POLITECNICO DI TORINO
Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Progettazione e sviluppo di un sistema di raccomandazione per la fruizione di servizi sanitari territoriali mediante tecniche di Data Mining

Relatori:
Prof.ssa Silvia Chiusano
Dott.ssa Elena Daraio

Candidato:
Davide Pani

Anno accademico 2020-2021

Sommario

Negli ultimi anni la quantità considerevole di dati clinici raccolta quotidianamente ha sollevato il bisogno di utilizzare in ambito sanitario sistemi in grado di analizzare, visualizzare ed impiegare i dati raccolti per supportare sia i pazienti sia i professionisti del settore nel prendere decisioni più efficienti ed accurate per quanto riguarda la salute.

Il presente lavoro di tesi si inserisce nel contesto dei servizi sanitari territoriali, i quali sono di fondamentale importanza per garantire ai pazienti fragili, con bisogni assistenziali complessi o in condizioni di non autosufficienza parziale o totale, di proseguire il processo di cura iniziato all'interno dell'ospedale. Il momento delle dimissioni dall'ospedale rappresenta infatti un passaggio critico per questa categoria di pazienti, in quanto sancisce il passaggio dalla fase acuta della patologia alla fase di cure riabilitative, necessarie per un recupero ottimale delle funzioni lese. Le strutture assistenziali post-degenza offrono una soluzione in grado di soddisfare i bisogni di natura sanitaria e sociosanitaria di tali pazienti. Il processo di individuazione della struttura assistenziale maggiormente adatta alle necessità del paziente risulta però spesso complicata sia per gli operatori sanitari che per il paziente ed i suoi familiari.

Il lavoro di tesi si è inserito all'interno dello sviluppo di un'applicazione per l'esplorazione di strutture assistenziali post-degenza regionali ed ha avuto come obiettivo lo sviluppo di soluzioni per supportare gli utenti (operatori sanitari, pazienti e familiari dei pazienti) nel processo di ricerca di una struttura assistenziale. Le strutture RSA sono le strutture considerate come esempio di studio nella tesi.

In particolare, il lavoro di tesi si è concentrato sullo sviluppo di soluzioni per la memorizzazione, per l'analisi e per l'impiego dei dati delle ricerche utenti per tali strutture con un duplice obiettivo: (i) analizzare e visualizzare le caratteristiche delle ricerche utenti e (ii) utilizzare tali informazioni per supportare l'utente nel processo di selezione di una struttura assistenziale. Le soluzioni proposte consistono (i) in un cruscotto informativo per la visualizzazione di statistiche riguardanti le ricerche per strutture RSA effettuate dagli utenti finali e (ii) in un sistema di raccomandazione per coadiuvare l'utente, durante la fase di ricerca, verso la struttura assistenziale maggiormente adatta alle esigenze sanitarie indicate.

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

Negli ultimi anni è stata raccolta una quantità considerevole di dati clinici circa lo stato di salute dei pazienti. Queste informazioni sono però distribuite tra molteplici siti, rendendo difficile per gli utenti trovare informazioni utili per il miglioramento della propria salute. Inoltre, nuove informazioni sulle terapie, nuovi test e nuovi farmaci sono resi disponibili al personale sanitario quotidianamente, rendendo difficile per gli stessi prendere le decisioni più appropriate nel trattamento dei pazienti.

Queste problematiche hanno sollevato il bisogno di utilizzare in ambito sanitario sistemi in grado di analizzare, visualizzare ed impiegare i dati raccolti per supportare sia gli utenti finali sia i professionisti sanitari nel prendere decisioni più efficienti ed accurate per quanto riguarda la salute [24]. I sistemi in grado di analizzare e visualizzare le caratteristiche dei dati raccolti sono i cruscotti informativi, mentre i sistemi in grado impiegare tali dati per guidare l'utente nel prendere decisioni efficaci sono i sistemi di raccomandazione.

I cruscotti informativi sono strumenti in grado di analizzare, riassumere e visualizzare attraverso un insieme di elementi grafici le informazioni rilevanti di un'organizzazione, ad esempio di un'azienda, per guidarne e supportarne i processi decisionali [20]. Negli ultimi anni molte organizzazioni sanitarie stanno adottando questi sistemi per poter misurare e valutare la qualità dei servizi sanitari offerti e per migliorare l'efficienza nelle cure [21].

I sistemi di raccomandazioni sono invece algoritmi che hanno come obiettivo quello di identificare e suggerire un insieme di contenuti, prodotti o servizi (e.g. libri, film, ristoranti, hotel) che siano il più possibile in accordo con le preferenze o le necessità di un utente [28]. Da diversi anni i sistemi di raccomandazione oltre a giocare un ruolo centrale in ambito *e-commerce* sono sempre più impiegati anche in ambito sanitario [24]. In tale contesto i sistemi di raccomandazione sono denominati *Health Recommender Systems*.

Gli *Health Recommender Systems* hanno il compito di analizzare lo stato di salute del paziente e, a seconda del caso, di fornire raccomandazioni sui servizi medicali più appropriati assicurando al tempo stesso accuratezza, affidabilità e privacy nei confronti delle informazioni del paziente. Rispetto ai sistemi di raccomandazione tradizionali gli *Health Recommender Systems* devono riconsiderare qualsiasi approccio basato esclusivamente sulle preferenze degli utenti, poiché ciò che può essere adatto per un utente può non esserlo o essere pericoloso per un altro. Ad esempio, se dei farmaci diuretici sono adatti per chi soffre di ipertensione, gli stessi possono essere pericolosi per chi è affetto da diabete. Anche nel caso di pazienti affetti dalla stessa patologia, non sempre i rimedi adatti per uno possono

essere consigliati agli altri poiché i vari pazienti potrebbero trovarsi in condizioni di salute anche molto diverse tra loro.

In base all'approccio utilizzato i sistemi di raccomandazione si suddividono principalmente nelle seguenti categorie [22]:

- **Collaborative Filtering**: nei sistemi di raccomandazione tradizionali l'idea alla base di questa tecnica è quella di raccomandare ad un utente gli articoli apprezzati dagli utenti simili a lui. In ambito sanitario questo approccio può essere interpretato nel seguente modo: *"ai pazienti che condividono condizioni di salute simili possono essere raccomandati trattamenti medicali simili"*.

- **Content-Based**: tradizionalmente questa tecnica raccomanda ad un utente gli articoli simili agli articoli che sono stati apprezzati da lui in passato. In ambito sanitario questo approccio può essere utilizzato per raccomandare al paziente servizi sanitari simili ad altri servizi che hanno giovato alle sue condizioni di salute nel passato. In tale ambito questo approccio deve comunque sempre tenere in considerazione le condizioni di salute attuali del paziente.

- **Knowledge-Based**: questo approccio è generalmente utilizzato nei casi in cui non sono note le preferenze sugli articoli o nei casi in cui è necessario che l'utente definisca dei vincoli espliciti sui prodotti da raccomandare. In ambito sanitario questo approccio potrebbe essere utilizzato, ad esempio, in un sistema di raccomandazione per la dieta per consentire all'utente di specificare gli alimenti di cui è allergico.

- **Association Rules-Based**: questo approccio si basa sulla tecnica di Data Mining delle *regole di associazione*, la quale consente di identificare relazioni nascoste tra gli articoli acquistati dagli utenti. In ambito sanitario tale tecnica può essere utilizzata per identificare eventuali relazioni tra le condizioni di salute dei pazienti e i servizi medicali di cui necessitano.

In letteratura gli scenari in cui i sistemi di raccomandazione sono stati utilizzati in ambito sanitario sono molteplici. Nella ricerca condotta da Tran *et al.* [24] sullo stato dell'arte degli *Health Recommender Systems*, sono stati rilevati i seguenti scenari di applicazione: raccomandazioni per la dieta, raccomandazioni di farmaci, raccomandazioni per l'attività motoria e raccomandazioni di medici e strutture sanitarie.

Raccomandazioni per la dieta A causa dell'aumento della varietà di cibo disponibile e di uno stile di vita sempre più frenetico, molte persone hanno difficoltà a fare scelte alimentari salutari. In questo contesto, i sistemi di raccomandazioni possono motivare gli utenti nel modificare le proprie abitudini alimentari verso scelte più salutari. Tra i sistemi proposti in letteratura ci sono sistemi per la raccomandazioni di diete, sistemi per la raccomandazione di alimenti per la prevenzione di determinate patologie e sistemi per la raccomandazione di alimenti sostitutivi.

Achananuparp *et al.* [25], ad esempio, hanno realizzato un sistema di raccomandazione per consigliare agli utenti dei possibili sostituti alimentari più salutari nei pasti selezionati dagli stessi. Nel loro approccio, gli alimenti sostitutivi sono ricercati in base alla similarità dei contesti in cui solitamente sono consumati. Gli alimenti ed

i corrispondenti contesti sono stati rappresentati in una matrice in cui le righe rappresentano gli alimenti mentre le colonne rappresentano i contesti. Ogni cella della matrice rappresenta l'associazione tra un alimento ed un contesto, indicata da un indice noto come *Positive Pointwise Mutual Information (PPMI)*. Il problema maggiore che hanno dovuto affrontare con questo tipo di matrice è dovuto alla notevole sparsità della stessa, che avrebbe avuto un impatto negativo nel calcolo delle similarità. Per risolvere questo problema hanno applicato un algoritmo di riduzione della dimensionalità noto come *Singular Value Decomposition (SVD)*. Nello scegliere i sostituti alimentari più adatti, il sistema calcola la *similarità del coseno* tra l'alimento in oggetto e tutte le righe della matrice e seleziona la classifica dei dieci alimenti più simili da raccomandare.

Raccomandazioni di farmaci La prescrizione di medicinali errati è uno degli errori più gravi che un medico può commettere e che può mettere in pericolo la vita del paziente. Le ragioni di questo fenomeno possono essere dovute all'inesperienza del medico e all'aumento delle informazioni disponibili sui farmaci, rendendo difficile la scelta dei più adeguati. In questo contesto, sono stati sviluppati sistemi di raccomandazione per il supporto degli utenti finali e dei professionisti sanitari nell'identificare i medicinali più adatti per specifiche patologie.

Bankhele *et al.* [26], ad esempio, hanno proposto un sistema di raccomandazione *collaborative filtering* con approccio *user-based* per suggerire a pazienti affetti da diabete una lista di medicinali utilizzati nel trattamento di tale patologia. Il sistema prevede inizialmente una fase di registrazione dell'utente in cui viene richiesto di inserire una serie di parametri di salute (e.g. livello di insulina, pressione sanguigna, indice di massa corporea, età, livello di glucosio nel sangue). Successivamente viene calcolata la similarità tra l'utente attivo e i pazienti memorizzati nel database sulla base dei parametri di salute utilizzando il *coefficiente di correlazione di Pearson*. Una volta identificato il paziente più simile, il sistema procede con il suggerire all'utente attivo la lista di medicinali utilizzati nel trattamento di tale paziente.

Raccomandazioni per l'attività motoria Oltre al fornire raccomandazioni per il trattamento di patologie, gli *Health Recommender Systems* si stanno anche focalizzando sul suggerimento di attività fisiche da svolgere, con l'obiettivo di ridurre il rischio di diventare pazienti fragili e di prevenire l'aggravarsi delle condizioni di salute.

Kulev *et al.* [27], ad esempio, hanno proposto un sistema di raccomandazione di tipo collaborativo per suggerire agli utenti quali attività fisiche svolgere per migliorare la propria condizione di salute. Il sistema utilizza uno storico dei dati riguardanti i parametri di salute degli utenti (i.e. peso corporeo, pressione sanguigna, battito cardiaco, livello di glucosio nel sangue) in combinazione con le attività fisica svolte dagli stessi. L'ipotesi su cui si basa tale sistema è che attività fisiche che hanno migliorato i parametri di salute per un certo utente possono migliorare gli stessi parametri di un altro utente, a condizione che i due utenti siano simili. I vari utenti sono quindi categorizzati e confrontati sulla base dei loro parametri medici e non in base alle valutazioni. Per consigliare quali attività fisiche svolgere, il sistema ricerca quindi gli utenti più simili all'utente attivo e raccomanda allo stesso le attività che hanno giovato agli utenti simili.

Raccomandazioni di medici e strutture sanitarie Negli ultimi anni, a causa dell'aumento della quantità di informazioni mediche disponibili, è diventato sempre più difficile per i pazienti trovare i migliori medici e le migliori strutture per il trattamento dei corrispettivi problemi di salute. In questo contesto, i sistemi di raccomandazione possono fornire supporto nella ricerca e nella scelta dell'opzione più adeguata.

Narducci *et al.* [28], ad esempio, hanno proposto l'utilizzo di un sistema di raccomandazione all'interno di un social network in ambito sanitario per il suggerimento di medici e strutture sanitarie. Il sistema suggerisce all'utente attivo pazienti simili alle condizioni specificate e fornisce supporto nella scelta di medici e strutture sanitarie. L'algoritmo di raccomandazione prima calcola la similarità tra i pazienti e il profilo utente per poi generare una classifica di dottori e strutture sanitarie per quel determinato profilo sfruttando i dati sanitari condivisi dalla comunità. Nel calcolo della similarità tra due pazienti sono tenuti in considerazione le condizioni di salute, i trattamenti a cui sono stati sottoposti e i sintomi, i quali sono combinati linearmente. La similarità tra pazienti è poi utilizzata per assegnare un punteggio a medici e strutture sanitarie con cui generare la lista dei migliori da raccomandare. Per quanto riguarda il punteggio assegnato ai medici, il sistema utilizza un approccio simile al *collaborative filtering* calcolando una media pesata delle valutazioni assegnate dai pazienti al medico, dove i pesi corrispondono ai valori di similarità tra i pazienti e l'utente attivo. Il punteggio delle strutture sanitarie è calcolato con lo stesso metodo utilizzato per i medici a cui viene combinato linearmente un indicatore di qualità assegnato dal Ministero della Salute italiano.

Han *et al.* [29] hanno proposto invece un sistema di raccomandazione ibrido, combinando un approccio *collaborative filtering* con un approccio *content-based*, per la raccomandazione di medici di base ai pazienti. Nel sistema proposto da loro, non disponendo di valutazioni esplicite dei pazienti nei confronti dei corrispettivi medici, hanno utilizzato due tipologie di informazioni. Per quanto riguarda la componente *content-based* hanno utilizzato le caratteristiche di medici e pazienti (e.g. l'età, il genere, il paese di origine) per creare associazioni tra persone simili, nell'ipotesi che un paziente sia più portato ad apprezzare un medico con caratteristiche simili. Per quanto riguarda la componente collaborativa hanno invece utilizzato i feedback impliciti dei pazienti sulla base delle visite ripetute verso lo stesso medico, nell'ipotesi che il rivolgersi più volte allo stesso medico sia segnale di fiducia ed apprezzamento verso il medesimo. Combinando queste informazioni, sono stati quindi in grado di generare raccomandazioni personalizzate ai pazienti di medici con cui potrebbero probabilmente instaurare un rapporto di fiducia.

1.1 Il lavoro di tesi

Il presente lavoro di tesi si inserisce nel contesto dei servizi sanitari territoriali, i quali sono di fondamentale importanza per garantire ai pazienti fragili, con bisogni assistenziali complessi o in condizioni di non autosufficienza parziale o totale, di proseguire il processo di cura iniziato all'interno dell'ospedale. Il momento delle dimissioni dall'ospedale rappresenta infatti un passaggio critico per questa categoria di pazienti, in quanto sancisce il passaggio dalla fase acuta della patologia alla fase di cure riabilitative, necessarie per un recupero

ottimale delle funzioni lese. Le strutture assistenziali post-degenza offrono una soluzione in grado di soddisfare i bisogni di natura sanitaria e sociosanitaria di tali pazienti. Il processo di individuazione della struttura assistenziale maggiormente adatta alle necessità del paziente risulta però spesso complicata sia per gli operatori sanitari che per il paziente ed i suoi familiari.

