari riportati:
1) Luigi Verdi (codice fiscale: XXXXXXXXXXX):
- PATIENT-SCCAI completato con punteggio di 0.0, quindi da questo questionario non è stata rilevata attività di malattia. Non è stato rilevato un rilevante cambiamento di punteggio rispetto al precedente questionario. Lo stato di malattia è stato classificato come: remissione

2) Matteo Bianchi (codice fiscale: XXXXXXXXXXX):
- IBD-DISK completato con un punteggio di 36.0.
- HBI completato con punteggio di 5.0, quindi da questo questionario non è stata rilevata attività di malattia. Non è stato rilevato un rilevante cambiamento di punteggio rispetto al precedente questionario. Lo stato di malattia è passato da un livello di attività classificata come remissione a attività lieve
- IPAQ-SF: Per maggiori informazioni controllare il questionario.
- WPAI-UC: Per maggiori informazioni controllare il questionario.
- TSQM: Per maggiori informazioni controllare il questionario.
- MIAH-CD completato con un punteggio di 3.73. Da questo questionario è stata rilevata una ripresa di attività di malattia visto che il punteggio è superiore a 3.5.
- PHQ-9: Questionario completato con un punteggio di 11.0. È stato rilevato un livello di depressione maggiore lieve.
- MMAS8: Questionario completato con un punteggio di 4.0, abbiamo rilevato che la terapia è poco aderente.
- IBDQ: Per maggiori informazioni controllare il questionario.

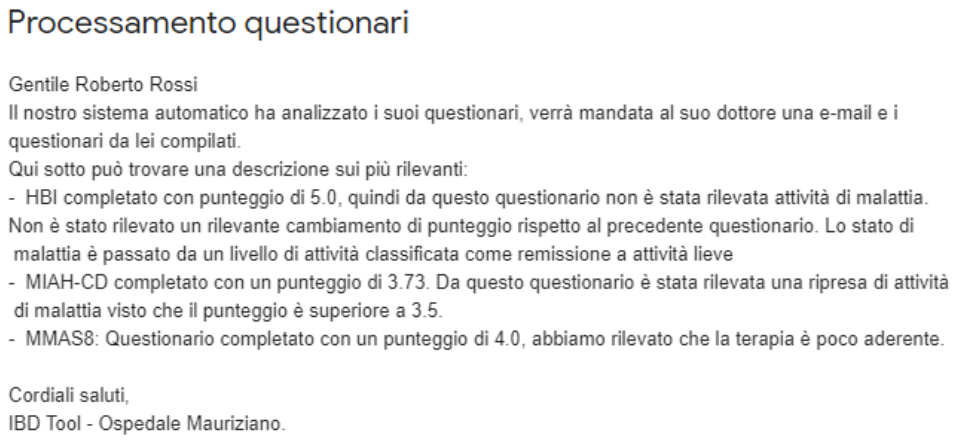
Cordiali saluti
IBD-Tool

Figura 4.1. Esempio di una e-mail riassuntiva indirizzata ad un medico

Inoltre utilizzando questo processo, si è successivamente aggiunta la funzionalità che permette di informare anche i pazienti sui risultati dei questionari. In questa ulteriore e-mail verranno elencati i risultati dell'elaborazione dei questionari e se da questi è stata riscontrata attività di malattia. A differenza della e-mail creata per i medici, in questa non vengono elencati tutti i questionari compilati, ma solo i seguenti:

- *MIAH-UC*

- *MIAH-CD*
- *PATIENT-SCCAI*
- *HBI*
- *MMAS-8*


Processamento questionari

Gentile Roberto Rossi
Il nostro sistema automatico ha analizzato i suoi questionari, verrà mandata al suo dottore una e-mail e i questionari da lei compilati.
Qui sotto può trovare una descrizione sui più rilevanti:
- HBI completato con punteggio di 5.0, quindi da questo questionario non è stata rilevata attività di malattia. Non è stato rilevato un rilevante cambiamento di punteggio rispetto al precedente questionario. Lo stato di malattia è passato da un livello di attività classificata come remissione a attività lieve
- MIAH-CD completato con un punteggio di 3.73. Da questo questionario è stata rilevata una ripresa di attività di malattia visto che il punteggio è superiore a 3.5.
- MMAS8: Questionario completato con un punteggio di 4.0, abbiamo rilevato che la terapia è poco aderente.

Cordiali saluti,
IBD Tool - Ospedale Mauriziano.

Figura 4.2. Esempio di una e-mail riassuntiva indirizzata ad un paziente

4.1.3 Aumento della sicurezza nelle registrazioni

Gli utenti registrabili nell'applicazione si dividono in due categorie: i medici e i pazienti. L'iscrizione di un **paziente** avviene sotto il controllo del medico, che accedendo alla sua pagina personale può inserire in maniera affidabile i dati del paziente. A questo punto, l'utente appena registrato riceverà una e-mail contenente le sue credenziali ed un invito a cambiare la password generata automaticamente con una più personale. La generazione della password automatica è stata cambiata durante questa tesi. Precedentemente la password era composta da 4 caratteri casuali, mentre ora viene creata da un algoritmo che produce una password di otto caratteri con:

- almeno una lettera minuscola;
- almeno una lettera maiuscola;
- almeno una cifra;
- almeno un carattere speciale.

Nonostante la password venga generata in maniera sicura, si richiede di cambiarla il prima possibile perché essendo comunicata via e-mail è visibile a chi ha accesso alla casella di posta elettronica dell'applicazione.

La registrazione di un utente come **medico** invece non era controllata dal sistema. Quindi, per rimediare a questo problema, sono stati aggiunti il passo 3 e 4 (indicati in seguito) nel percorso che porta all'iscrizione di un medico. I passaggi sono:

1. l'utente che vuole registrarsi compila un form in cui inserisce: indirizzo e-mail, nome, cognome, codice fiscale, data di nascita, numero di telefono e password.

2. una e-mail contenente un link di conferma con validità di un giorno viene inviata alla casella postale dichiarata dal nuovo utente.

3. una volta confermata l'e-mail viene inviato un messaggio contenente i dati dell'utente alla casella di posta *ibd.conferma.registrazioni@gmail.com*. A questo punto un operatore decide chi ha il permesso di registrarsi e chi no.

4. se viene accettato il nuovo utente verrà mandata una e-mail di benvenuto e il nuovo utente potrà finalmente fare accesso. Altrimenti il processo si conclude eliminando la richiesta senza notificare l'esito al richiedente.

Inoltre durante lo sviluppo di questi nuovi passaggi è stata cancellata dall'e-mail di benvenuto al medico, il riepilogo delle credenziali contenente la password. Questo è stato fatto perché a differenza del paziente, il medico ha la possibilità di scegliere la password durante la registrazione, quindi comunicarla tramite e-mail la rendeva vulnerabile non solo su IBD Tool, ma anche sugli altri siti in cui è stata utilizzata.