Il lavoro di tesi si è inserito all'interno dello sviluppo di un'applicazione per l'esplorazione di strutture assistenziali post-degenza regionali ed ha avuto come obiettivo lo sviluppo di soluzioni per supportare gli utenti (operatori sanitari, pazienti e familiari dei pazienti) nel processo di ricerca di una struttura assistenziale. Tale processo è schematizzato in Figura 1.1.

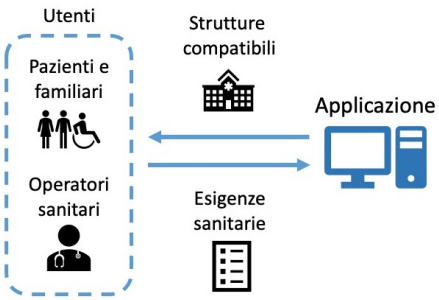


Figura 1.1: Processo di ricerca di una struttura assistenziale attraverso l'applicazione.

Le strutture RSA sono le strutture considerate come esempio di studio nella tesi. In particolare, il lavoro di tesi si è concentrato sullo sviluppo di soluzioni per la memorizzazione, per l'analisi e per l'impiego dei dati delle ricerche utenti per tali strutture con un duplice obiettivo: (i) analizzare e visualizzare le caratteristiche delle ricerche utenti e (ii) utilizzare tali informazioni per supportare l'utente nel processo di selezione di una struttura assistenziale. Le soluzioni proposte consistono (i) in un cruscotto informativo per la visualizzazione di statistiche riguardanti le ricerche per strutture RSA effettuate dagli utenti finali e (ii) in un sistema di raccomandazione per coadiuvare l'utente, durante la fase di ricerca, verso la struttura assistenziale maggiormente adatta alle esigenze sanitarie indicate. Le soluzioni proposte consentono inoltre ai gestori della sanità di usufruire dell'applicazione, i quali attraverso di essa possono analizzare le esigenze dei pazienti e valutare di conseguenza miglioramenti sulla qualità e sulla disponibilità dei servizi offerti. Le soluzioni proposte nel lavoro di tesi sono schematizzate in Figura 1.2.

La tesi è stata quindi articolata in due attività principali.

Nella prima parte del lavoro di tesi è stato definito il modello per la rappresentazione dei dati delle ricerche utenti nella base di dati più adatta al contesto considerato. Dopo uno studio dei modelli di basi di dati esistenti, è stato selezionato un modello NoSQL per la rappresentazione dei dati ed è stato selezionato MongoDB come base di dati di riferimento. È stato quindi progettato e realizzato un cruscotto per analizzare le caratteristiche delle ricerche effettuate dagli utenti finali. Il cruscotto è composto da un insieme di grafici che consentono di visualizzare statistiche riguardanti la distribuzione dei valori dei parametri di ricerca selezionati dagli utenti nelle ricerche passate. Il cruscotto prevede inoltre due modalità di filtraggio dei dati per consentire la visualizzazione di statistiche sulla base delle specifiche esigenze sanitarie espresse dagli utenti.

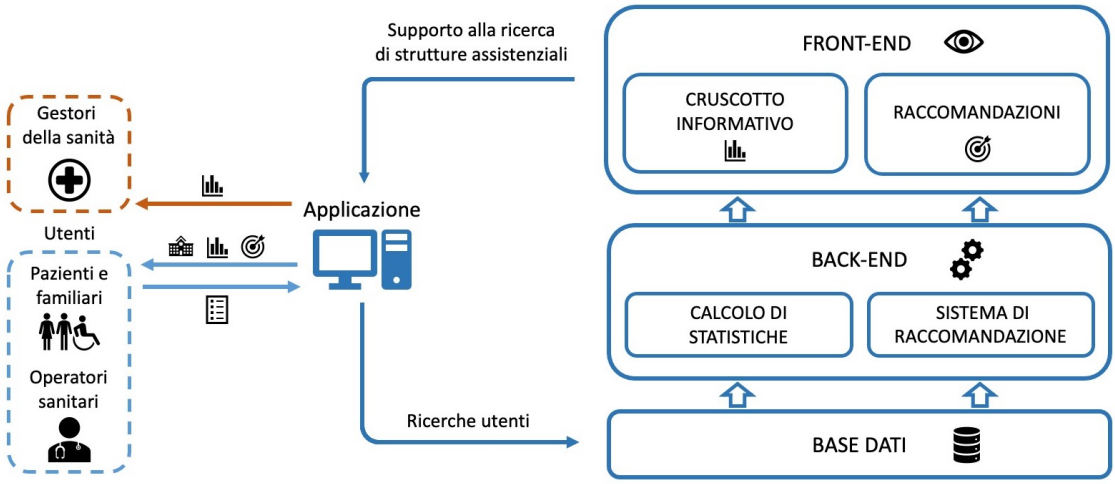


Figura 1.2: Soluzioni proposte per supportare gli utenti nel processo di ricerca di una struttura assistenziale.

Nella seconda parte della tesi è stata valutata la realizzazione di un sistema di raccomandazione per suggerire all'utente finale, durante la fase di ricerca, la struttura sanitaria più adatta alle sue esigenze. Sono state studiate le principali tipologie di sistemi di raccomandazione esistenti ed è stata valutata la loro applicabilità al contesto di analisi. Tali valutazioni hanno mostrato che, con i dati a disposizione, una raccomandazione diretta di una struttura RSA non sarebbe stata possibile; è stato quindi proposto un sistema di raccomandazione per coadiuvare l'utente, durante la fase di ricerca stessa, nella selezione dei parametri di ricerca più appropriati da utilizzare. La soluzione proposta consiste in un sistema di raccomandazione ibrido composto da due componenti basate sugli approcci *collaborative filtering* e *association rules-based*.

La componente basata sul *collaborative filtering* riceve in ingresso la configurazione dei parametri di ricerca base selezionati dall'utente nella pagina di ricerca e produce in uscita una configurazione iniziale dei parametri di ricerca avanzata consigliati. In particolare, tale componente è basata sull'approccio *neighborhood-based* ed utilizza l'algoritmo *k-nearest neighbors* per selezionare, tra le ricerche memorizzate, le ricerche più simili a quella dell'utente attivo basandosi sulla configurazione dei parametri di ricerca base. Per consentire il calcolo della similarità tra le ricerche, i parametri di ricerca base sono prima codificati tramite *one-hot encoding* e successivamente confrontati utilizzando una versione pesata della metrica *similarità del coseno*. Le raccomandazioni generate sono poi composte da una media pesata dei valori di rilevanza espressi all'interno delle ricerche *neighbors* e da una classifica pesata dei servizi maggiormente richiesti all'interno delle stesse.

La componente *association rules-based* si pone come un'estensione progressiva delle raccomandazioni di servizi fornite inizialmente dalla componente collaborativa. Questa componente riceve in ingresso l'insieme dei servizi inseriti dall'utente nella pagina di ricerca congiuntamente alla configurazione dei parametri di ricerca base selezionata e produce in uscita la lista dei servizi richiesti spesso congiuntamente a quelli inseriti dall'utente. In particolare, la generazione delle raccomandazioni è basata sulle informazioni derivanti

dall'estrazione di regole di associazione, le quali sono estratte a partire dai servizi richiesti congiuntamente nelle ricerche utenti passate. Per consentire tale operazione la componente prevede una prima fase di preprocessing del dataset per la trasformazione dello stesso in formato transazionale. Inoltre, per garantire una maggiore affinità dei servizi raccomandati con le esigenze di servizi medicali richiesti dall'utente, l'estrazione delle regole avviene esclusivamente a partire dalle ricerche memorizzate aventi la stessa configurazione di ricerca base selezionata dall'utente. L'estrazione delle regole di associazione è poi effettuata in tempo reale utilizzando l'algoritmo FP-growth con una configurazione predefinita dei valori di supporto e confidenza. Le raccomandazioni generate sono poi composte dai servizi presenti nel conseguente delle regole estratte, ordinati in ordine decrescente in base al valore di confidenza.

Di seguito vine introdotto il contenuto dei sette capitoli successivi che compongono il presente elaborato di tesi. Nel Capitolo 2 viene descritta l'applicazione di riferimento su cui è stato svolto il lavoro di tesi; ne vengono introdotte brevemente le sezioni che la compongono e ne viene descritta l'architettura. Nel Capitolo 3 sono esaminate e confrontate le categorie principali di basi di dati esistenti, con il fine di identificare la categoria più adatta al contesto in analisi per la memorizzazione dei dati delle ricerche utenti per strutture RSA. Identificata tale categoria vengono quindi descritte le caratteristiche della base dati di riferimento selezionata, i.e. MongoDB. Nel Capitolo 4 sono analizzati i dati delle ricerche utenti da memorizzare e viene descritto il modello dei dati progettato per la rappresentazione di tali dati all'interno della base dati. Nel Capitolo 5 viene descritto il cruscotto proposto per l'analisi delle caratteristiche delle ricerche utenti e ne viene descritto lo sviluppo e l'implementazione all'interno dell'applicazione. Nel Capitolo 6 sono introdotti i sistemi di raccomandazione e ne vengono descritte le categorie principali. Nel Capitolo 7 sono riportate le valutazioni di applicabilità al contesto delle categorie principali di sistemi di raccomandazione e viene presentata la soluzione proposta. Viene quindi descritta nel dettaglio la fase di progettazione e la fase di sviluppo e di implementazione all'interno dell'applicazione del sistema di raccomandazione proposto. Nel Capitolo 8 sono infine presentate le conclusioni e gli sviluppi futuri del lavoro realizzato.

Capitolo 2

Descrizione dell'applicazione

In questo capitolo viene presentata l'applicazione sulla quale è stato svolto il lavoro di tesi.

Il capitolo è suddiviso in tre parti:

- nella prima parte vengono introdotte le sezioni dell'applicativo dedicate all'esplorazione e alla ricerca di strutture RSA;
- nella seconda parte viene descritta l'architettura iniziale dell'applicazione, precedentemente il lavoro di tesi;
- nella terza parte viene descritta l'architettura estesa dell'applicazione, a seguito del lavoro di tesi.

2.1 Introduzione alle sezioni

L'applicazione in oggetto è un applicativo web che consente l'esplorazione e la scelta di strutture assistenziali post-degenza presenti sul territorio piemontese. La parte dell'applicativo dedicata all'esplorazione delle strutture RSA su cui si è concentrato il lavoro di tesi è composta da tre sezioni: la pagina *Strutture*, la pagina *Indicatori* e la pagina *Ricerca*. Le tre sezioni sono descritte di seguito.

Pagina Strutture

Nella pagina *Strutture* è possibile visualizzare attraverso una mappa la posizione delle strutture RSA presenti nella regione Piemonte. Interagendo con tale mappa l'utente può visualizzare nel dettaglio la zona di interesse.

Pagina Indicatori

Nella pagina *Indicatori* è possibile esplorare alcuni indicatori esemplificativi delle caratteristiche delle strutture, al fine di conoscere il quadro generale della regione Piemonte e di confrontare diverse strutture rispetto ad aspetti di interesse, quali ad esempio il numero di posti letto e il tempo di attesa medio.

Pagina Ricerca

Nella pagina *Ricerca*, mostrata in Figura 2.1, è possibile cercare le strutture compatibili con particolari esigenze oggettive e soggettive dell'utente.

Figura 2.1: Pagina *Ricerca*.

In questa pagina l'utente può scegliere tra due modalità di ricerca:

- **Ricerca Base**: modalità con la quale è possibile ottenere un elenco di strutture compatibili con le esigenze oggettive dell'utente, quali ad esempio le prestazioni mediche necessarie.
- **Ricerca Ottimizzata**: modalità con la quale è possibile ottenere una classifica delle strutture individuate tramite la *Ricerca Base* secondo diversi criteri di preferenza soggettivi. La classifica delle strutture proposte è realizzata mediante un sistema di raccomandazione di tipo *knowledge-based*.

Alla pressione del pulsante *Cerca*, se vengono trovate strutture compatibili con i parametri di ricerca selezionati, viene infine visualizzato l'elenco delle strutture trovate.

2.2 Architettura iniziale

L'applicazione in oggetto è un applicativo web di tipo *MPA (Multi Page Application)* basato su un'architettura a microservizi. In particolare, l'architettura è stata realizzata tramite l'utilizzo di Docker. Una rappresentazione dell'architettura iniziale è mostrata in Figura 2.2.

L'architettura iniziale è composta da tre *container*: uno ospitante il server web dell'applicazione e due dedicati ai servizi. Il server web è un server *Apache* in ascolto sulla porta 80. I due servizi sono invece sviluppati in Python mediante il framework Flask. I servizi sono indipendenti l'uno dall'altro e sono ciascuno in ascolto su una porta specifica. La comunicazione con essi avviene mediante il protocollo HTTP: i parametri in ingresso al servizio sono forniti come parametri della richiesta GET mentre la risposta del servizio è codificata in formato JSON.

Per quanto riguarda la memorizzazione dei dati, inizialmente l'applicazione non prevedeva l'utilizzo di un DBMS; i dati visualizzati nelle diverse pagine venivano reperiti da file CSV memorizzati localmente nel server.

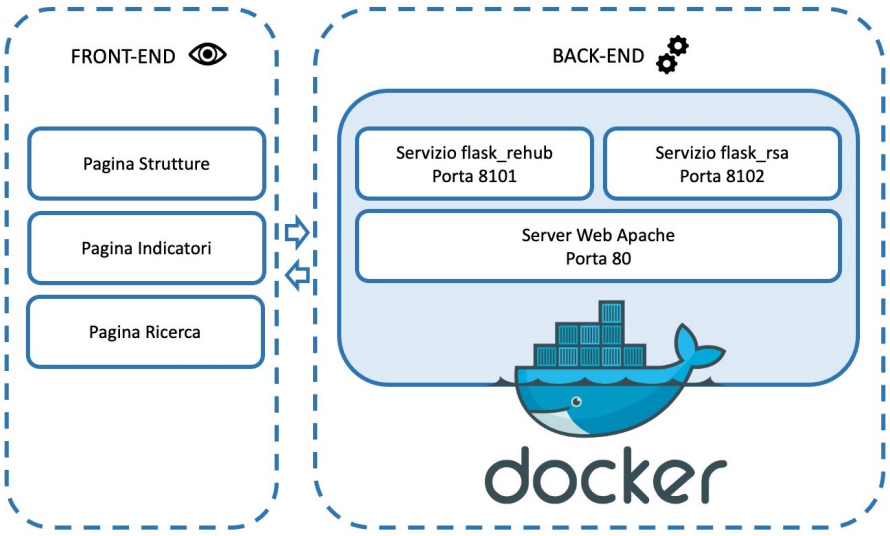


Figura 2.2: Architettura dell'applicazione iniziale.

2.2.1 Tecnologie utilizzate

Di seguito sono descritte brevemente le tecnologie utilizzate nell'architettura iniziale dell'applicazione, i.e. Docker e Flask.

Docker

Docker [1] è una piattaforma software che permette la creazione e l'utilizzo di *container* Linux. La tecnologia Docker si appoggia al kernel di Linux e alle sue funzionalità per eseguire molteplici *container* sulla stessa stessa macchina e per segregarne i processi in modo che possano essere eseguiti in modo indipendente l'uno dall'altro. I vantaggi principali offerti dai *container* Docker sono l'autosufficienza e l'indipendenza. L'autosufficienza è dovuta al fatto che ciascun *container* contiene al suo interno tutte le dipendenze, le librerie e le configurazioni necessarie all'esecuzione dell'applicazione, rendendolo anche facilmente portabile verso altre macchine. La leggerezza è dovuta al fatto che i *container* in esecuzione sulla stessa macchina condividono lo stesso kernel del sistema operativo ospitante, per il quale non è quindi necessaria la virtualizzazione come avviene nel caso delle macchine virtuali. Questo permette quindi di ottimizzare le risorse dell'infrastruttura e di poter eseguire un gran numero di *container* sulla stessa macchina fisica.