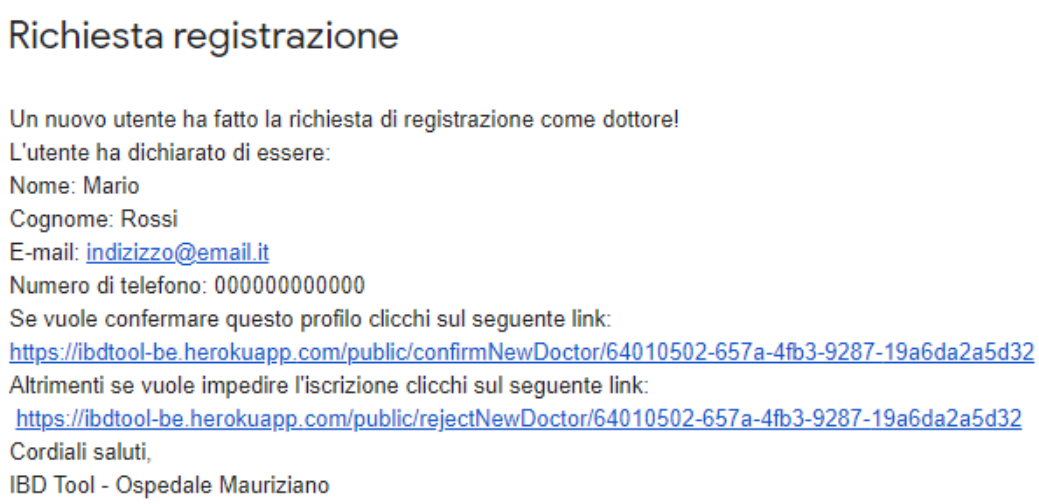
4.1.4 Gestione password

Cambio password

Il cambiamento della password è la prima operazione che viene richiesta a qualsiasi paziente all'interno dell'applicazione. Per questo motivo è stata costruita una finestra che rendesse il processo il più *user friendly* possibile. In precedenza, l'unico metodo per cambiare la password era tramite la procedura di recupero password. Questa procedura si divide nei seguenti passi:

1. inserire in un form indirizzo e-mail e codice fiscale. Questo form è raggiungibile, solo se non si è fatto accesso alla pagina personale, attraverso un link presente nella pagina d'accesso, altrimenti visitando l'indirizzo: *https://ibd-tool-mauriziano.web.app/change-password*.

2. una volta completato il primo passo si riceve una e-mail contenente l'url da visitare per cambiare la password. L'url rimane valida per la durata di un giorno;


Richiesta registrazione

Un nuovo utente ha fatto la richiesta di registrazione come dottore!
L'utente ha dichiarato di essere:
Nome: Mario
Cognome: Rossi
E-mail: indizizzo@email.it
Numero di telefono: 000000000000
Se vuole confermare questo profilo clicchi sul seguente link:
https://ibdtool-be.herokuapp.com/public/confirmNewDoctor/64010502-657a-4fb3-9287-19a6da2a5d32
Altrimenti se vuole impedire l'iscrizione clicchi sul seguente link:
https://ibdtool-be.herokuapp.com/public/rejectNewDoctor/64010502-657a-4fb3-9287-19a6da2a5d32
Cordiali saluti,
IBD Tool - Ospedale Mauriziano

Figura 4.3. Esempio della e-mail generata per il controllo di un nuovo utente medico

3. cliccato il link si apre una pagina contenente un form in cui inserire la nuova password.

Per semplificare questo processo quindi si è creata una nuova pagina all'interno della pagina personale che permettesse di cambiare la password in un unico passaggio. Il form richiede di inserire la vecchia password e due volte la nuova (Fig. 4.4).

Durante lo sviluppo di questa funzionalità, per le password si è voluto aggiungere il requisito di 8 caratteri minimi. Inoltre in testa al form è stata aggiunta un'informativa che spiegasse come generare una password sicura. Si è preferito consigliare di generare una password che rispetti tutti i requisiti indicati in Fig. 4.4 rispetto che renderli obbligatori, in quanto non tutti gli utenti sono capaci di gestire password così elaborate.

Perfezionamento ripristino password

Come appena descritto, la funzionalità di ripristino password era già presente nell'applicazione. Questa sezione però ha portato diversi utenti a segnalare un errore nella procedura nel momento in cui viene richiesto l'inserimento dell'e-mail e del codice fiscale. La radice del malfunzionamento è stata localizzata nel metodo che attraverso e-mail e codice fiscale, chiedeva al database l'oggetto rappresentante l'utente che faceva la richiesta. Infatti questo metodo era *case sensitive* e quindi restituiva un errore ogni volta che gli utenti inserivano il codice fiscale in minuscolo. Per risolvere il problema è stato sufficiente inserire l'annotazione **@Query** sopra il metodo usato per la ricerca dell'utente nel seguente modo:

```
@Query("{$and:[{email: ?0}, {SSN:{$regex :?1, $options:'i'}}]}")
```

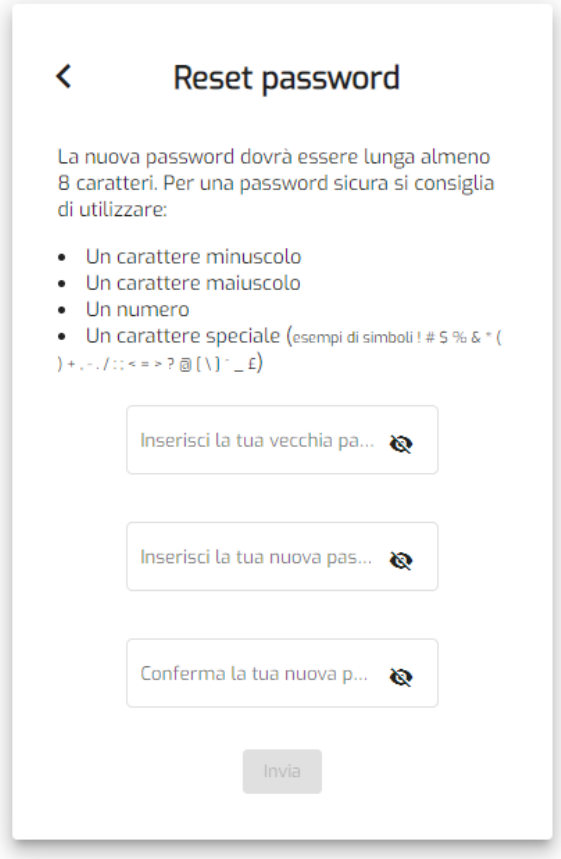


Figura 4.4. Form per il cambio password

Dove *$options: 'i'* indica che la ricerca sul codice fiscale deve essere *case insensitive*

4.1.5 Revisione questionari

Alcuni tra i questionari già presenti nell'applicazione sono stati modificati perché presentavano alcune inesattezze. Tra questi ci sono:

- **MIAH-UC** e **MIAH-CD**: l'algoritmo per il calcolo del punteggio non era ancora noto, quindi una volta ricevuto dai collaboratori dell'ospedale Mauriziano, è stata inserita la versione corretta. Inoltre è stato aggiornato il punteggio dei questionari compilati prima di questa modifica. Infine è stato cambiato il testo della prima domanda di entrambi i questionari, in quanto poteva risultava ambigua;
- **PATIENT-SCCAI**: tra le risposte multiple sono state rimosse alcune delle possibili scelte che permettevano al paziente di non rispondere alla domanda.

Quindi per una maggiore precisione è stata rimossa questa opzione;

- **PRISM**: cambiate le indicazioni per la compilazione del questionario in modo da renderlo più comprensibile;
- **IPAQ-SF**: cambiata l'interfaccia grafica che non permetteva di lasciare dei campi vuoti nonostante il testo del questionario in alcuni casi lo richiedesse.

4.2 Creazione nuove funzionalità

In collaborazione con i medici del reparto di gastroenterologia dell'Azienda Ospedaliera Ordine Mauriziano di Torino, sono state pensate ed elaborate una serie di nuove funzionalità. L'obiettivo è quello di aumentare il numero di operazioni possibili, rendendo sempre più utile e semplice l'utilizzo di **IBD Tool**. Di seguito vedremo le più importanti.

4.2.1 Riepilogo questionari non letti

Dall'utilizzo dell'applicazione è nata l'esigenza di perfezionare la procedura che permette ad un medico di controllare i questionari compilati dai pazienti. Infatti i passi che quotidianamente un medico doveva fare per la lettura dei questionari appena compilati erano i seguenti:

1. scoprire quali pazienti hanno compilato dei questionari. Per fare questo si può usare l'e-mail che quotidianamente informa il dottore su tutti i questionari compilati.
2. andare nella pagina dei pazienti personali e per ciascun paziente presente nella e-mail bisogna:
   a) cercarlo tramite nome e cognome
   b) entrare nella pagina dei suoi questionari
   c) per ciascun questionario entrare nella cartella del questionario e selezionare il questionario non letto.

Visto che il punto numero due deve essere eseguito per ogni paziente, potenzialmente può richiedere un elevato numero di iterazioni. Da qui nasce l'esigenza di avere uno specchietto unico contenente tutti i questionari non letti. Inoltre utilizzando questo specchietto si elimina la possibilità che qualche questionario non venga letto per errore. Per realizzarlo si è creata una pagina dedicata in cui è presente una tabella con l'elenco dei questionari non letti. Per aprire un questionario basterà cliccare su una delle righe della tabella. Il questionario appena letto verrà quindi subito rimosso dalla tabella. Sulla tabella è possibile fare un filtraggio sui tipi

di questionari oppure è possibile ordinare le righe per nome, cognome, punteggio questionario, tipo questionario oppure data di compilazione. Inoltre sono disponibili delle check-box che permettono di segnare come letti più di un questionario alla volta. Questo può essere utile nel caso il medico preferisca conoscere solo il punteggio del questionario senza doverlo necessariamente aprire.

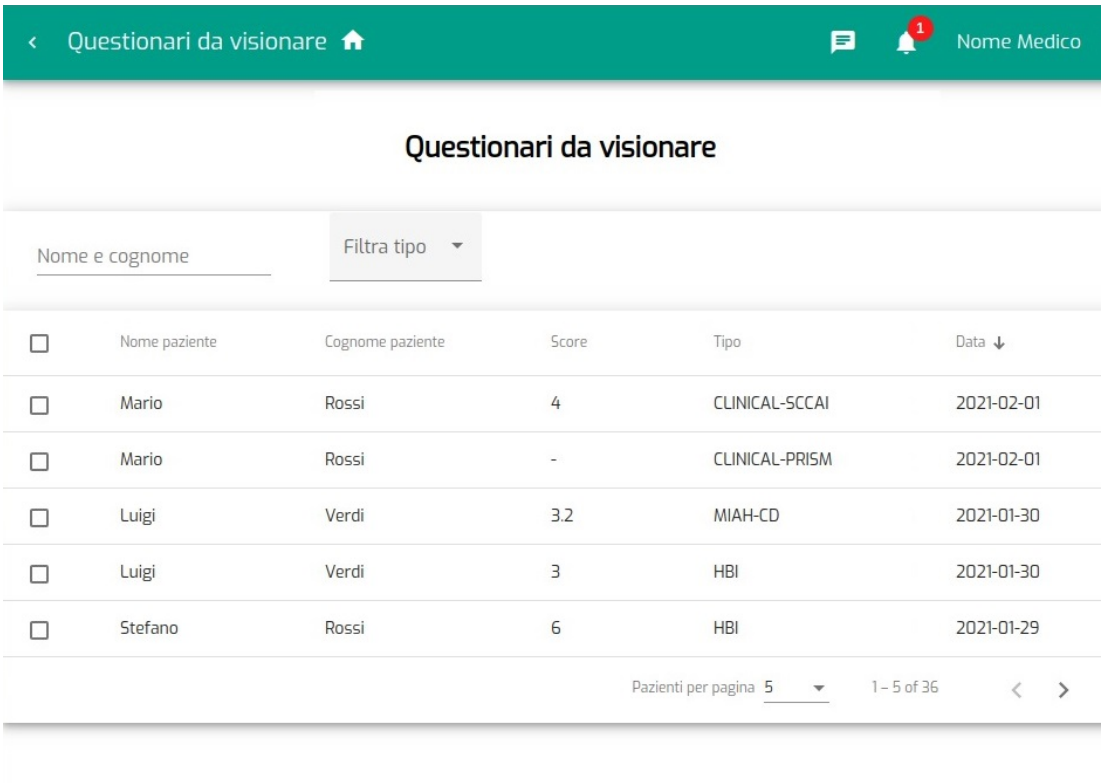


Figura 4.5. Specchietto per la visualizzazione dei questionari non letti

4.2.2 Gestione pazienti inattivi

L'attività di monitoraggio dell'applicazione richiedeva la creazione di una funzionalità che permettesse ai medici di conoscere quali tra i pazienti iscritti non rispondono ai questionari forniti. Infatti è possibile che qualche paziente abbia inizialmente dato il consenso ad utilizzare IBD Tool, ma successivamente, per qualche motivo, abbia abbandonato l'applicazione. I motivi possono essere vari, quindi il medico potrebbe voler contattare il paziente per comprendere meglio perché ha smesso di utilizzare questo strumento.

Per realizzare questa funzionalità si è deciso di creare uno specchietto raggiungibile tramite un bottone presente nel pannello dei pazienti personali. Lo specchietto

è composto da una descrizione iniziale e da una tabella in cui ogni riga rappresenta un paziente *inattivo*. Per ogni paziente è possibile visionare la scheda personale, controllare i questionari o eliminarlo dall'applicazione. L'informativa presente in testa alla tabella spiega secondo quale principio i pazienti vengono catalogati come inattivi. Per fare ciò, viene fatta innanzitutto una distinzione tra i pazienti in telemedicina e quelli seguiti con cure tradizionali. Infatti i pazienti in telemedicina ricevendo i questionari mensilmente, si è deciso di utilizzare dei parametri più restringenti rispetto a quelli dei pazienti seguiti con cure tradizionali, che invece ricevono un questionario ogni tre mesi. Quindi per essere catalogati come inattivi, i pazienti in telemedicina non devono collegarsi all'applicazione da almeno 4 mesi, mentre per i pazienti seguiti con cure tradizionali si contano 7 mesi dall'ultimo accesso.