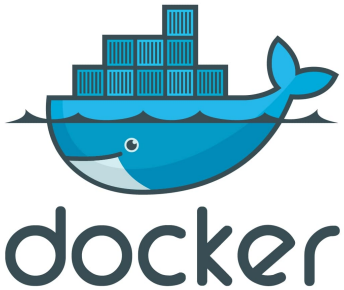


Figura 2.3: Logo di Docker. [1]

Flask

Flask [2] è un framework web scritto in Python che fornisce una collezione di librerie e di moduli per lo sviluppo di applicazioni web. Flask è basato sul *Jinja template engine* e sul *Werkzeug WSGI toolkit* ed appartiene alla categoria dei micro-framework.

Figura 2.4: Logo di Flask. [2]

2.2.2 Servizi iniziali

Di seguito sono descritti brevemente i due servizi presenti inizialmente nell'applicazione:

- **`flask_rehub`**: servizio in ascolto sulla porta 8101 che implementa il calcolo della compatibilità e la generazione di una classifica di strutture riabilitative all'interno della sezione *Riabilitazione*.
- **`flask_rsa`**: servizio in ascolto sulla porta 8102 che implementa il calcolo della compatibilità e la generazione della classifica di strutture RSA nella pagina *Ricerca* della sezione *RSA*.

2.3 Architettura estesa

Nel lavoro di tesi, l'applicazione è stata estesa con (i) l'aggiunta di un cruscotto composto da due pagine, i.e. la pagina *Statistiche* e la pagina *Esporta*, e con (ii) l'aggiunta di un sistema di raccomandazione per supportare l'utente durante la fase di ricerca di strutture RSA. Per realizzare queste funzionalità, l'architettura iniziale è stata quindi estesa con

l'aggiunta di quattro nuovi servizi e con l'aggiunta di un database per la memorizzazione dei dati delle ricerche utenti. Una rappresentazione dell'architettura estesa è mostrata in Figura 2.5.

I quattro servizi sono stati sviluppati con la stessa metodologia dei due servizi presenti inizialmente: ciascuno di essi è stato sviluppato con il framework Flask, è ospitato all'interno di un *container* Docker separato ed è in ascolto su una porta specifica. Questi servizi svolgono le funzioni di analisi e manipolazione dei dati richiesti dalle funzionalità aggiunte.

Per quanto riguarda la memorizzazione dei dati, si è scelto di integrare nell'applicazione un DBMS per la gestione dei dati delle ricerche utenti. Dopo uno studio dei modelli esistenti, è stato selezionato un modello NoSQL per la rappresentazione dei dati ed è stato selezionato MongoDB come base di dati di riferimento, come descritto nel Capitolo 3. In particolare, nel lavoro di tesi si è scelto di utilizzare la piattaforma cloud di MongoDB, i.e. MongoDB Atlas, in modo da delegare la gestione dell'istanza della base dati al servizio e poter focalizzare l'attenzione sugli aspetti di progettazione e sviluppo delle funzionalità.

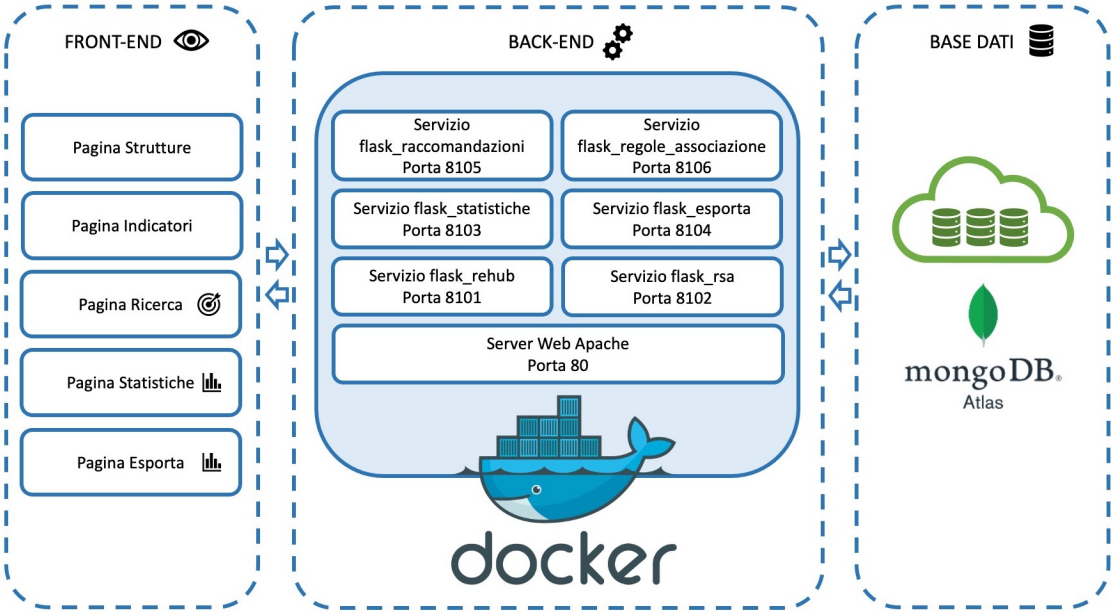


Figura 2.5: Architettura dell'applicazione estesa nel lavoro di tesi.

2.3.1 Tecnologie utilizzate

Di seguito sono descritte brevemente le tecnologie utilizzate nell'architettura estesa, i.e. le librerie Python per l'analisi dei dati e MongoDB Atlas.

Librerie Python per l'analisi dei dati

Nello sviluppo dei nuovi servizi, sono state utilizzate le seguenti librerie Python per l'analisi e la manipolazione dei dati:

- **Pandas** [3]: libreria che fornisce strumenti veloci, potenti e flessibili per l'analisi e la manipolazione dei dati. In particolare, di questa libreria è stata utilizzata la struttura dati *DataFrame* che permettere di importare i dati da diversi formati, quali il CSV e il JSON, e permette di effettuare operazioni di pulizia, interrogazione e aggregazione degli stessi.
- **NumPy** [4]: libreria che fornisce funzioni matematiche di alto livello per operare con vettori n-dimensionali e con matrici.
- **SciPy** [5]: libreria per il calcolo scientifico contenente moduli per svolgere operazioni di ottimizzazione, integrazione, interpolazione, elaborazione di segnali e di immagini.
- **Scikit-learn** [6]: libreria contenente le implementazioni di innumerevoli algoritmi di Data Mining e Machine Learning per effettuare operazioni di preprocessing, classificazione, regressione e clustering.
- **MLxtend** [7]: libreria contenente ulteriori implementazioni di algoritmi di Machine Learning, tra cui l'implementazione di algoritmi per l'estrazione delle regole di associazione.

MongoDB Atlas

MongoDB Atlas [16] è il servizio cloud globale del database MongoDB. Il servizio si appoggia ad Amazon AWS, Microsoft Azure e Google Cloud e garantisce disponibilità, scalabilità e conformità agli standard di privacy e sicurezza dei dati più elevati.

Figura 2.6: Logo di MongoDB Atlas. [16]

2.3.2 Servizi aggiunti

I servizi che sono stati aggiunti nel lavoro di tesi sono:

- `flask_statistiche`: servizio in ascolto sulla porta 8103 che si occupa del calcolo delle statistiche delle ricerche utenti per strutture RSA da visualizzare nella pagina *Statistiche* del cruscotto. I dettagli di questo servizio sono descritti nel Paragrafo 5.2.2.
- `flask_esporta`: servizio in ascolto sulla porta 8104 che si occupa della generazione del file CSV da esportare nella pagina *Esporta* del cruscotto. I dettagli di questo servizio sono descritti nel Paragrafo 5.3.2.

- **`flask_raccomandazioni`**: servizio in ascolto sulla porta 8105 che implementa la componente *collaborative filtering* del sistema di raccomandazione proposto. I dettagli di questo servizio sono descritti nel Paragrafo 7.4.1.
- **`flask_regole_associazione`**: servizio in ascolto sulla porta 8106 che implementa la componente *association rules-based* del sistema di raccomandazione proposto. I dettagli di questo servizio sono descritti nel Paragrafo 7.4.2.

Capitolo 3

Memorizzazione dei dati

3.1 Introduzione

Le basi di dati sono lo strumento informatico che permette la memorizzazione e la gestione affidabile ed efficiente dei dati [8]. Le basi di dati differiscono tra loro principalmente per il modello di dati che implementano.

Il modello dei dati consiste in una collezione di costrutti utilizzati per organizzare i dati di interesse e descriverne la struttura in modo che risulti comprensibile ad un elaboratore. In base al modello utilizzato, le basi di dati si suddividono principalmente in due categorie: le basi di dati relazionali e le basi di dati non relazionali. Nei paragrafi seguenti sono descritte e confrontate le caratteristiche principali di queste due categorie.

3.2 Basi di dati relazionali

Le basi di dati relazionali sono una categoria di database che strutturano e manipolano i dati secondo il modello relazionale.

3.2.1 Il modello relazionale

Il modello relazionale è un modello logico proposto da Edgar F. Codd [9] che si basa sul concetto matematico di relazione. L'assunto fondamentale di questo modello è che tutti i dati siano rappresentati come relazioni e manipolati tramite gli operatori dell'algebra relazionale.

Nel modello di Codd una relazione è definita come un'entità contenente un insieme di tuple, ciascuna composta da una serie di valori appartenenti ad attributi determinati. I valori assunti da ciascun attributo devono essere conformi al dominio specifico di quell'attributo, il quale definisce l'insieme di valori che esso può assumere. Secondo il modello relazionale, un database memorizza quindi i dati sotto forma di un insieme di relazioni composte da tuple contenenti valori. Lo schema di relazioni, attributi e domini, a cui tutti i dati devono sottostare uniformemente, è definito a priori nel momento della creazione della base dati.

3.2.2 Database RDBMS

I sistemi di gestione di basi di dati che si basano sul modello relazionale prendono il nome di RDBMS (acronimo di *Relational Data Base Management Systems*).

Componenti principali

Nelle applicazioni concrete, gli RDBMS riprendono i concetti teorici di relazione e di tupla proposti dal modello relazionale per definire i componenti principali utilizzati per la memorizzazione dei dati.

I componenti principali di un RDBMS sono:

- **Database**: memorizza una o più tabelle.
- **Tabella**: è la rappresentazione del concetto teorico di relazione e definisce il modo in cui i dati devono essere strutturati per essere memorizzati. Una tabella è definita da un insieme di colonne, ciascuna delle quali è caratterizzata da un nome univoco e un tipo di dato che specifica la categoria di dati a cui devono appartenere i valori che verranno memorizzati in essa. All'interno di una tabella i dati sono memorizzati sotto forma di righe.
- **Colonna**: è la rappresentazione del concetto di attributo; il tipo di dato associato ad essa corrisponde al suo dominio.
- **Riga**: è la rappresentazione di una tupla, anche detta record, all'interno di una tabella e definisce un insieme di possibili valori assunti dagli attributi della tabella. Ciascuno dei valori deve appartenere al dominio dell'attributo corrispondente.

Un esempio di tabella è mostrato in Figura 3.1.

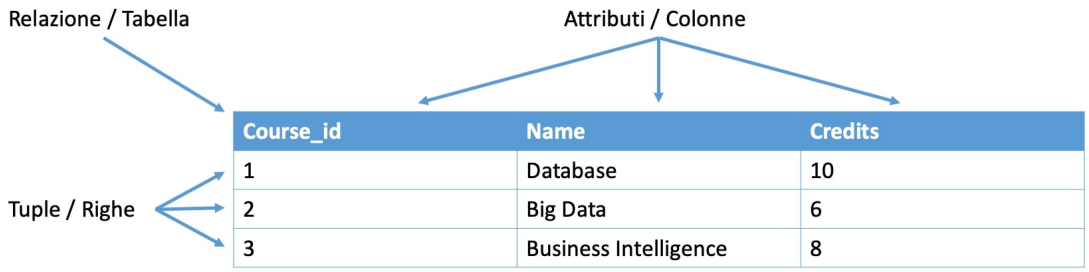


Figura 3.1: Rappresentazione di una tabella relazionale.

Il linguaggio SQL

I DBMS relazionali si basano sullo standard SQL (acronimo di *Structured Query Language*) per quanto riguarda il linguaggio di interfacciamento con il database. SQL è un

linguaggio basato sull'algebra relazionale che permette non solo di effettuare operazioni di manipolazione dei dati ma anche di definire o alterare lo schema della base dati stessa.

Il linguaggio SQL, per la varietà di operazioni che consente di effettuare, appartiene alle seguenti categorie di linguaggio:

- **DDL**: acronimo di *Data Definition Language* in quanto permette di definire lo schema della base dati permettendo di creare nuove tabelle e di eliminare quelle esistenti.
- **DML**: acronimo di *Data Manipulation Language* in quanto permette di inserire nuovi record all'interno delle tabelle e di cancellare o aggiornare record già memorizzati.
- **DQL**: acronimo di *Data Query Language* in quanto permette di interrogare il database consentendo di reperire i dati memorizzati nelle tabelle.

Esempi di comandi SQL di tipo DDL, DML e DQL sono riportati nel Listato 3.1.

```sql
/* Comando DDL */
CREATE TABLE Courses (
    Course_id INTEGER,
    Name VARCHAR,
    Credits INTEGER
);

/* Comando DML */
INSERT INTO Courses (Course_id, Name, Credits)
VALUES (1, "Database", 10);

/* Comando DQL */
SELECT Name FROM Courses
WHERE Credits=10;
```

Listato 3.1: Esempi di comandi SQL.

Transazioni ACID

Nel contesto delle basi di dati le transazioni consistono in una sequenza di operazioni che costituiscono una singola unità logica di lavoro. Il meccanismo delle transazioni permette di eseguire operazioni multiple in modo affidabile garantendo sempre la consistenza e l'integrità dei dati anche in caso di fallimenti: o tutte le operazioni vengono eseguite con successo sulla base dati oppure l'intera transazione viene annullata ripristinando lo stato precedente del database.

Le basi di dati relazionali garantiscono l'integrità e la correttezza dei dati a seguito dell'esecuzione di ogni transazione, imponendo il rispetto di quattro proprietà denominate *ACID*:

- **Atomicità**: ogni transazione deve essere considerata come un'unità atomica indivisibile. La transazione è considerata avere successo solo se tutte le singole operazioni che la compongono sono eseguite con successo; se solo una di esse fallisce anche l'intera transazione fallisce e deve essere annullata.

- **Consistenza**: dopo ogni transazione deve essere sempre garantito uno stato consistente della base dati, preservandone i vincoli di integrità.
- **Isolamento**: l'esecuzione di ogni transazione deve essere indipendente dall'esecuzione di ogni altra transazione sul database. Se più transazioni vengono eseguite in parallelo, gli effetti di ciascuna di esse non devono avere ripercussioni sulle altre, come se venissero eseguite singolarmente.
- **Durabilità**: I cambiamenti apportati dal completamento di una transazione devono essere salvati nel database in modo da non essere persi in caso di guasti.

3.3 Basi di dati NoSQL

I database NoSQL (acronimo di *Not Only SQL*) sono una categoria di database che non si basano sul tradizionale modello relazionale per la memorizzazione dei dati. Il termine NoSQL fu usato per la prima volta nel 1998 da Carlo Strozzi per indicare una base dati open-source che non usava un'interfaccia SQL.

Questa categoria di database negli ultimi anni ha acquisito sempre più popolarità a causa della necessità di memorizzare quantità sempre maggiori di dati e di tipologia sempre meno strutturata. Questo andamento sembrerebbe in continua crescita: gli analisti di IDC stimano infatti che per il 2025 il volume dei dati prodotti in un anno a livello globale raggiungerà i 175 zettabyte e che circa l'80% di questi dati saranno di tipo non strutturato [10]. In questo contesto le caratteristiche principali delle basi di dati relazionali, ossia la necessità di dover definire a priori uno schema strutturato per la memorizzazione dei dati e la necessità di garantire le proprietà ACID, possono risultare fortemente limitanti.

I database NoSQL si pongono quindi come obiettivo di superare le limitazioni del modello relazionale proponendo nuovi modelli per la memorizzazione e la gestione del dato, anche a costo di perdere alcuni dei benefici garantiti dagli RDBMS.

3.3.1 Caratteristiche principali

Flessibilità

I database NoSQL non vincolano il modello dei dati ad essere strutturato secondo uno schema predefinito, rendendoli quindi adatti anche alla memorizzazione di dati semi-strutturati e non-strutturati. Questa caratteristica viene anche definita *schema-less* e permette di apportare modifiche alla struttura dei dati in modo semplice e in tempo reale senza periodi di disservizio.

Scalabilità orizzontale

I database NoSQL sono progettati come sistemi distribuiti per poter scalare facilmente in maniera orizzontale. La scalabilità orizzontale prevede la suddivisione dei dati e del carico di lavoro tra molteplici macchine diverse, in modo da poter incrementare la capacità di memorizzazione e le prestazioni complessive del sistema con l'aggiunta di nuove macchine.

Grazie a questo sistema ogni macchina, gestendo solo un sottoinsieme dei dati e delle operazioni totali, può fornire prestazioni efficienti anche senza l'uso di hardware di fascia

alta e il sistema, nel complesso, può fornire prestazioni superiori rispetto ad una singola macchina ad alta capacità e alte prestazioni. L'aggiunta di nuove macchine al sistema può avere inoltre un costo minore che incrementare le prestazioni di una singola macchina con hardware di alto livello.

Oltre ai vantaggi in termini di facilità nell'incrementare capacità e prestazioni, questo sistema garantisce inoltre disponibilità costante della base dati anche in caso di guasti delle singole macchine. Se una delle macchine del sistema smette di funzionare, le operazioni di lettura e scrittura possono comunque essere dirette a tutte le altre macchine funzionanti, senza tempi di disservizio.

Lo svantaggio che comportano tutti i vantaggi della scalabilità orizzontale è un incremento di complessità nell'architettura del sistema e una maggiore difficoltà di mantenimento.

Proprietà BASE

I database NoSQL, essendo sistemi distribuiti, non possono fornire forti garanzie di consistenza come quelle garantite dai database relazionali con le proprietà ACID. Questo per via di alcune limitazioni intrinseche nell'utilizzo di sistemi distribuiti che furono enunciate in un teorema da Eric Brewer nel 2000 al PODC (*Principle of Distributed Computing*) [11]. Il teorema prende il nome di teorema CAP, acronimo delle seguenti tre proprietà:

- **Consistency**: questa proprietà garantisce che tutti i client che si connettono al sistema distribuito vedano gli stessi dati nello stesso momento, indipendentemente dal nodo a cui sono connessi.

- **Availability**: questa proprietà garantisce che ad ogni richiesta da parte di un client sia sempre fornita una risposta anche se uno o più nodi sono temporaneamente non disponibili.

- **Partition Tolerance**: questa proprietà garantisce che il sistema distribuito continui a funzionare anche nel caso in cui la comunicazione tra due o più nodi sia temporaneamente interrotta.

Secondo il teorema, un sistema distribuito è in grado di soddisfare al massimo due di queste proprietà allo stesso tempo, ma non tutte e tre contemporaneamente. Brewer propose quindi un nuovo insieme di proprietà, in contrapposizione alle proprietà ACID, che fossero più adatte ad un contesto distribuito. Queste proprietà sono conosciute tramite il loro acronimo BASE:

- **Basic Availability**: questa proprietà indica la capacità di un sistema di garantire l'*availabilty* come definita nel teorema CAP.

- **Soft State**: questa proprietà indica che lo stato del sistema può cambiare nel tempo, anche in periodi di tempo senza input a causa della proprietà di *eventual consistency*. La consistenza dei dati diventa quindi un problema da risolvere da parte dello sviluppatore e non deve essere gestita dal database.

- **Eventual Consistency**: questa proprietà indica che quando nuovi dati vengono aggiunti al sistema, essi si propagheranno gradatamente fino a far diventare consistente l'intero sistema, come definito nel teorema CAP, in un certo intervallo di tempo durante il quale il sistema non riceverà input dall'esterno.

I database NoSQL si basano quindi sulle proprietà BASE per poter garantire maggiore flessibilità e semplicità nello scalare orizzontalmente come sistemi distribuiti, a scapito di una minore garanzia di consistenza.

3.3.2 Modelli principali

Al contrario dei database relazionali che si basano su un modello dei dati comune, i.e. il modello relazionale, i database NoSQL non fanno tutti riferimento ad un modello di dati in particolare. I database non-relazionali si possono quindi suddividere in diverse categorie in base al modello con cui memorizzano i dati. Le categorie di basi di dati NoSQL attualmente più diffuse sono [12]:

- **Key-Value**: è il modello di database NoSQL meno complesso. Ogni dato nella base dati è memorizzato sotto forma di coppia chiave-valore. Il valore può essere di qualsiasi tipo e può contenere a sua volta un nuovo insieme di coppie chiave-valore incapsulate in un oggetto. La semplicità di questo modello rende questa categoria la più facilmente scalabile tra i database NoSQL, che è quindi particolarmente adatta a memorizzare quantità di dati molto elevate. Esempi di database di questa categoria sono Redis, Memcached e Scalaris.

- **Column**: in contrapposizione ai database relazionali che memorizzano i dati in righe di tabelle, i database orientati alle colonne memorizzano i dati in singole colonne separatamente. Questa struttura rende questa categoria di database particolarmente adatta all'analisi dei dati, in quanto è possibile accedere direttamente e velocemente ai dati di interesse ed effettuare operazioni di aggregazione in modo efficiente ed ottimizzato. Questo modello permette inoltre di aggiungere facilmente nuove colonne alla base dati in quanto ogni colonna è indipendente da tutte le altre. Lo svantaggio principale di questa struttura si presenta all'aggiunta di nuovi record, in quanto è necessario aggiornare tutte le colonne interessate e richiede più scritture su memoria di massa. I database orientati alle colonne sono quindi poco adatti in scenari che richiedono aggiunta o aggiornamento costanti di record nella base dati. Esempi di database di questa categoria sono Apache HBase, Cassandra, Hypertable e Google BigTable.

- **Graph**: è la categoria di database NoSQL più complessa, specializzata nella memorizzazione di connessioni tra elementi. Ogni elemento è memorizzato come un nodo di un grafo. Le connessioni sono anche chiamate collegamenti o relazioni e sono memorizzate direttamente nel database. Questa struttura risulta particolarmente adatta nel rilevare e ricercare connessioni tra i dati in modo efficiente, cosa che in un database relazionale richiederebbe invece di effettuare operazioni di JOIN tra diverse tabelle. Esempi di database di questa categoria sono Neo4J, Infinite Graph e Amazon Neptune.

- **Document**: questa categoria di database memorizza i dati sotto forma di documenti semi-strutturati come JSON, BSON o XML. All'interno di ciascun documento i dati sono memorizzati come un insieme di campi a cui è associato un valore. I documenti possono essere memorizzati e recuperati in una struttura che è molto simile alle entità oggetto usate per rappresentare i dati nelle applicazioni, al contrario di quanto avviene con i database SQL in cui è spesso necessaria una fase di conversione durante il trasferimento di dati da e verso l'applicazione. Inoltre, questa categoria di database è particolarmente popolare tra gli sviluppatori in quanto la struttura flessibile dei documenti rende facilmente adattabile il modello con cui i dati sono memorizzati nella base dati all'evolversi della struttura dei dati lato applicazione. Esempi di database basati su documenti sono MongoDB e CouchDB.

3.4 Confronto e valutazioni

In Tabella 3.1 è riportato il confronto tra le caratteristiche principali dei database relazionali e quelle dei database NoSQL.

RDBMS	*NoSQL*
Modello dei dati relazionale	Modello dei dati variabile
Schema dei dati strutturato e normalizzato	Schema dei dati flessibile
Scalabilità verticale	Scalabilità orizzontale
Proprietà ACID	Proprietà BASE

Tabella 3.1: Confronto tra le basi di dati relazionali e le basi di dati NoSQL.

A seguito del confronto di queste due categorie di database e dei relativi vantaggi e svantaggi in riferimento ai requisiti delle funzionalità dell'applicazione da sviluppare, si è optato per una base dati di tipo NoSQL basata su documenti. Le motivazioni che hanno portato a questa scelta sono:

- la possibilità di gestire uno schema dei dati flessibile, il quale risulta particolarmente utile nel caso di un'applicazione in fase di sviluppo, come quella in oggetto, in quanto permette di adattare i dati da memorizzare ad eventuali future evolutive dell'applicazione, come ad esempio l'aggiunta di nuovi parametri nella sezione di ricerca, in modo semplice senza dover apportare modifiche alla base dati;
- la possibilità di memorizzare tutti i dati relativi ad una ricerca utente all'interno di un singolo documento, permettendo quindi una maggiore efficienza nelle operazioni di lettura e scrittura dei dati richieste dalle nuove funzionalità da sviluppare. Grazie a questa caratteristica è infatti possibile leggere tutti i dati delle ricerche tramite una singola operazione di lettura, senza la necessità di raggruppare dati da tabelle diverse tramite costose operazioni di JOIN come accadrebbe nel caso di utilizzo di un database relazionale.

Per quanto riguarda la caratteristica di scalabilità orizzontale, non la si ritiene attualmente rilevante nel contesto in analisi in quanto, essendo l'applicazione in oggetto un'applicazione utilizzabile solo per casi di esigenza sanitaria molto specifici e limitata al solo

territorio regionale, non si prevede la memorizzazione di grandi moli di dati e la gestione di grandi carichi di lavoro. Potrebbe invece risultare utile nel caso in cui, negli sviluppi futuri dell'applicazione, l'utilizzo della stessa venisse esteso all'intero territorio nazionale.

Lo svantaggio principale che comporta invece la scelta di un modello NoSQL è che la consistenza dei dati memorizzati non è garantita dalla base dati. Questa responsabilità viene quindi spostata sul lato applicativo.

Tra i database NoSQL basati su documenti è stato scelto come database di riferimento MongoDB, il database più popolare della categoria.

3.5 MongoDB

MongoDB [13] è un DBMS non-relazionale open-source basato su documenti.

Figura 3.2: Logo di MongoDB. [13]

3.5.1 Caratteristiche principali

I componenti principali che MongoDB usa per memorizzare i dati sono:

- **Database**: memorizza una o più collezioni di documenti.
- **Collezione**: memorizza un insieme di documenti semanticamente correlati tra loro. I documenti all'interno di una collezione non seguono uno schema di memorizzazione statico predefinito e possono avere una struttura diversa l'uno dall'altro. Una collezione è il componente corrispondente a una tabella di un database relazionale.
- **Documento**: è il formato con cui MongoDB memorizza i dati. All'interno di un documento i dati sono strutturati secondo un formato simile al JSON (*JavaScript Object Notation*) chiamato BSON (*Binary JSON*) composto da un insieme di campi contenenti i valori da memorizzare. Tutti i documenti memorizzati in MongoDB contengono un campo `_id` che svolge il ruolo di chiave primaria. Un documento è il componente corrispondente ad una riga/record di un database relazionale mentre un campo corrisponderebbe ad una colonna di una tabella.

Formato dei documenti

BSON (*Binary JSON*) [14] è il formato di serializzazione binario con cui MongoDB memorizza i documenti. Questo formato è stato progettato per essere molto più efficiente, sia in termini di spazio richiesto per la memorizzazione dei dati e sia in termini di rapidità nelle operazioni di codifica e decodifica, rispetto al JSON che è un formato testuale. Inoltre,

oltre a supportare gli stessi tipi di dato del JSON, ne estende il formato permettendo la rappresentazione di tipi di dato più complessi, come ad esempio il tipo di dato `Date` e il tipo di dato `Binary` per la memorizzazione di dati temporali e di dati in binario. La dimensione massima di un documento BSON che MongoDB può memorizzare è di 16 MB.

Sharding

Lo *sharding* è il metodo che MongoDB utilizza per supportare nativamente la scalabilità orizzontale. Questo metodo consiste nel distribuire i dati tra diverse macchine chiamate *shard*, che insieme compongono uno *sharded cluster*.

Ciascuno *shard* contiene un sottoinsieme dei dati totali del cluster e, al crescere del volume dei dati, nuovi *shard* possono essere aggiunti al cluster per aumentare la capacità di memorizzazione. Anche il carico delle operazioni di lettura e scrittura è distribuito tra i vari *shard* presenti nel cluster, permettendo a ciascuno di essi di gestire solo un sottoinsieme di tutte le operazioni e garantendo prestazioni elevate anche in situazioni di alto carico di lavoro. Inoltre, nel caso in cui uno o più *shard* diventassero non disponibili, lo *sharded cluster* può comunque continuare ad eseguire operazioni di lettura e scrittura parziali dirigendole verso gli altri *shard* disponibili.

L'architettura che MongoDB utilizza per implementare lo *sharding* è composta dai seguenti componenti:

- **Shard**: componenti che contengono un sottoinsieme dei dati.
- **Mongos**: componenti che rappresentano l'interfaccia tra le applicazioni client e il cluster.
- **Config Servers**: componente che memorizza metadati e impostazioni di configurazione del cluster.

I vantaggi dello *sharding* comportano quindi un'architettura più complessa e più difficile da mantenere.

3.5.2 Interfacciamento con Python: PyMongo

MongoDB ha reso disponibili le API ufficiali per i più popolari linguaggi di programmazione e ambienti di sviluppo per permettere l'interazione con il database [15]. La libreria ufficiale per il linguaggio di programmazione Python è *PyMongo* [17], che è quindi la libreria selezionata per l'interazione tra i servizi del back-end dell'applicazione e il database.

Nei paragrafi seguenti vengono descritte le operazioni principali realizzabili tramite PyMongo: dall'installazione, alla connessione con un'istanza della base dati, ai metodi per operare con essa.

Installazione

Il metodo raccomandato per l'installazione di pymongo prevede l'utilizzo del packet-manager ufficiale di Python, chiamato PIP[1]. Questo sistema permette di installare nuovi moduli

[1] https://pypi.org/project/pip/.

eseguendo un comando da terminale. Nel caso di PyMongo l'istruzione da eseguire è la seguente:

```
$ python -m pip install pymongo
```

Connessione al database

Completata l'installazione della libreria, il primo passo per lavorare con MongoDB consiste nell'importare dal modulo `pymongo` la classe `MongoClient` e nell'istanziare un oggetto della stessa. Questa classe è l'implementazione di un client con cui è possibile connettersi ad un'istanza in esecuzione della base dati. Per identificare l'istanza del server è possibile fornire al costruttore della classe la coppia indirizzo IP e porta oppure fornire una stringa di connessione in formato MongoDB URI.

Effettuata con successo la connessione al server, l'oggetto client fornisce l'accesso ai database memorizzati su di esso. Per accedere al database desiderato, si considera l'oggetto client come se fosse un dizionario Python a cui si fornisce come chiave il nome del database. In base alla chiave inserita verrà restituito un oggetto della classe `Database` che rappresenta l'istanza del database selezionato. In modo analogo, per ottenere l'accesso ad una delle collezioni memorizzate nella base dati è sufficiente fornire come chiave all'oggetto database il nome della collezione, il quale restituisce l'oggetto della classe `Collection` della collezione corrispondente.

Di seguito è riportato un listato di codice di esempio per accedere alla collezione `courses` del database `university` memorizzate in un'istanza di MongoDB in esecuzione localmente:

```
from pymongo import MongoClient

# Connessione ad un'istanza di MongoDB in esecuzione
# all'indirizzo IP 127.0.0.1 e in ascolto sulla porta 27017
client = MongoClient( "127.0.0.1", 27017 )

# Connessione al database
database = client[ "university" ]

# Connessione alla collezione
collection = database[ "courses" ]
```

Operazioni CRUD

Attraverso l'oggetto `collection` è possibile interagire con la collezione ed effettuare tutte le operazioni CRUD fondamentali:

- inserire nuovi documenti tramite i metodi `insert_one()` e `insert_many()`;
- leggere i documenti memorizzati tramite i metodi `find_one()` e `find()`;
- aggiornare i documenti esistenti tramite i metodi `update_one()` e `update_many()`;
- cancellare i documenti esistenti tramite i metodi `delete_one()` e `delete_many()`.