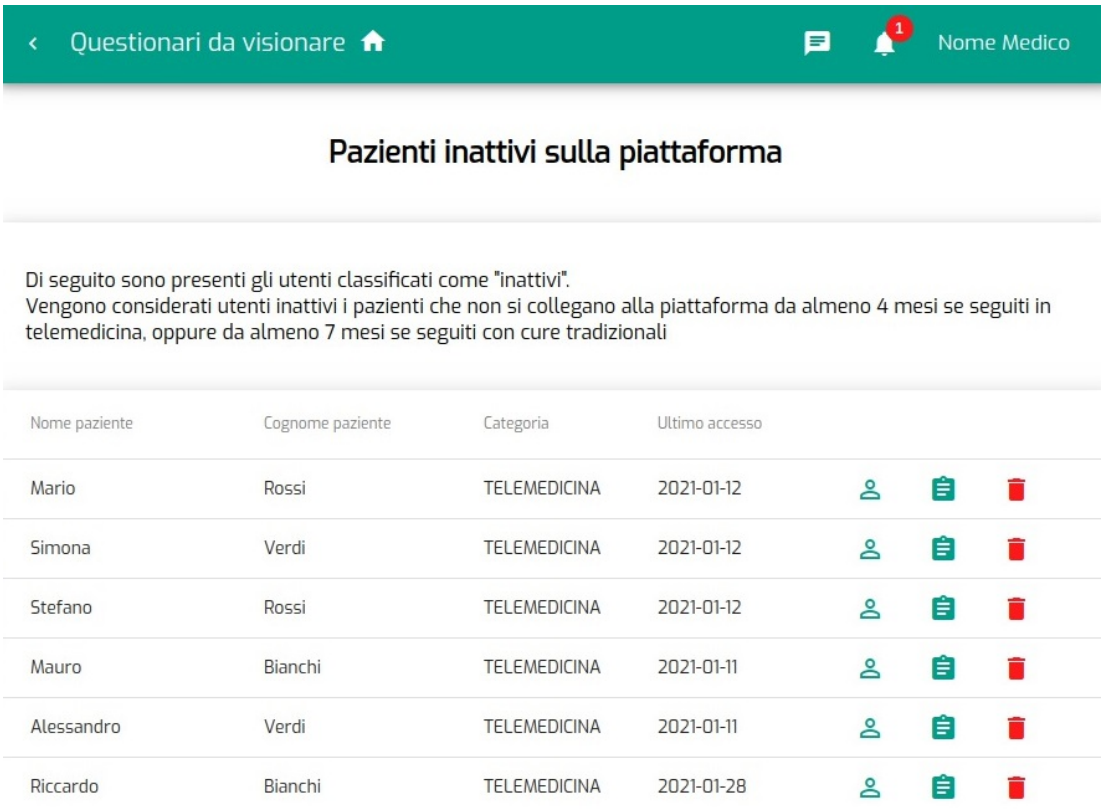


Figura 4.6. Specchietto per la visualizzazione dei pazienti inattivi

Il codice sviluppato per realizzare questo specchietto è stato riutilizzato in seguito durante lo sviluppo di una pagina in cui sono riassunti i dati dell'applicazione utilizzando una serie di grafici. Infatti tra questi ne esiste uno che permette di visualizzare quanti sono i pazienti inattivi e le relative percentuali in telemedicina e seguiti con cure tradizionali.

4.2.3 Nuova home page per i pazienti

Con l'obiettivo di ospitare due nuove funzionalità che vedremo nei prossimi due paragrafi, è stata modificata l'interfaccia della home page per i pazienti. In Fig. 4.7 vediamo che nella parte bassa della finestra è stata aggiunta un'icona che reindirizza ad una pagina in cui il paziente potrà interagire autonomamente con il medico. I nuovi modi per interagire sono due: attraverso i questionari o attraverso una chat dedicata.

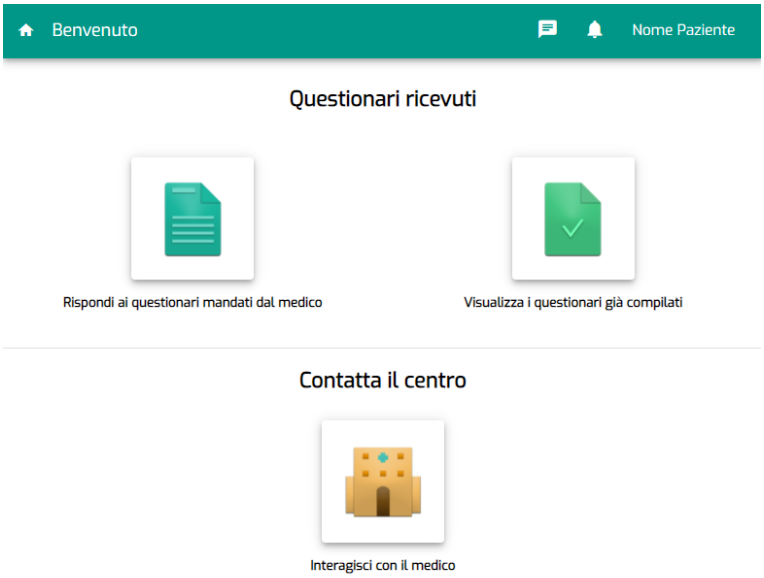


Figura 4.7. Home page per i pazienti aggiornata

4.2.4 Compilazione questionari non programmati

Come abbiamo visto, IBD Tool si basa sull'invio automatico di questionari. Però spesso può essere utile lasciare al paziente la possibilità di informare il medico con un questionario, anche se non gli è stato richiesto dall'applicazione. I motivi che spingono un paziente ad avere questo tipo di esigenza possono essere:

- volontà di notificare un cambiamento nello stato della malattia;
- correzione di un questionario compilato in maniera errata;
- volontà di aggiornare più frequentemente il medico curante.

Per realizzare questa funzionalità, si è creata una pagina (rappresentata in Fig. 4.9) in cui vengono elencati i questionari compilabili divisi in sottogruppi. Per

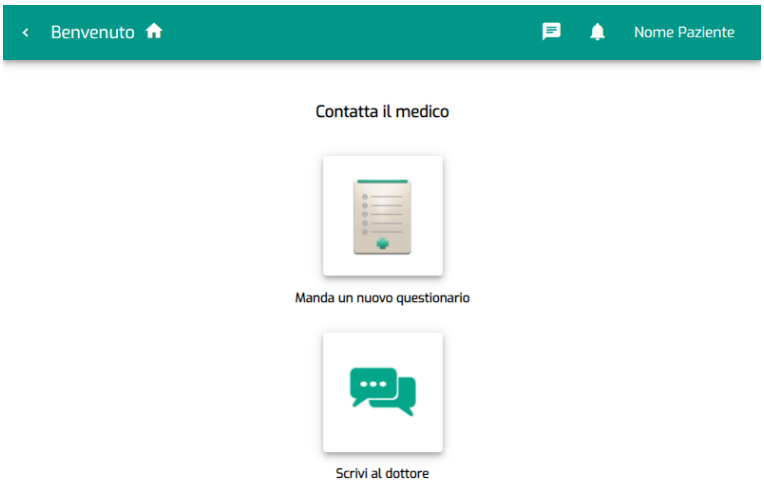


Figura 4.8. Pagina che permette di scegliere come interagire con il medico

ogni questionario viene inoltre aggiunta una descrizione che permette al paziente di identificare quale sia il più adatto per ciò che vuole comunicare. Basterà quindi cliccare sul questionario che si è scelto per poter iniziare la compilazione.

Durante lo sviluppo si è scelto di mantenere invariata la logica che impedisce di sottomettere più questionari dello stesso tipo in un singolo giorno. Per fare questo si è fatto in modo che in caso fosse già stato compilato un questionario di quel tipo, nel momento in cui viene scelto la seconda volta, un avviso notifica che quel questionario non è più compilabile per tutta la giornata (rappresentato in Fig. 4.10). Inoltre, nel caso un questionario compilato tramite questa funzionalità sia presente nella cartella dei questionari da compilare inviati automaticamente, per evitare che il paziente provi a ricompilarlo, viene eliminato dalla cartella.