Di seguito sono descritte le due operazioni che verranno utilizzate nei moduli da sviluppare: l'inserimento e la lettura di documenti.

Inserimento di documenti L'inserimento di documenti in una collezione avviene mediante i metodi `insert_one()` e `insert_many()`. Questi metodi ricevono in input il/i documento/i che si vogliono inserire. In PyMongo, per rappresentare l'istanza di un documento si usa un dizionario Python. Le chiavi del dizionario rappresentano le chiavi del documento BSON mentre i valori del dizionario vengono associati ai valori corrispondenti nel documento. I valori, che inizialmente vengono espressi tramite i tipi di dato nativi di Python, vengono successivamente convertiti in modo automatico nei tipi di dato del formato BSON più appropriati. Di seguito è riportato un listato di codice di esempio per la creazione e l'inserimento di un documento all'interno della collezione `courses`:

```
# Creazione di un documento
document = {
   "Name": "Databases",
   "Credits": 10
}

# Inserimento del documento nella collezione
collection.insert_one( document )
```

Lettura di documenti La lettura di documenti memorizzati nella collezione avviene attraverso i metodi `find_one()` e `find()`. Questi metodi ricevono in input due parametri opzionali che consentono di specificare il tipo di documento/i da leggere:

- **`query`**: parametro di tipo dizionario che permette di filtrare i documenti da leggere in base ai valori specificati al suo interno. Se questo parametro non viene specificato, non viene effettuato nessun filtraggio;
- **`projection`**: parametro di tipo dizionario che permette di limitare i campi presenti nei documenti letti ai soli specificati al suo interno. Se questo parametro non viene fornito, i documenti vengono restituiti integralmente e contenenti tutti i campi con cui sono stati memorizzati.

Di seguito è riportato un listato di codice di esempio per la lettura di documenti dalla collezione `courses`:

```
# Definizione del parametro query
query = { "Name": "Databases" }

# Lettura di tutti i documenti aventi il valore Databases
# nel campo Name
documents = collection.find( query )
```

Capitolo 4

Definizione del modello dei dati

In questo capitolo viene descritta la definizione del modello dei dati di MongoDB per la rappresentazione delle ricerche utenti per strutture RSA all'interno della base dati.

Il capitolo è suddiviso in due parti:

- nella prima parte vengono analizzate le sezioni e i campi che compongono la pagina *Ricerca* della sezione *RSA* dell'applicazione, con l'obiettivo di definire i dati da memorizzare;
- nella seconda parte viene descritto il modello selezionato per la memorizzazione di tali dati e vengono descritti nel dettaglio i campi che lo compongono.

4.1 Analisi delle sezioni di ricerca

La pagina *Ricerca* è composta da due sezioni, i.e. la sezione *Ricerca Base* e la sezione *Ricerca Ottimizzata*, attraverso le quali l'utente può effettuare una ricerca in una delle due modalità descritte nel Paragrafo 2.1:

- **Sezione Ricerca Base**: sezione contenente campi che l'utente deve compilare obbligatoriamente e che descrivono oggettivamente le necessità di servizi medicali del paziente.
- **Sezione Ricerca Ottimizzata**: sezione contenente campi che l'utente può compilare facoltativamente e che esprimono delle preferenze soggettive utili per una ricerca più affinata della struttura più appropriata.

Di seguito sono descritte nel dettaglio le due sezioni.

4.1.1 Sezione Ricerca Base

La sezione *Ricerca Base* mostrata in Figura 4.1 è composta dai seguenti campi:

- **Autosufficienza**: campo che permette all'utente di specificare il grado di autosufficienza richiesto agli ospiti della struttura. I valori selezionabili per questo campo sono: *Autosufficienti*, *Non autosufficienti*.
- **Residenzialità**: campo che permette all'utente di specificare la tipologia di assistenza fornita dalla struttura. I valori selezionabili per questo campo sono: *Residenziale*, *Semiresidenziale*.
- **Presidio Alzheimer**: campo che permette all'utente di richiedere la presenza di un'area dedicata a soggetti con disturbi cognitivi e del comportamento. I valori selezionabili per questo campo sono: *Si*, *No*.
- **Accreditamento**: campo che permette all'utente di specificare il grado di accreditamento della struttura. I valori selezionabili per questo campo sono: *Tutti*, *Accreditato*, *Non accreditato*.
- **Provincia**: campo che permette all'utente di selezionare la provincia di locazione della struttura. I valori selezionabili per questo campo sono: *Tutte*, *Alessandria*, *Asti*, *Biella*, *Cuneo*, *Novara*, *Torino*, *Vercelli*, *Verbano-Cusio-Ossola*.

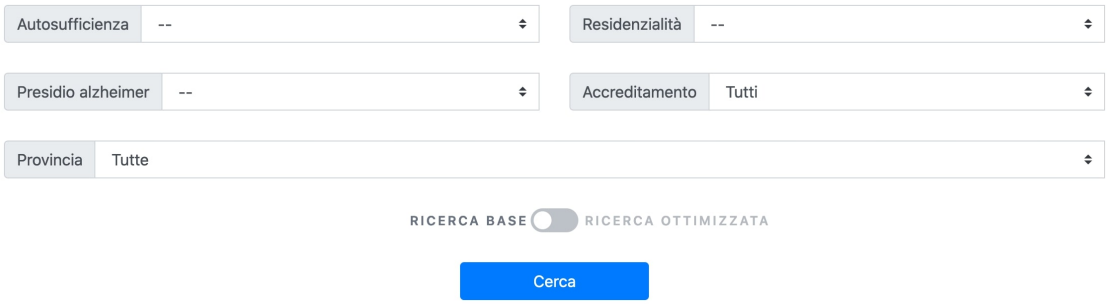


Figura 4.1: Sezione *Ricerca Base*.

4.1.2 Sezione Ricerca Ottimizzata

La sezione *Ricerca Ottimizzata* mostrata in Figura 4.2 è composta dai campi descritti di seguito:

- **Indirizzo**: campo che permette all'utente di specificare l'indirizzo di partenza del paziente. Il valore di questo campo è inserito manualmente dall'utente.
- **Comune**: campo che permette all'utente di specificare il comune di partenza del paziente. Il valore di questo campo è inserito manualmente dall'utente.
- **Distanza**: campo che permette all'utente di indicare il grado di importanza che assegna alla distanza della struttura RSA dalla posizione di partenza del paziente. Il valore di questo campo è selezionato dall'utente spostando il cursore dello slider.

- **Rating**: campo che permette all'utente di indicare il grado di importanza che assegna alle recensioni ricevute dalla struttura RSA. Il valore di questo campo è selezionato dall'utente spostando il cursore dello slider.
- **Tempo di attesa**: campo che permette all'utente di indicare il grado di importanza che assegna al tempo medio di attesa per accedere alla struttura RSA. Il valore di questo campo è selezionato dall'utente spostando il cursore dello slider.
- **Servizi RSA**: campo che permette all'utente di indicare il grado di importanza che assegna ai servizi offerti internamente dalla struttura RSA. Il valore di questo campo è selezionato dall'utente spostando il cursore dello slider.
- **Servizi al contorno**: campo che permette all'utente di indicare il grado di importanza che assegna ai servizi presenti nei dintorni della struttura RSA. Il valore di questo campo è selezionato dall'utente spostando il cursore dello slider.

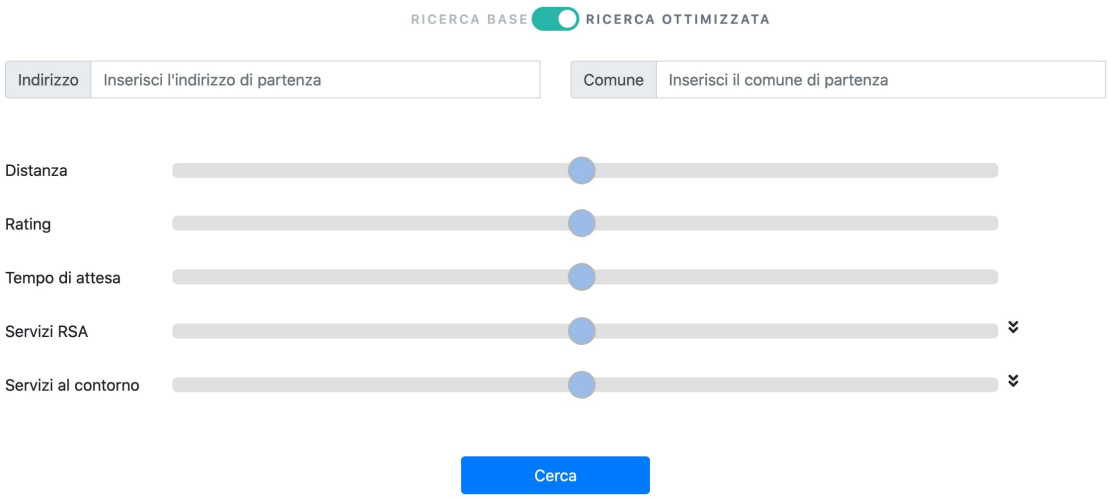


Figura 4.2: Sezione *Ricerca Ottimizzata*.

In questa sezione sono inoltre presenti due sottosezioni che permettono all'utente di indicare, opzionalmente, preferenze specifiche riguardo i servizi offerti dalla struttura RSA. Queste sottosezioni non sono immediatamente accessibili e possono essere rese visibili tramite il bottone presente di fianco agli slider *Servizi RSA* e *Servizi al contorno*. Entrambe le sottosezioni, mostrate in Figura 4.3, sono costituite dai seguenti campi:

- **Categoria**: campo che permette all'utente di selezionare la categoria di servizi di cui desidera indicare una preferenza. A seguito della selezione di una categoria specifica, il contenuto del menu a tendina del campo seguente *Istanza* viene modificato dinamicamente con le istanze di servizi di quella particolare categoria. Esempi di valori selezionabili per questo campo sono: *Ristorazione*, *Cura della persona* e *Attività fisica*.

- **Istanza**: campo che permette all'utente di selezionare il servizio specifico che desidera sia offerto dalla struttura RSA. I valori selezionabili in questo campo dipendono dal valore selezionato nel campo precedente *Categoria*. Esempi di valori selezionabili per questo campo sono: *Animazione, Estetista* e *Podologo*.
- **Interesse**: campo che permette all'utente di indicare il grado di interessamento per il servizio di cui desidera indicare una preferenza. I valori selezionabili per questo campo sono: *Interessato, Molto interessato*.

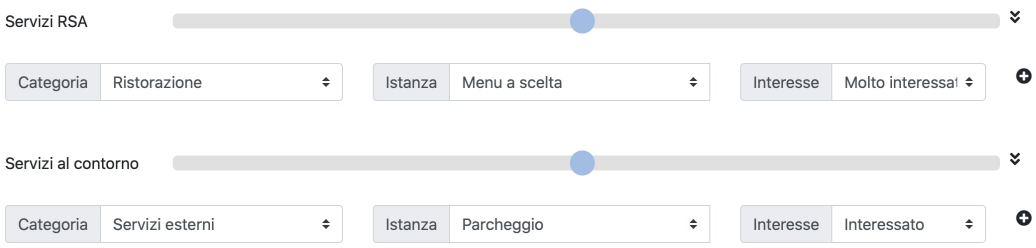


Figura 4.3: Sottosezioni della *Ricerca Ottimizzata* per l'inserimento di servizi.

Una volta selezionati i valori di *Categoria*, *Istanza* e *Interesse*, l'utente può indicare la preferenza per il servizio premendo il bottone con il simbolo + presente sulla destra. L'utente può successivamente ripetere l'operazione quante volte desidera per indicare preferenze per più servizi. Dopo l'inserimento dei servizi le sottosezioni appaiono come mostrato in Figura 4.4.

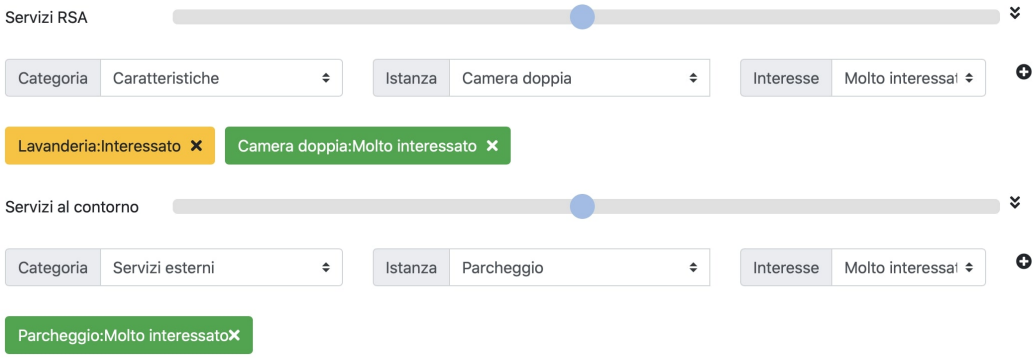


Figura 4.4: Sottosezioni della *Ricerca Ottimizzata* dopo l'inserimento di servizi.

4.2 Strutturazione del modello

Analizzati i dati che compongono una ricerca utente per strutture RSA, si è proceduto con la strutturazione di un modello per la memorizzazione di tali dati.

Come descritto nel Paragrafo 3.5.1, i componenti principali che MongoDB utilizza per la memorizzazione dei dati sono le collezioni e le tabelle. Strutturare un modello per questo DBMS consiste quindi nel definire in quante e quali collezioni suddividere i dati e nel definire la struttura dei documenti per ciascuna di esse.

Per quanto riguarda la definizione delle collezioni, considerando che i dati delle ricerche utenti sono autodescrittivi e che gli stessi non fanno riferimento ad altre sezioni dell'applicazione, si è ritenuto opportuno memorizzare tali dati all'interno di un'unica collezione. Tale soluzione consente inoltre di poter leggere l'intero storico delle ricerche tramite una singola operazione di lettura della collezione, senza la necessità di dover effettuare costose operazioni di aggregazione da collezioni separate.

Il nome selezionato per la collezione è `log_ricerche_RSA` e la definizione della struttura dei documenti da inserire all'interno di essa è descritta nel dettaglio nel paragrafo seguente.

4.2.1 La collezione `log_ricerche_RSA`

La struttura definita per la collezione `log_ricerche_RSA` prevede due modelli di documenti diversi: un primo modello dedicato alla memorizzazione dei dati delle ricerche base ed un secondo modello dedicato alla memorizzazione dei dati delle ricerche ottimizzate. La struttura selezionata è riportata in Tabella 4.1.

		Tipo di ricerca	
Campo	*Tipo di dato*	*Base*	*Ottimizzata*
Utenza	String	✔	✔
Residenzialita	String	✔	✔
Alzheimer	String	✔	✔
Accreditamento	String	✔	✔
Provincia	String	✔	✔
Timestamp	Date	✔	✔
Ricerca_avanzata	Boolean	✔	✔
Luogo_partenza	Object		✔
Coordinate_partenza	Object		✔
Rilevanza	Object		✔
Servizi_RSA	Object		✔
Servizi_contorno	Object		✔

Tabella 4.1: Struttura dei documenti della collezione `log_ricerche_RSA`.

I campi presenti all'interno del modello della ricerca base sono un sottoinsieme dei campi del modello della ricerca ottimizzata. Si è scelto di adottare questa soluzione, al posto di un modello unico per tutte le ricerche, in quanto i dati provenienti dalle ricerche sono diversi in base al tipo di ricerca effettuata. È stato inoltre possibile implementare questa soluzione grazie alla flessibilità offerta da MongoDB nel poter memorizzare all'interno della stessa collezione documenti con strutture diverse. Di seguito sono elencati brevemente i campi inseriti nel modello.