4.2.5 Servizio di messaggistica in tempo reale

Lo scambio di questionari non può essere l'unico modo in cui medici e pazienti comunicano. Durante il monitoraggio del paziente spesso nasce la necessità di avere un canale diretto che colleghi il medico al paziente e viceversa. Una serie di possibili scenari che evidenziano in quali casi può essere necessario utilizzare questo canale sono:

- *da parte del medico al paziente*:
  - comunicare un cambiamento nel piano terapeutico;
  - approfondire quanto letto su un questionario;

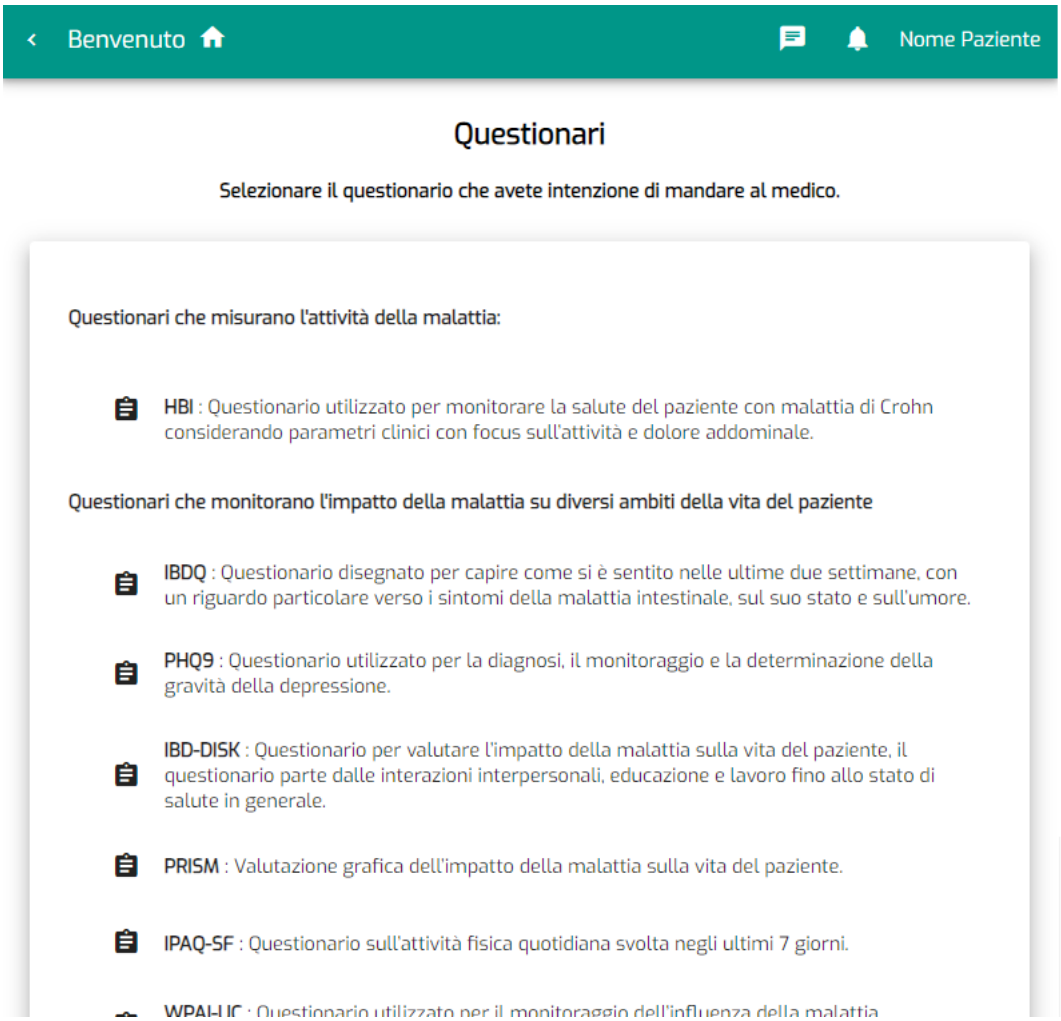


Figura 4.9. Pagina per selezionare il questionario da inviare

- *da parte del paziente al medico*:
  - richiedere informazioni sulla terapia in corso;
  - comunicare l'insorgenza di un problema legato alla terapia;
  - necessità di chiarimenti riguardo un questionario;
  - spedire il referto di un'analisi.

Precedentemente al lavoro svolto durante questa tesi, le comunicazioni erano gestite attraverso l'utilizzo della posta elettronica. Questo servizio, utilizzato come supporto a IBD Tool, presenta alcune criticità che vedremo di seguito:

- *collaborazione tra più medici.* Non è facilmente realizzabile un servizio di cooperazione che permetta ad un gruppo di medici di rispondere alle domande

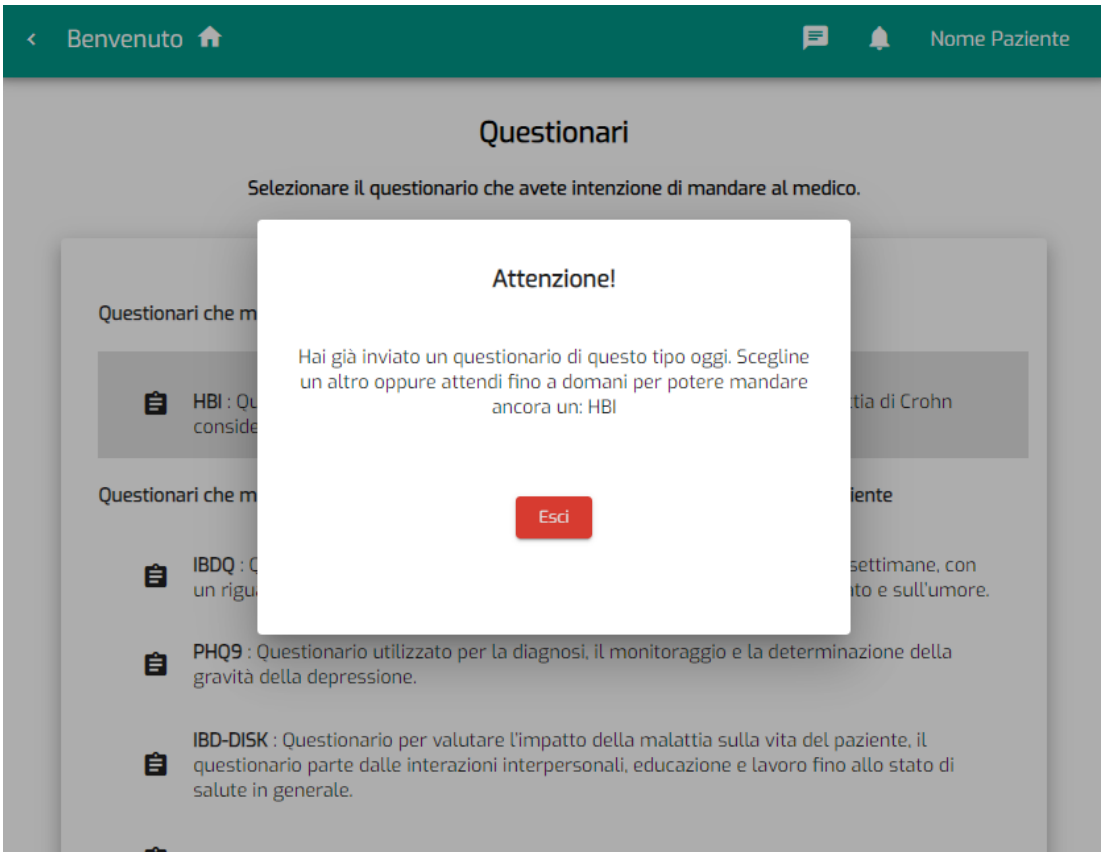


Figura 4.10. Avviso che impedisce di compilare due questionari dello stesso tipo

dei pazienti. Infatti, per permettere questa di collaborazione è necessario che il paziente che inizia la conversazione conosca l'indirizzo di tutti i medici e li metta in copia alla mail. Questa soluzione presenta subito un problema di gestione degli indirizzi e-mail che devono essere ricordati dal paziente. Inoltre la gestione degli indirizzi diventa ancora più complessa nel caso in cui il gruppo di medici che collabora al progetto cambi nel tempo;