I campi presenti in entrambi i modelli di documento sono:

- **Utenza**, **Residenzialita**, **Alzheimer**, **Accreditamento**, **Provincia**: campi contenenti i valori selezionati dall'utente nella sezione *Ricerca Base*.
- **Timestamp**: campo aggiunto per la memorizzazione temporale del momento in cui è stata effettuata la ricerca.
- **Ricerca_avanzata**: campo aggiunto per distinguere formalmente le due tipologie di documento.

I campi presenti esclusivamente nei documenti riferiti ad una ricerca ottimizzata sono:

- **Luogo_partenza**: campo contenente i valori dei campi *Indirizzo* e *Comune* inseriti dall'utente nella sezione *Ricerca Ottimizzata*.
- **Coordinate_partenza**: campo aggiunto per la memorizzazione delle coordinate del luogo di partenza del paziente.
- **Rilevanza**: campo contenente i valori di rilevanza espressi dall'utente attraverso gli slider nella sezione *Ricerca Ottimizzata*.
- **Servizi_RSA**: campo contenente la lista di servizi RSA di cui l'utente ha indicato un interessamento nella sottosezione *Servizi RSA*.
- **Servizi_contorno**: campo contenente la lista di servizi al contorno di cui l'utente ha indicato un interessamento nella sottosezione *Servizi al contorno*.

In fase di definizione del modello, oltre ai campi elencati in precedenza sono stati valutati e in seguito scartati i seguenti campi per entrambi i modelli di documento:

- **Indirizzo_IP**: campo valutato per la memorizzazione dell'indirizzo IP del client che effettua la ricerca, in quanto informazione aggiuntiva per la geolocalizzazione approssimativa del luogo di provenienza della ricerca.
- **Cookie_ID**: campo valutato per la memorizzazione di un cookie lato client, con il fine di poter correlare le ricerche effettuate dallo stesso utente non essendo prevista dall'applicazione la registrazione di un profilo utente.

Nei paragrafi seguenti vengono descritti nel dettaglio i campi elencati in precedenza e le valutazioni effettuate in fase di definizione del modello per ciascuno di essi. Sono infine mostrati alcuni esempi di ricerche utenti memorizzate all'interno della collezione.

Descrizione dei campi

Utenza, Residenzialita, Alzheimer, Accreditamento, Provincia Questi campi riportano i valori selezionati dall'utente nei campi omonimi della sezione *Ricerca Base*: *Autosufficienza, Residenzialità, Presidio Alzheimer, Accreditamento* e *Provincia*. Il tipo di dato selezionato per la memorizzazione di questi campi è il tipo `String` che riporta letteralmente il valore selezionato dall'utente. Inoltre, al fine di uniformare i valori memorizzati, tutte le stringhe vengono rese via codice interamente maiuscole

prima di essere inserite nella base dati. Ad esempio, se l'utente effettuasse una ricerca selezionando per il campo *Accreditamento* il valore *Non accreditato*, esso verrebbe memorizzato come `NON ACCREDITATO`. In fase di definizione del modello, per questi campi è stata anche valutata la possibilità di una memorizzazione con tipo di dato numerico intero `Int32`, poiché per ognuno di essi l'utente può sempre e solo scegliere tra un insieme finito di valori. Così, ad esempio, i valori del campo *Accreditamento* sarebbero potuti essere memorizzati come 0, 1 e 2 in corrispondenza dei valori possibili per quel campo. Si è comunque preferita la prima soluzione in quanto i valori in formato `String` garantiscono una maggiore chiarezza sia in fase di interrogazione della base dati e sia in fase di analisi dei dati lato server, a discapito di una maggiore quantità di memoria richiesta per la memorizzazione.

`Timestamp` Questo campo è un campo addizionale, non proveniente direttamente dai parametri selezionati dall'utente. Esso è stato inserito per poter memorizzare in modo formale un'indicazione temporale precisa del momento in cui la ricerca è stata effettuata. Questo campo viene calcolato lato server al momento della ricezione della query di ricerca ed inserito nel documento corrispondente prima della memorizzazione nella base dati. In particolare, il server calcola il valore di questo campo utilizzando il metodo `now` della classe `datetime` presente all'interno della libreria Python omonima `datetime`. Il formato del timestamp risultante è una data in formato UTC composta nel modo seguente: `YYYY-mm-ddTHH:MM:ss`. Poiché MongoDB supporta nativamente la memorizzazione di dati temporali mediante il tipo di dato `Date`, questo è il tipo di dato selezionato per memorizzare questo campo.

`Ricerca_avanzata` Questo campo è un campo addizionale, non proveniente direttamente dai parametri selezionati dall'utente. Esso è stato inserito per memorizzare esplicitamente la tipologia di ricerca effettuata. Per discriminare tra una ricerca base e una ricerca ottimizzata, essendo possibili solo queste due opzioni, per questo campo è stato selezionato il tipo di dato `Boolean`: nel caso di una ricerca ottimizzata il campo assume il valore `True`, mentre nel caso di una ricerca base il campo assume il valore `False`. In fase di definizione del modello, si è valutato se inserire o meno questo campo in quanto, poiché i documenti che memorizzano una ricerca base contengono solo un sottoinsieme dei campi contenuti nei documenti che memorizzano una ricerca ottimizzata, si sarebbe potuto discriminare tra le due tipologie di ricerca in base alla presenza o meno dei campi riferiti ad una ricerca ottimizzata all'interno del documento. Si è comunque optato per l'inserimento di questo ulteriore campo per consentire una maggiore formalità nel distinguere tra i due tipi di ricerca in fase di interrogazione del database e in fase di analisi dei dati.

`Luogo_partenza` Questo campo, presente solo se l'utente effettua una ricerca ottimizzata, contiene i valori inseriti dall'utente nei campi *Comune* e *Indirizzo*. Il tipo di dato scelto per entrambi i campi, poiché si tratta di dati testuali inseriti dall'utente, è il tipo di dato `String`. Poiché i due campi sono semanticamente correlati tra loro, in quanto entrambi riferiti all'indirizzo di partenza del paziente, si è optato per una memorizzazione raggruppata in un unico oggetto. La struttura dell'oggetto è riportata in Tabella 4.2.

Campo	*Tipo di dato*
Comune	String
Indirizzo	String

Tabella 4.2: Struttura del campo Luogo_partenza.

Coordinate_partenza Questo campo è un campo addizionale, non proveniente direttamente dai parametri selezionati dall'utente. Esso è stato inserito per memorizzare le coordinate del luogo di partenza del paziente. Tali coordinate sono calcolate a partire dai valori inseriti dall'utente nei campi *Comune* e *Indirizzo* della sezione *Ricerca Ottimizzata*; questo campo è quindi presente solo se l'utente effettua una ricerca ottimizzata. Il calcolo delle coordinate viene effettuato lato server al momento della ricezione della query di ricerca. Per il calcolo è stato utilizzato il modulo Python **geopy** che consente di ottenere i valori di latitudine e longitudine a partire da una stringa contenente un indirizzo. Per memorizzare questo campo, in fase di definizione del modello, sono state valutate le due tipologie di strutture rese disponibili da MongoDB per la memorizzazione nativa di dati geospaziali, i.e. gli oggetti GeoJSON e le coppie di coordinate tradizionali. Oltre alla differente struttura dell'oggetto utilizzato per la memorizzazione, la differenza tra questi due metodi consiste nel fatto che quando si effettua un'interrogazione su dati geospaziali, Mongo calcola in modo differente le distanze tra punti. Nel caso le coordinate vengano memorizzate come oggetto GeoJSON le distanze tra punti sono calcolate come se essi fossero situati su una superficie sferica, come nel caso di coordinate geografiche. Invece, nel caso le coordinate vengano memorizzate come coppie di valori tradizionali, le distanze tra punti sono calcolate come se i punti fossero situati su un piano euclideo. Poiché nel caso in oggetto le coordinate rappresentano le coordinate geografiche del luogo di partenza del paziente, si è optato per la memorizzazione tramite oggetti GeoJSON. Un oggetto di tipo GeoJSON è composto dai campi seguenti:

- **type**: definisce il tipo di oggetto GeoJSON rappresentato. Per indicare la memorizzazione di coordinate geografiche viene utilizzato il valore Point.
- **coordinates**: definisce un array di due Double contenente i valori delle coordinate da memorizzare, rispettivamente longitudine e latitudine.

La struttura dell'oggetto è riportato in Tabella 4.3.

Campo	*Tipo di dato*
type	String
coordinates	Array<Double>

Tabella 4.3: Struttura del campo Coordinate_partenza.

Rilevanza Questo campo, presente solo se l'utente effettua una ricerca ottimizzata, contiene i valori delle preferenze che l'utente ha espresso attraverso gli slider presenti nella

sezione *Ricerca Ottimizzata*, i.e. *Distanza, Rating, Tempo di attesa, Servizi RSA* e *Servizi al contorno*. Tutti i valori, poiché correlati tra loro, sono stati raggruppati in un unico oggetto. Il tipo di dato scelto per questi campi è il tipo di dato numerico `Int32` in quanto il valore esprimibile attraverso uno slider, a seguito della conversione della posizione del cursore in valore numerico, è un valore intero compreso tra 0 e 100. La struttura dell'oggetto è riportata nella Tabella 4.4.

Campo	*Tipo di dato*
`Distanza`	`Int32`
`Rating`	`Int32`
`Tempo_medio_attesa`	`Int32`
`Servizi_RSA`	`Int32`
`Servizi_contorno`	`Int32`

Tabella 4.4: Struttura del campo `Rilevanza`.

`Servizi_RSA` Questo campo, presente solo se l'utente effettua una ricerca ottimizzata, contiene la lista di servizi RSA per cui l'utente ha espresso un interessamento. Nel caso in cui l'utente non indicasse alcuna preferenza riguardo ai servizi RSA, in quanto opzionale, questo campo risulterà vuoto. Per memorizzare questo campo, in fase di definizione del modello, sono state valutate due tipologie di strutture differenti. La prima struttura valutata consiste in una struttura di tipo `Array<Object>`, costituita da un array di oggetti, ciascuno contenente i campi dei singoli servizi selezionati raggruppati insieme: `Categoria`, `Istanza` e `Interesse`. Come per gli altri campi della stessa tipologia, il tipo di dato selezionato per la memorizzazione è il tipo di dato `String`. Il vantaggio principale di questa struttura è quello di memorizzare i dati raggruppati in modo logicamente più simile a come inseriti dall'utente in fase di ricerca. Lo svantaggio che comporta questa struttura è quello di avere le informazioni dei singoli servizi annidate all'interno della lista di oggetti. La seconda struttura valutata consiste invece in un unico oggetto composto dai campi `Categorie`, `Interessato` e `Molto_interessato`:

- **`Categorie`** raggruppa al suo interno la lista di categorie di servizi RSA per cui l'utente ha espresso un interessamento.
- **`Interessato`** raggruppa al suo interno la lista di istanze di servizi RSA per cui l'utente ha espresso un interessamento ed ha selezionato nel campo *Interesse* della sottosezione *Servizi RSA* l'opzione *Interessato*.
- **`Molto_interessato`** raggruppa al suo interno la lista di istanze di servizi RSA per cui l'utente ha espresso un interessamento ed ha selezionato nel campo *Interesse* della sottosezione *Servizi RSA* l'opzione *Molto interessato*.

Il tipo di dato selezionato per la memorizzazione di questi tre campi è il tipo `Array<String>`. Questa seconda struttura, raggruppando insieme tutti i valori di *Categoria* e tutti i valori di *Istanza* in liste di stringhe, comporta il vantaggio di permettere un filtraggio semplificato dei documenti in fase di interrogazione del database ed una

gestione più immediata di questi dati in fase di elaborazione lato server. Ad esempio, se si volessero analizzare solo le ricerche in cui è stato esplicitato un interesse per il servizio *Parrucchiere*, sarebbe sufficiente filtrare le ricerche verificando la presenza di tale servizio nei campi `Interessato` e `Molto_interessato`. Gli svantaggi che comporta questo tipo di struttura sono quello di avere una struttura più elaborata rispetto alla prima valutata e quello di perdere la corrispondenza immediata tra i valori di *Istanza* e i valori di *Categoria*. In fase di definizione del modello, si è optato per la seconda struttura in quanto semplifica la fase di elaborazione dati. La struttura selezionata è riportata nella Tabella 4.5.

Campo	*Tipo di dato*
`Categorie`	`Array<String>`
`Interessato`	`Array<String>`
`Molto_interessato`	`Array<String>`

Tabella 4.5: Struttura del campo `Servizi_RSA`.

`Servizi_contorno` Questo campo, presente solo se l'utente effettua una ricerca ottimizzata, contiene la lista di servizi al contorno per cui l'utente ha espresso un interessamento. Nel caso in cui l'utente non indicasse alcuna preferenza riguardo ai servizi al contorno, in quanto opzionale, questo campo risulterà vuoto. Poiché la struttura ed il tipo di dati con cui l'utente può esprimere un interessamento per i servizi al contorno sono i medesimi dei servizi RSA, in fase di definizione del modello le considerazioni su come memorizzare questo campo sono state le stesse del campo `Servizi_RSA` descritto in precedenza. La struttura selezionata per la memorizzazione è quindi la stessa del campo `Servizi_RSA` ed è riportata nella Tabella 4.6.

Campo	*Tipo di dato*
`Categorie`	`Array<String>`
`Interessato`	`Array<String>`
`Molto_interessato`	`Array<String>`

Tabella 4.6: Struttura del campo `Servizi_contorno`.

Campi valutati ma non inseriti

Di seguito sono riportati i campi valutati in fase di definizione del modello che non sono stati poi inseriti nella struttura selezionata, i.e. `Indirizzo_IP` e `Cookie_ID`.

`Indirizzo_IP` Questo campo è stato valutato per la memorizzazione dell'indirizzo IP del client che effettua la ricerca, ricavabile lato server dai pacchetti ricevuti. L'inserimento di questo campo è stato valutato per poter avere un'indicazione del luogo approssimativo dell'utente che effettua la ricerca. Grazie ad essa si sarebbero potute analizzare le caratteristiche delle ricerche effettuate dagli utenti in base alla zona.

Questa informazione, inoltre, non sarebbe stata ridondante rispetto alle informazioni memorizzate nel campo `Luogo_partenza` descritto in precedenza. Primo, perché tale informazione è presente solo nel caso in cui l'utente effettui una ricerca ottimizzata, mentre l'indirizzo IP sarebbe disponibile per qualsiasi tipo di ricerca, anche quelle base. Secondo, perché il luogo di partenza del paziente potrebbe non essere lo stesso luogo dell'utente che effettua la ricerca, ad esempio nel caso in cui fosse un parente del paziente ad effettuarla. In fase di definizione del modello, si è optato infine per il non inserimento del campo in quanto l'informazione di geolocalizzazione fornita dall'indirizzo IP potrebbe non essere attendibile. L'indirizzo IP dei pacchetti ricevuti lato server potrebbe infatti non essere il medesimo del client che effettua la ricerca, come ad esempio accadrebbe nel caso in cui l'utente utilizzasse una VPN o un proxy o si trovasse dietro NAT. In questi casi, l'indirizzo IP del nodo di uscita potrebbe risultare anche molto distante rispetto alla posizione dell'utente. Inoltre, l'informazione riguardante l'indirizzo IP è considerata nel regolamento generale sulla protezione dei dati (GDPR) come informazione personale e avrebbe necessitato di una gestione consona di tale dato.

`Cookie_ID` Questo campo è stato valutato nell'ipotesi di memorizzare un cookie lato client nel browser dell'utente. L'impostazione del cookie è stata valutata in quanto, non essendo prevista dall'applicazione la registrazione di un profilo utente, non sarebbe altrimenti possibile correlare tra loro eventuali ricerche consecutive effettuate dall'utente. Questa informazione sarebbe potuta essere rilevante nello sviluppo del sistema di raccomandazione con il fine di creare uno pseudo-profilo utente a cui fornire raccomandazioni. In fase di definizione del modello, questo campo non è stato infine inserito nel modello in quanto l'informazione basata su un cookie lato browser sarebbe troppo volatile per la creazione di un profilo utente consistente; l'utente infatti causerebbe la reimpostazione di un nuovo cookie ad ogni cancellazione dei dati di navigazione del browser. Inoltre, nel caso in cui l'utente utilizzasse più di un dispositivo per accedere all'applicazione, le ricerche effettuate dallo stesso sarebbero comunque considerate provenienti da dispositivi diversi e, di conseguenza, associate a utenti diversi.

Esempi di ricerche memorizzate

Esempi di documenti che memorizzano una ricerca base e una ricerca ottimizzata sono riportati nei Listati 4.1 e 4.2.

```
{
    "Alzheimer": "NO",
    "Accreditamento": "ACCREDITATO",
    "Residenzialita": "SEMIRESIDENZIALE",
    "Utenza": "AUTOSUFFICIENTI",
    "Provincia": "CUNEO",
    "Timestamp": 2020-07-12T08:32:01,
    "Ricerca_avanzata": False
}
```

Listato 4.1: Esempio di documento di una ricerca base.

```
{
    "Alzheimer": "SI",
    "Accreditamento": "NON ACCREDITATO",
    "Residenzialita": "RESIDENZIALE",
    "Utenza": "NON AUTOSUFFICIENTI",
    "Provincia": "TORINO",
    "Timestamp": 2020-07-12T08:32:01,
    "Ricerca_avanzata": True,
    "Luogo_partenza": {
        "Comune": "Torino",
        "Indirizzo": "Corso Duca degli Abruzzi, 24"
    },
    "Coordinate_partenza": {
        "type": "Point",
        "coordinates": [7.66,45.06]
    },
    "Rilevanza": {
        "Distanza": 25,
        "Rating": 80,
        "Tempo_medio_attesa": 60,
        "Servizi_RSA": 90,
        "Servizi_contorno": 15
    },
    "Servizi_RSA": {
        "Categorie": ["Attivita' interne","Caratteristiche"],
        "Interessato": ["Animazione","Laboratorio"],
        "Molto_interessato": ["Camera singola"]
    },
    "Servizi_contorno": {
        "Categorie": ["Servizi esterni"],
        "Interessato": ["Parcheggio"],
        "Molto_interessato": []
    },
}
```


Listato 4.2: Esempio di documento di una ricerca ottimizzata.

Capitolo 5

Sviluppo del cruscotto

In questo capitolo viene descritto come è stato sviluppato e implementato il cruscotto all'interno dell'applicazione.

Il capitolo è suddiviso in tre parti:

- nella prima parte viene descritto come è stata creata e popolata con dati sintetici la collezione `log_ricerche_RSA`, i cui dati sono poi stati utilizzati per la dimostrazione delle funzionalità del cruscotto;
- nella seconda parte viene descritta come è stata sviluppata la prima delle due pagine che compongono il cruscotto proposto, i.e. la pagina *Statistiche*;
- nella terza parte viene descritta come è stata sviluppata la seconda delle due pagine che compongono il cruscotto proposto, i.e. la pagina *Esporta*.

5.1 Popolamento della base dati

La prima fase nello sviluppo del cruscotto è consistita nella creazione e nel popolamento con dati sintetici di un'istanza della collezione `log_ricerche_RSA`. Tale operazione è stata effettuata, a fini dimostrativi, per simulare la presenza di dati di ricerche utenti reali; questi dati non sono infatti realmente disponibili in quanto l'applicazione di riferimento, nel momento della stesura di questa tesi, è ancora in fase di sviluppo.

Prima dell'operazione di popolamento è stata creata un'istanza di una base dati MongoDB in cui ospitare la collezione. Come descritto nel Paragrafo 2.3, si è optato per la creazione di un'istanza cloud del database attraverso il servizio offerto da MongoDB Atlas [16]. Tale servizio consente poi l'accesso al database, utilizzando ad esempio PyMongo, attraverso una URI denominata MongoDB URI. All'interno di tale istanza è stata quindi creata la collezione `log_ricerche_RSA`.

Per popolare la collezione è stato successivamente sviluppato uno script Python per la generazione automatica e casuale di documenti di ricerche utenti sintetiche, secondo il modello descritto nel Paragrafo 4.2.1, da inserire all'interno della stessa. Di seguito viene descritta brevemente la struttura dello script e come esso è stato utilizzato per il popolamento della collezione.

Struttura dello script

Lo script esegue le tre azioni seguenti:

1. crea una connessione con l'istanza di MongoDB Atlas ed accede alla collezione `log_ricerche_RSA`;

2. istanzia un oggetto della classe `RandomLogGenerator`, i.e. una classe creata appositamente per la generazione di documenti casuali;

3. genera una lista di n documenti casuali attraverso il metodo `generate_document()` della classe `RandomLogGenerator` e li inserisce all'interno della collezione.

Di seguito è descritta brevemente la classe `RandomLogGenerator` utilizzata dallo script per la generazione dei documenti casuali.

La classe `RandomLogGenerator`

La classe `RandomLogGenerator` è una classe creata per la generazione casuale di documenti sintetici secondo il modello della collezione `log_ricerche_RSA`.

Per la generazione casuale dei campi dei documenti nella classe sono stati utilizzati i seguenti moduli:

- **`random`**[1]: modulo della libreria standard di Python dedicato alla generazione pseudo-casuale di numeri. Le funzioni utilizzate sono `randrange()`, che consente di generare un numero casuale all'interno di un intervallo, e `choices()`, che permette di campionare n elementi da un insieme.

- **`numpy.random`**[2]: modulo appartenente alla libreria NumPy di cui è stata utilizzata la funzione `random.random()`, che consente di generare un vettore di n numeri di tipo float casuali, scelti con distribuzione uniforme nell'intervallo tra 0 e 1.

La classe espone un metodo pubblico, i.e. il metodo `generate_document()`, che genera un documento casuale. Il documento è restituito sotto forma di dizionario Python, il formato in cui sono rappresentati i documenti MongoDB da PyMongo, predisposto quindi per l'inserimento all'interno della collezione.

Generazione dei documenti

Nel Listato 5.1 è riportato il codice dello script con cui sono stati generati e inseriti nella collezione `log_ricerche_RSA` 10K documenti casuali.

[1] https://docs.python.org/3.8/library/random.html.

[2] https://numpy.org/doc/stable/reference/random/index.html.

```
from pymongo import MongoClient
from . import RandomLogGenerator

# Connessione all'istanza cloud di MongoDB
client = MongoClient( "mongodb+srv://user:***@cluster***.mongodb.net/" )
database = client[ "database" ]
collection = database[ "log_ricerche_RSA" ]

NUM_DOCUMENTS = 10000
generator = RandomLogGenerator()
documents = []

# Generazione dei documenti
for i in range( NUM_DOCUMENTS ):
    document = generator.generate_document()
    documents.append( document )

# Inserimento dei documenti nella collezione
collection.insert_many( documents )
```

Listato 5.1: Codice dello script con cui è stata popolata la collezione `log_ricerche_RSA`.

5.2 Pagina Statistiche

In questa sezione viene descritto come è stata sviluppata e implementata nell'applicazione la pagina *Statistiche*.

5.2.1 Descrizione

La pagina *Statistiche* consiste in un insieme di elementi grafici che consentono di analizzare le caratteristiche delle ricerche per strutture RSA effettuate dagli utenti finali. In particolare, i grafici mostrati consentono di visualizzare statistiche riguardanti la distribuzione dei valori dei parametri di ricerca selezionati dagli utenti nelle ricerche passate. Tali statistiche sono calcolate a partire dai dati delle ricerche memorizzati all'interno della collezione `log_ricerche_RSA`.

La pagina è stata realizzata con un duplice scopo: fornire un riscontro agli operatori e ai gestori della sanità sulle esigenze di servizi medicali espresse dagli utenti che ricercano una struttura RSA, e fornire a pazienti e familiari informazioni sulle tipologie di servizi richiesti in passato dagli utenti con esigenze sanitarie similari. A tal fine sono state previste diverse modalità di filtraggio dei dati per consentire all'utente di visualizzare statistiche specifiche per determinate configurazioni dei parametri di ricerca selezionati.

5.2.2 Sviluppo

Di seguito viene descritto come sono stati sviluppati il front-end e il back-end della pagina *Statistiche*.

Front-End

La pagina, mostrata in Figura 5.1, è composta verticalmente da due sezioni:

- la sezione di sinistra consiste in un riquadro che permette all'utente di selezionare vari parametri con cui filtrare i dati di cui visualizzare le statistiche. Nella parte superiore del riquadro sono presenti due tab con cui è possibile selezionare due modalità di filtraggio e visualizzazione dei dati: *Statistiche Ricerca Base* e *Statistiche Ricerca Ottimizzata*;
- la sezione di destra consiste in un riquadro contenete i grafici[3] per la visualizzazione delle statistiche. Nella parte superiore del riquadro sono presenti diversi tab con cui è possibile selezionare il parametro del quale si vuole visualizzare il grafico.

Di seguito sono descritte nel dettaglio le due modalità *Statistiche Ricerca Base* e *Statistiche Ricerca Ottimizzata*.

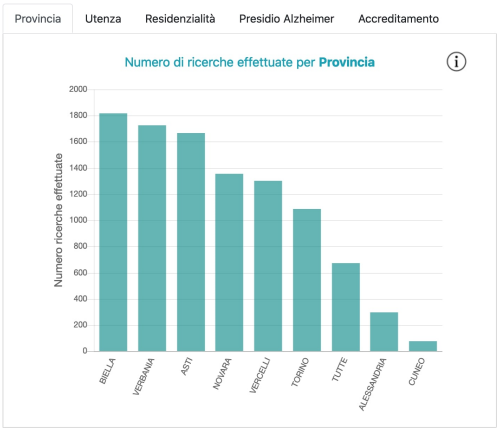
Figura 5.1: Pagina *Statistiche*.

Statistiche Ricerca Base

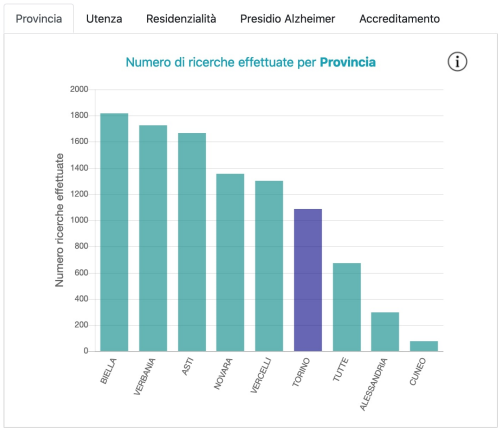
Lo scopo della modalità *Statistiche Ricerca Base* è quello di mostrare i grafici delle statistiche relative ai parametri di ricerca selezionati dagli utenti nella sezione *Ricerca Base*, i.e. *Provincia, Utenza, Residenzialità, Presidio Alzheimer* e *Accreditamento*. Tutti i grafici visualizzabili in questa modalità sono mostrati in Figura 5.2.

Per ciascun campo l'utente ha la possibilità di visualizzare l'istogramma dei valori selezionati nelle ricerche utenti passate per quel campo: sull'asse orizzontale sono riportati tutti valori selezionabili per quel campo, mentre sull'asse verticale è riportato il numero di ricerche effettuate in cui è stato selezionato un dato valore.

[3]Per la realizzazione dei grafici è stata utilizzata la libreria JavaScript Chart.js [18].



(a) Statistiche del campo *Provincia*.

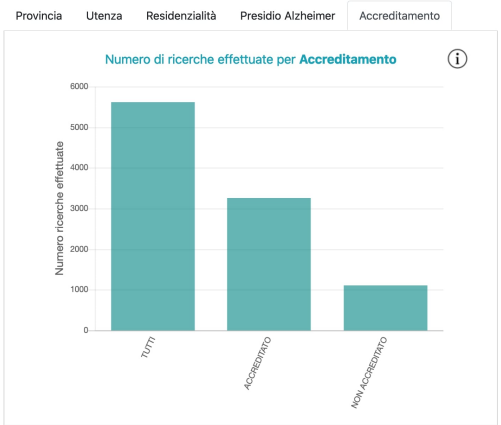


(b) Statistiche del campo *Provincia* a seguito della selezione di una provincia.

(c) Statistiche del campo *Autosufficienza*.

(d) Statistiche del campo *Residenzialità*.

(e) Statistiche del campo *Alzheimer*.



(f) Statistiche del campo *Accreditamento*.

Figura 5.2: Grafici delle statistiche visualizzabili nella modalità *Statistiche Ricerca Base*.

In questa modalità l'utente ha inoltre la possibilità di visualizzare le statistiche filtrate in base al valore del parametro *Provincia*. L'utente può quindi vedere l'andamento dei parametri di ricerca selezionati dagli utenti che hanno ricercato una struttura RSA in una particolare provincia. Il riquadro che consente il filtraggio dei dati in questa modalità è mostrato in Figura 5.3.

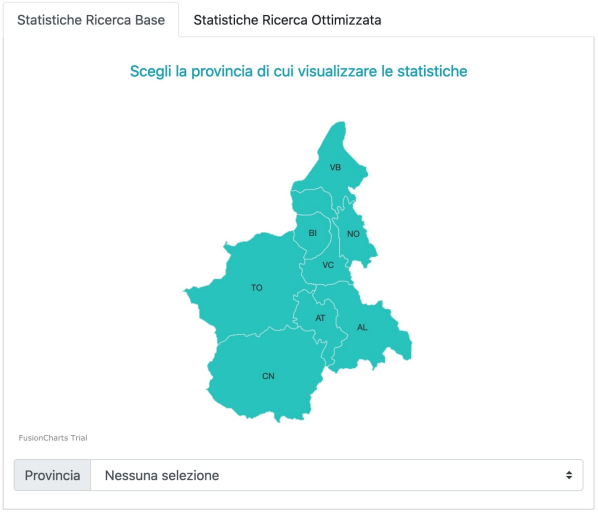


Figura 5.3: Riquadro per il filtraggio dei dati nella modalità *Statistiche Ricerca Base*.

Il riquadro è composto da due elementi: una mappa[4] della regione Piemonte suddivisa in province e un menu a tendina contenente i valori selezionabili per il campo *Provincia*. Interagendo con la mappa oppure selezionando un valore dal menu a tendina, l'utente può selezionare la provincia di cui filtrare i dati. In fase di progettazione del cruscotto, si è optato per mantenere entrambi gli elementi, nonostante svolgano una funzione simile, in quanto il campo *Provincia*, nella pagina *Ricerca*, può assumere anche il valore *Tutte*, il quale risulterebbe difficilmente rappresentabile esclusivamente attraverso la mappa.

Ogni qualvolta l'utente seleziona una provincia, i grafici delle statistiche mostrati nella sezione adiacente vengono aggiornati automaticamente mostrando esclusivamente le informazioni delle ricerche effettuate con quel valore del campo *Provincia*. Ad esempio, se l'utente selezionasse la provincia di Torino, i valori mostrati nei grafici saranno calcolati a partire esclusivamente dalle ricerche in cui è stato selezionato il valore *Torino* nel campo *Provincia*.

Tra i grafici mostrati, l'unico a non subire variazioni quantitative a seguito del filtraggio per provincia è il grafico del campo *Provincia*, in quanto le statistiche mostrate sono già suddivise per i diversi valori che il campo può assumere. Visivamente, per evidenziare la

[4]Per la visualizzazione della mappa è stata usata la libreria JavaScript FusionCharts Suite XT [19] prodotta da FusionCharts. La versione della libreria utilizzata è la versione trial che include il link `www.fusioncharts.com` al fondo di ogni grafico. Nel caso di utilizzo per fini commerciali è richiesto l'acquisto di una licenza.

provincia selezionata, la barra corrispondente alla provincia viene mostrata con un colore più scuro, come mostrato in Figura 5.2b.