- *gestione della cronologia messaggi.* Il servizio di posta elettronica non è particolarmente indicato se si vuole avere un quadro immediato dei messaggi scambiati nel tempo. Un utente per controllare tutti i messaggi scambiati, deve innanzitutto fare il filtro in base all'indirizzo e-mail dell'interlocutore e in seguito aprire singolarmente tutti i messaggi. Inoltre, è possibile che l'interlocutore utilizzi più di un indirizzo per comunicare, richiedendo all'utente di ricordare tutti gli indirizzi utilizzati, altrimenti c'è la possibilità che si perda qualche informazione;

- *filtraggio dello spam.* Gli utenti molto frequentemente comunicano utilizzando

un indirizzo di posta elettronica comune per diversi servizi. Alcuni utenti meno propensi alla manutenzione della casella di posta elettronica, possono avere delle grosse quantità di *spam* che rischiano di nascondere l'arrivo di messaggi importanti, come ad esempio quelli di un medico.

Dall'analisi di queste problematiche e dalla volontà di arricchire l'insieme delle funzionalità disponibili su **IBD Tool**, è stato elaborato un sistema di messaggistica integrato nell'applicazione. Dal punto di vista tecnico si è sfruttato l'utilizzo di **Firebase Cloud Messaging**, il quale veniva già utilizzato per la gestione degli *alert*. L'invio di un messaggio da parte di un utente è gestito in maniera molto semplice dal front-end. Il client eseguirà verso il server una richiesta di tipo *POST*, inserendo il messaggio nel corpo della richiesta.

Invece la logica che gestisce l'aggiornamento dei messaggi ricevuti è molto più complessa. Viene utilizzato Firebase Cloud Messaging per notificare all'utente l'arrivo di un nuovo messaggio. Vediamo di seguito i passaggi su cui si basa la logica alla base della ricezione dei messaggi:

1. l'utente ogni volta che accede alla piattaforma attraverso un nuovo dispositivo dà il consenso alla ricezione di notifiche su quel dispositivo. L'utilizzo delle notifiche permette di non generare traffico inutile, evitando al client di eseguire un *polling* continuo verso il server, nella ricerca di nuovi messaggi. Nel caso l'utente decida di bloccare le notifiche, se l'applicazione è chiusa o in background, non potrà essere avvisato di un nuovo messaggio. Nonostante questo aspetto, se l'applicazione è in esecuzione il servizio di messaggistica continuerà a funzionare implementando un polling in cui si richiede una volta al minuto se il server ha dei nuovi messaggi;

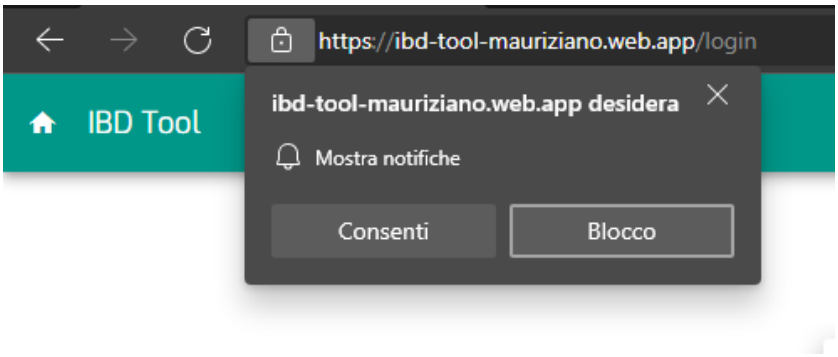


Figura 4.11. Pop-up per consentire la ricezione di notifiche

2. una volta registrato il consenso, ad ogni accesso alla pagina personale, il client inoltrerà al server una richiesta al server per sapere se esistono dei nuovi messaggi non letti associati a quell'utente. In caso positivo l'icona della chat presente nella *toolbar* si evidenzierà con un punto esclamativo (Fig. 4.12).

Figura 4.12. Icona che indica che la presenza messaggi non letti

3. quando l'utente visiterà la pagina dedicata alla messaggistica, come prima azione verrà generata una richiesta verso il server per ricevere tutti i messaggi scambiati fino al momento.

4. se il client ha dato il consenso alla ricezione di notifiche allora il client aspetterà l'arrivo di una notifica attraverso Firebase Cloud Messaging per richiedere un aggiornamento della chat, altrimenti si utilizzerà il polling come indicato nel primo punto. Si sarebbe potuto diminuire il numero di interazioni includendo direttamente nella notifica firebase il testo del nuovo messaggio in arrivo, però per aumentare il livello di sicurezza si è preferito chiedere al client di richiedere tramite una richiesta di tipo *GET* l'aggiornamento della chat, garantendo l'identità dell'utente attraverso il *JWT*.

All'interno del server viene sfruttata per la creazione della notifica la classe **Message** del package *com.google.firebase.messaging*. Il codice che permette la creazione di questo oggetto è mostrata di seguito:

```
Message message = Message.builder()
        .setToken(notificationRequestDto.getTarget())
        .setNotification(new Notification(notificationRequestDto.
        getTitle(), notificationRequestDto.getBody()))
        .putData("content", notificationRequestDto.getTitle())
        .putData("body", notificationRequestDto.getBody())
        .putData("type", topic)
        .build();
```

Vediamo singolarmente il significato di ogni metodo:

- **setToken**: permette di inserire il token associato al dispositivo del destinatario del messaggio. I token dei dispositivi associati a ciascun utente vengono salvati sul database e richiesti dall'applicazione nel momento dell'invio della notifica. Per ogni oggetto Message può essere associato un solo token, quindi in caso l'utente abbia più dispositivi associati, verranno generati più oggetti ognuno con un token diverso.

- **setNotification**: permette di inserire un oggetto di tipo Notification. Sul front-end questo oggetto verrà usato per creare la notifica pop. La logica del

programma permette ai medici di disattivare le notifiche pop direttamente dalle impostazioni personali. Quindi in tal caso, nella creazione dell'oggetto *Message*, questo metodo non verrebbe utilizzato.

- **setData**: permette di inserire qualsiasi tipo di dato in forma di *Chiave-Valore*. Tra virgolette vediamo le chiavi utilizzate e sono:
  - content: indica il titolo della notifica;
  - body: indica il contenuto della notifica;
  - type: indica il tipo di notifica. In IBD Tool abbiamo due tipi di notifiche: la prima per la ricezione degli *alert*, la seconda per il funzionamento della chat.

Dal punto di vista grafico si è creata un'interfaccia molto simile ai più comuni sistemi di messaggistica, cosicché gli utenti possano avere un'immediata confidenza con il sistema. La visualizzazione della chat risulta differente per gli utenti registrati come pazienti rispetto a quelli registrati come medici.

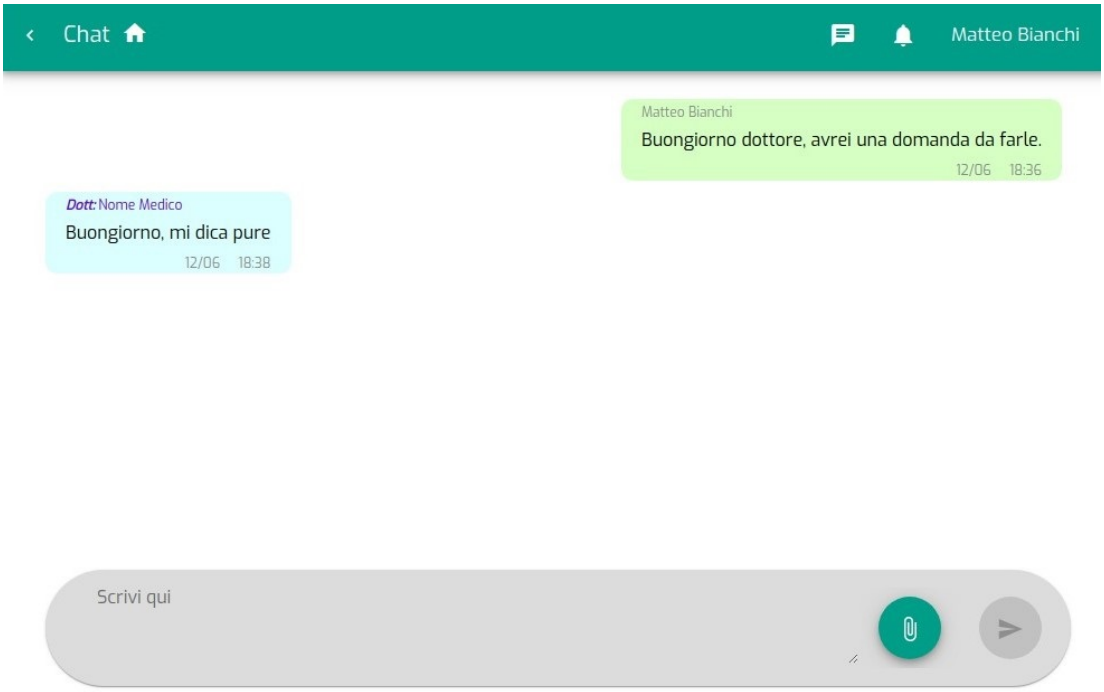


Figura 4.13. Interfaccia della chat per i pazienti

Il paziente ha una schermata più semplice rispetto a quella dei medici. Ogni paziente potrà scrivere un messaggio nella pagina rappresentata in Fig. 4.13, senza

doversi preoccupare di scegliere a quale medico inviarlo. Il messaggio arriverà a tutti i medici, i quali potranno collaborare per rispondere più velocemente possibile a tutte le richieste.

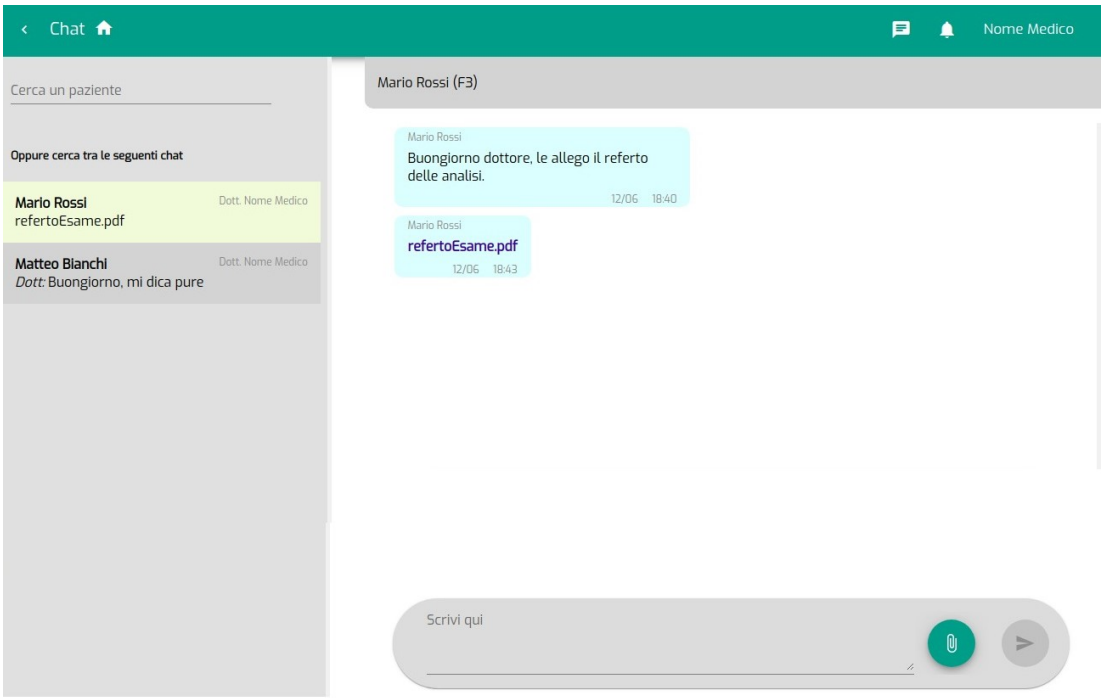


Figura 4.14. Interfaccia della chat per i medici

L'interfaccia della chat del medico invece è più complessa. La vista è divisa in due parti: a sinistra è presente una sezione in cui sono ordinate le ultime conversazioni, invece a destra è presente una sezione in cui viene mostrata la conversazione selezionata. Nella sezione di sinistra le conversazioni sono ordinate in modo da posizionare in alto le più recenti, e tra queste vengono segnate con una piccola icona sulla destra, tutte le conversazioni che non sono ancora state lette.

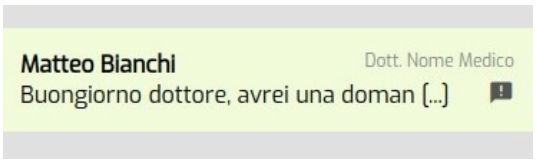


Figura 4.15. Esempio di messaggio non letto

Sempre in questa sezione è presente un campo che permette di ricercare il paziente da contattare utilizzando come parametri nome e cognome.

Nella sezione di destra inizialmente viene visualizzato un messaggio che chiede al medico di selezionare un paziente, in modo da poter iniziare la conversazione. Quando verrà selezionato il paziente, sempre in questa sezione, verranno riportati tutti i messaggi scambiati fino a quel momento. Inoltre, è stato reso possibile il collegamento diretto alla scheda del paziente selezionato. Per eseguire questa operazione basterà cliccare sul nome dell'interlocutore, presente nella parte alta della sezione.

Sia per i pazienti che per i medici è possibile utilizzare la chat per spedire dei file. Spesso questa funzionalità risulta molto utile per la comunicazione di referti. In Fig. 4.14 vediamo come il file viene visualizzato in grassetto, basterà quindi cliccare sul nome per scaricarlo sul proprio dispositivo.

La chat risulta molto comoda anche su dispositivi mobile. L'unica differenza rispetto a quella su desktop è presente nella finestra dei medici. Infatti si è scelto di dividere l'elenco degli ultimi messaggi ricevuti e la sezione contenente la conversazione in due pagine separate.

Capitolo 5

Conclusioni e lavori futuri

IBD Tool è un progetto in continua evoluzione. Nonostante abbia un'ampia varietà di funzionalità, può essere ancora ampliato con l'introduzione di nuovi strumenti a supporto di medici e pazienti. Sono state infatti definite, come proseguimento del lavoro di questa tesi, una serie di nuove possibili funzionalità da integrare nell'applicazione. Tra queste si hanno:

- una finestra che descrive, attraverso dei grafici, lo stato dell'applicazione (ad esempio numero di utenti iscritti o percentuale di utenti attivi) e fornisce uno specchietto sull'andamento generale della malattia dei pazienti (ad esempio quanti pazienti sono in remissione o recidivi);

- una funzionalità per l'estrazione dei dati ricavati dai questionari, in modo da eseguirne studi di concordanza tra i questionari compilati dai pazienti e quelli dei medici;

- la modifica della scheda paziente con l'ampliamento dei dati clinici inseribili dai medici;

- una bacheca avvisi per i pazienti in modo da informare in maniera uniforme tutti gli utenti con una singola operazione.

La piattaforma in questo primo anno ha fornito ai clinici un mezzo versatile per il monitoraggio dei pazienti con malattie croniche intestinali. Anche alla luce dell'attuale situazione sanitaria, è sempre più evidente che al giorno d'oggi il sistema sanitario non può prescindere dalla telemedicina. I pazienti iscritti all'applicazione hanno risposto in maniera molto positiva, applicando un impegno costante nella compilazione dei questionari. Tra i pazienti iscritti all'applicazione circa il 90% risultano attivi. Questo è un segnale di quanto l'applicazione di queste tecnologie, che permettono di fornire servizi sanitari a distanza, siano visti con fiducia sia dal personale sanitario che dai pazienti.

La telemedicina difatti, contribuisce a migliorare l'efficacia della terapia e la prevenzione della malattia, mettendo a disposizione gli strumenti più innovativi e versatili che la tecnologia odierna fornisce nel campo medico-sanitario.

Acronimi

Di seguito gli acronimi usati nel documento:

IBD - Inflammatory Bowel Diseases
FE - Front end
BE - Back end
AOP - Aspect Oriented Programming
IoC - Inversion of Control
ICT - Informationand Communication Technologie
JWT - JSON Web Token
HBI - Harvey Bradshaw Index
MIAH-CD - Monitoring IBD at Home Crohn's Disease
Patient-SCCAI - Simple Clinical Colitis Activity Index
MIAH-UC - Monitoring IBD at Home Ulcerative Colitis
EQ5D5L - European Quality, version 5D, 5 levels
IPAQ-SF - International Physical Activity Questionnaire Short Form
MMAS-8 - Medication Morinsky Adherence Scale
PRISM - Pictorial Representation of Illness and Self Measure
PSQI - Pittsburg Sleep Quality Index
TSQM - Treatment Satisfaction Questionnaire for Medication
WPAI - Work Productivity and Activity Impairment Questionnaire
HADS - Hospital Anxiety and Depression Scale
PHQ-9 - Patient Health Questionnaire 9

Bibliografia

[1] "Telemedicina, linee di indirizzo nazionale." https://www.salute.gov.it/imgs/C_17_pubblicazioni_2129_allegato.pdf.

[2] *AMICI Onlus. Il burden economico delle MICI in Italia.*, Visitato il: 18/05/2021. https://amiciitalia.eu/sites/default/files/Burden_economico_mici_mail.pdf.

[3] C. B. G. M. A. A. Giulio Perrone, Stefania Zerbo, "Telemedicine during covid-19 pandemic: Advantage or critical issue?," *Medico-Legal Journal*, vol. 88, pp. 76–77, 2020.

[4] Protezione civile, Visitato il: 02/05/2021. http://www.protezionecivile.it/.

[5] K. L. Rockwell and A. S. Gilroy, "Incorporating telemedicine as part of COVID-19 outbreak response systems," *Am J Manag Care*, vol. 26, pp. 147–148, 04 2020.

[6] D. Jakhar, S. Kaul, and I. Kaur, "WhatsApp messenger as a teledermatology tool during coronavirus disease (COVID-19): from bedside to phone-side," *Clin Exp Dermatol*, vol. 45, pp. 739–740, Aug 2020.

[7] F. Pecoraro, F. Clemente, and D. Luzi, "The efficiency in the ordinary hospital bed management in Italy: An in-depth analysis of intensive care unit in the areas affected by COVID-19 before the outbreak," *PLoS One*, vol. 15, no. 9, 2020.

[8] D. L. Fabrizio Pecoraro, Fabrizio Clemente, "Number of hospital beds, hospitalization and length of stay in the years 2010 and 2017 highlighting the differences in each italia region.." URL: https://journals.plos.org/plosone/article/figure?id=10.1371/journal.pone.0239249.t001.

[9] D. L. Fabrizio Pecoraro, Fabrizio Clemente, "Number of hospital beds, hospitalization and length of stay in the years 2010 and 2017 in the intensive care units highlighting the differences in each italian region.." URL:

https://journals.plos.org/plosone/article/figure?id=10.1371/journal.pone.0239249.t001.

[10] D. L. Fabrizio Pecoraro, Fabrizio Clemente, "Overall classification of each italian region considering both the hospital bed management indicators and the complexity and performance indicators.," *PLOS ONE*. URL: https://journals.plos.org/plosone/article/figure?id=10.1371/journal.pone.0239249.t001.

[11] J. K. A. S. F. C. Michael D Kappelman 1, Kristen R Moore, "Recent trends in the prevalence of crohn's disease and ulcerative colitis in a commercially insured us population," *Digestive Diseases and Sciences*, 2013.

[12] P. M. T. O. M. M. Karen A. Chachu, MD, "How to diagnose and treat ibd mimics in the refractory ibd patient who does not have ibd," *Inflammatory Bowel Diseases*, vol. 22, no. 5, p. 1262–1274, 2016.

[13] M. D. M. Seyed Saeid Seyedian, Forogh Nokhostin, "A review of the diagnosis, prevention, and treatment methods of inflammatory bowel disease," *J Med Life*, vol. 12, no. 2, pp. 113–122, 2019.

[14] *AMICI Onlus. La Malattia di Crohn.*, Visitato il: 18/05/2021. https://amiciitalia.eu/categorie/la-malattia-di-crohn.

[15] *AMICI Onlus. La Colite Ulcerosa.*, Visitato il: 18/05/2021. https://amiciitalia.eu/categorie/la-colite-ulcerosa.

[16] P. Collins and J. Rhodes, "Ulcerative colitis: diagnosis and management," vol. 333, no. 7563, pp. 340–343, 2006.

[17] D. C. Baumgart, "The diagnosis and treatment of crohn's disease and ulcerative colitis," *Dtsch Arztebl International*, vol. 106, no. 8, pp. 123–133, 2009.

[18] *Scegliere tra app Web tradizionali e a pagina singola*, Visitato il: 19/05/2021. https://docs.microsoft.com/it-it/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps.

[19] *Introduction to Angular concepts*, Visitato il: 19/05/2021. https://angular.io/guide/architecture.

[20] *What is Angular?*, Visitato il: 19/05/2021. https://angular.io/guide/what-is-angular.

[21] *Spring, Microservices*, Visitato il: 19/05/2021. https://spring.io/microservices.

[22] *Introduction to Spring AOP*, Visitato il: 20/05/2021. https://www.baeldung.com/spring-aop.