Statistiche Ricerca Ottimizzata

Lo scopo della modalità *Statistiche Ricerca Ottimizzata* è quello di mostrare i grafici delle statistiche relative ai parametri di ricerca che l'utente può selezionare nella sezione *Ricerca Ottimizzata*, i.e. i valori di rilevanza espressi tramite gli slider *Distanza, Rating, Tempo di attesa, Servizi RSA* e *Servizi al contorno* e i servizi inseribili nelle sottosezioni *Servizi RSA* e *Servizi al contorno*.

I grafici visualizzabili in questa modalità, mostrati in Figura 5.4, sono:

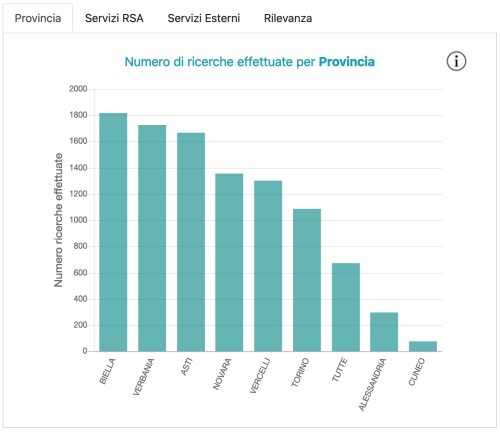
- **Provincia**: istogramma che mostra il numero di ricerche effettuate in base alla provincia selezionata dall'utente. Nonostante questo grafico non si riferisca ad un campo della *Ricerca Ottimizzata*, e sia già visualizzabile nella modalità *Statistiche Ricerca Base*, si è optato per riportarlo anche in questa modalità in quanto si è ritenuto rilevante poter visualizzare la distribuzione delle ricerche effettuate per provincia in base alle esigenze di servizi medicali espresse dagli utenti. Come nella modalità *Statistiche Ricerca Base*, quando si filtra per un valore di provincia, la barra del grafico corrispondente alla stessa viene evidenziata con un colore più scuro.

- **Servizi RSA**: istogramma che visualizza il numero di ricerche nelle quali è stato specificato un interesse per un particolare servizio interno alla struttura RSA. Sull'asse orizzontale sono riportati i nomi dei servizi mentre sull'asse verticale è riportato il numero di ricerche effettuate.

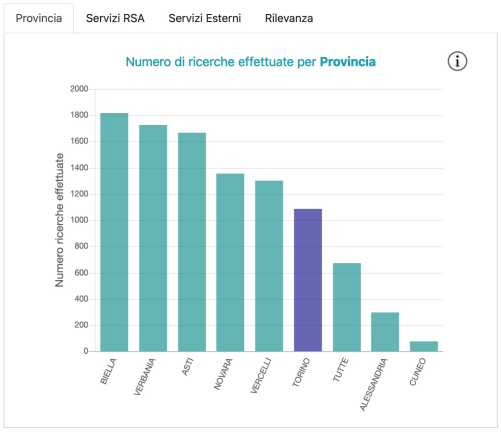
- **Servizi Esterni**: istogramma che visualizza il numero di ricerche nelle quali è stato specificato un interesse per un particolare servizio esterno alla struttura RSA. Sull'asse orizzontale sono riportati i nomi dei servizi mentre sull'asse verticale è riportato il numero di ricerche effettuate.

- **Rilevanza**: istogramma che visualizza la distribuzione di valori di rilevanza espressi dagli utenti attraverso gli slider. Poiché i valori esprimibili attraverso gli slider sono valori interi tra 0 e 100, per una maggiore leggibilità del grafico, si è deciso di raggruppare questi valori in tre categorie: *Rilevanza bassa* (valori tra 0 e 33), *Rilevanza media* (valori tra 34 e 66) e *Rilevanza alta* (valori tra 67 a 100). Per ciascuno degli slider sono quindi presenti tre barre, una per ciascuna categoria. Ognuna di esse riporta la percentuale di ricerche per cui è stato espresso un valore di rilevanza all'interno dell'intervallo. In fase di progettazione del cruscotto, per questo grafico inizialmente è stata valutata anche l'opzione di rappresentare per ciascuno degli slider la media aritmetica dei valori di rilevanza espressi attraverso di essi. Si è poi optato per la suddivisione in categorie in quanto si è ritenuto più rilevante mostrare la distribuzione dei valori invece della sola media aritmetica.

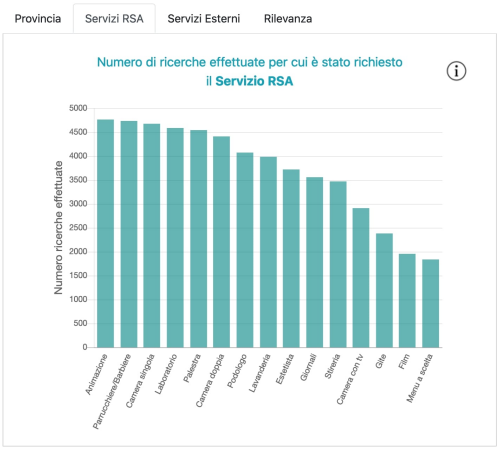
In questa modalità l'utente ha la possibilità di filtrare i dati delle statistiche in modo più approfondito rispetto alla modalità *Statistiche Ricerca Base*. In particolare, è possibile selezionare un valore di filtraggio per tutti i parametri di ricerca presenti nella sezione



(a) Statistiche del campo *Provincia*.



(b) Statistiche del campo *Provincia* a seguito della selezione di una provincia.



(c) Statistiche del campo *Servizi RSA*.

(d) Statistiche del campo *Servizi Esterni*.

(e) Statistiche del campo *Rilevanza*.

Figura 5.4: Grafici delle statistiche visualizzabili nella modalità *Statistiche Ricerca Ottimizzata*.

Ricerca Base: *Autosufficienza, Residenzialità, Presidio Alzheimer, Autosufficienza* e *Provincia*. L'utente ha quindi la possibilità di visualizzare i servizi medicali richiesti dagli utenti con particolari esigenze sanitarie. Il riquadro che consente il filtraggio dei dati in questa modalità è mostrato in Figura 5.5.

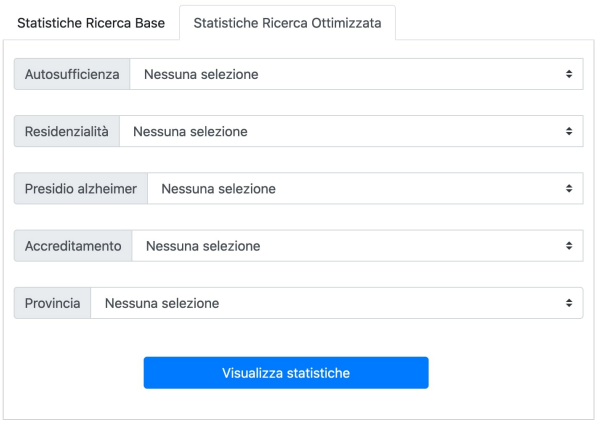


Figura 5.5: Riquadro per il filtraggio dei dati nella modalità *Statistiche Ricerca Ottimizzata*.

Il riquadro è composto da cinque menu a tendina contenenti i valori selezionabili per ciascuno dei campi presenti nella *Ricerca Base*. L'utente, selezionando un valore per uno o più di questi parametri e premendo il bottone *Visualizza statistiche* può effettuare il filtraggio dei dati. I grafici nella sezione adiacente sono aggiornati di conseguenza.

Back-End

Per l'implementazione del calcolo delle statistiche lato server è stato sviluppato il servizio `flask_statistiche` che risponde sulla porta 8103.

Il servizio è suddiviso in due parti:

- una parte dedicata al calcolo delle statistiche da visualizzare nella modalità *Statistiche Ricerca Base*;
- una parte dedicata al calcolo delle statistiche da visualizzare nella modalità *Statistiche Ricerca Ottimizzata*.

La distinzione tra queste due modalità avviene in base al parametro `tipo_statistica` ricevuto nella richiesta del client: se il valore del parametro ricevuto è `STATISTICHE_BASE` il servizio opera nella prima modalità mentre se il valore è `STATISTICHE_AVANZATE` opera nella seconda.

Di seguito vengono descritte le specifiche del servizio nelle due modalità.

Statistiche Ricerca Base

I dettagli della parte del servizio riguardante la modalità *Statistiche Ricerca Base* sono riportati in Tabella 5.1.

`flask_statistiche`	
Porta	8103
Protocollo	HTTP
Metodo	GET
Request (Parametri GET)	`tipo_statistica: String` `provincia: String`

Response (JSON)	`provincia: Object`
	`utenza: Object`
	`residenzialita: Object`
	`alzheimer: Object`
	`accreditamento: Object`

Tabella 5.1: Dettagli del servizio `flask_statistiche` (*Statistiche Ricerca Base*).

Descrizione dei parametri in ingresso (Statistiche Ricerca Base)

I parametri ricevuti in ingresso dal servizio come parametri della richiesta GET inviata dal client sono:

- **`tipo_statistica`**: parametro contenente il valore per selezionare la modalità del servizio. Nella modalità *Statistiche Ricerca Base* il valore è `STATISTICHE_BASE`.
- **`provincia`**: parametro contenente il valore di provincia, selezionato sul client attraverso la mappa o attraverso il menu a tendina, con cui filtrare i dati. Se nessuna provincia è stata selezionata il valore di questo parametro è una stringa vuota.

Descrizione dei parametri in uscita (Statistiche Ricerca Base)

La risposta del server è una risposta in formato JSON contenente i valori delle statistiche calcolate. I campi presenti nella risposta sono:

- **`provincia`**: parametro contenente le statistiche riguardanti il campo *Provincia*. I dati di questo parametro sono utilizzati lato client per popolare il grafico in Figura 5.2a.
- **`utenza`**: parametro contenente le statistiche riguardanti il campo *Autosufficienza*. I dati di questo parametro sono utilizzati lato client per popolare il grafico in Figura 5.2c.
- **`residenzialita`**: parametro contenente le statistiche riguardanti il campo *Residenzialità*. I dati di questo parametro sono utilizzati lato client per popolare il grafico in Figura 5.2d.
- **`alzheimer`**: parametro contenente le statistiche riguardanti il campo *Presidio alzheimer*. I dati di questo parametro sono utilizzati lato client per popolare il grafico in Figura 5.2e.

- **accreditamento**: parametro contenente le statistiche riguardanti il campo *Accreditamento*. I dati di questo parametro sono utilizzati lato client per popolare il grafico in Figura 5.2f.

Statistiche Ricerca Ottimizzata

I dettagli della parte del servizio riguardante la modalità *Statistiche Ricerca Ottimizzata* sono riportati in Tabella 5.2.

`flask_statistiche`	
Porta	8103
Protocollo	HTTP
Metodo	GET

Request (Parametri GET)	`tipo_statistica: String`
	`provincia: String`
	`utenza: String`
	`residenzialita: String`
	`alzheimer: String`
	`accreditamento: String`

Response (JSON)	`provincia: Object`
	`rilevanza: Object`
	`servizi_rsa: Object`
	`servizi_esterni: Object`

Tabella 5.2: Dettagli del servizio `flask_statistiche` (*Statistiche Ricerca Ottimizzata*).

Descrizione dei parametri in ingresso (Statistiche Ricerca Ottimizzata)

I parametri ricevuti in ingresso dal servizio come parametri della richiesta GET inviata dal client sono:

- **tipo_statistica**: parametro contenente il valore per selezionare la modalità del servizio. Nella modalità *Statistiche Ricerca Ottimizzata* il valore è `STATISTICHE_AVANZATE`.
- **accreditamento**: parametro contenente il valore di *Accreditamento*, selezionato sul client attraverso il menu a tendina corrispondente, con cui filtrare i dati. Se non è stato selezionato nessun valore, il valore di questo parametro è una stringa vuota.
- **alzheimer**: parametro contenente il valore di *Presidio Alzheimer*, selezionato sul client attraverso il menu a tendina corrispondente, con cui filtrare i dati. Se non è stato selezionato nessun valore, il valore di questo parametro è una stringa vuota.
- **provincia**: parametro contenente il valore di *Provincia*, selezionato sul client attraverso il menu a tendina corrispondente, con cui filtrare i dati. Se non è stato selezionato nessun valore, il valore di questo parametro è una stringa vuota.

- **`residenzialita`**: parametro contenente il valore di *Residenzialità*, selezionato sul client attraverso il menu a tendina corrispondente, con cui filtrare i dati. Se non è stato selezionato nessun valore, il valore di questo parametro è una stringa vuota.

- **`utenza`**: parametro contenente il valore di *Autosufficienza*, selezionato sul client attraverso il menu a tendina corrispondente, con cui filtrare i dati. Se non è stato selezionato nessun valore, il valore di questo parametro è una stringa vuota.

Descrizione dei parametri in uscita (Statistiche Ricerca Ottimizzata)

La risposta del server è una risposta in formato JSON contenente i valori delle statistiche calcolate. I campi presenti nella risposta sono:

- **`provincia`**: parametro contenente le statistiche riguardanti il parametro *Provincia*. I dati di questo parametro sono utilizzati lato client per popolare il grafico in Figura 5.4a.

- **`servizi_rsa`**: parametro contenente le statistiche riguardanti i servizi RSA di cui è stato espresso interessamento in fase di ricerca. I dati di questo parametro sono utilizzati lato client per popolare il grafico in Figura 5.4c.

- **`servizi_esterni`**: parametro contenente le statistiche riguardanti i servizi esterni di cui è stato espresso interessamento in fase di ricerca. I dati di questo parametro sono utilizzati lato client per popolare il grafico in Figura 5.4d.

- **`rilevanza`**: parametro contenente le statistiche riguardanti i valori di rilevanza espressi attraverso gli slider. I dati di questo parametro sono utilizzati lato client per popolare il grafico in Figura 5.4e.

Calcolo delle statistiche

Per il calcolo delle statistiche di entrambe le modalità è stata utilizzata la struttura dati DataFrame della libreria Pandas [3], nella quale sono stati importati i dati da MongoDB. In particolare, di tale struttura è stato utilizzato il metodo `value_counts()` per il conteggio dei valori di ciascuno dei parametri di cui visualizzare il grafico.

5.3 Pagina Esporta

In questa sezione viene descritto come è stata sviluppata e implementata nell'applicazione la pagina *Esporta*.

5.3.1 Descrizione

La pagina *Esporta* è stata creata con l'obiettivo di rendere accessibili i dati grezzi raccolti dall'applicazione anche all'esterno di essa, in modo da consentire sia agli operatori sanitari e sia ai gestori della sanità, nel caso ne avessero necessità, di analizzare tali dati con tecniche diverse da quelle proposte nell'applicazione. La pagina permette quindi di esportare il

contenuto della collezione `log_ricerche_RSA` all'interno di un file sul dispositivo dell'utente ed è accessibile esclusivamente alle due categorie di utenti sopra citate.

Il formato selezionato per l'esportazione dei dati è il formato CSV (acronimo di *Comma-Separated Values*).

5.3.2 Sviluppo

Di seguito viene descritto come sono stati sviluppati il front-end e il back-end della pagina *Esporta*.

Front-End

La pagina, mostrata in Figura 5.6, è composta da cinque menu a tendina contenenti i campi presenti nella sezione *Ricerca Base*: *Autosufficienza, Residenzialità, Presidio alzheimer, Accreditamento, Provincia*. I valori selezionabili per questi campi sono i medesimi di quelli dei campi di tale sezione con l'aggiunta dell'opzione *Nessuna selezione*.

Figura 5.6: Pagina *Esporta*.

Attraverso tali campi l'operatore può selezionare, se lo desidera, per quali valori filtrare il contenuto delle ricerche da esportare. Ad esempio, se l'operatore selezionasse per il campo *Presidio alzheimer* il valore *Si*, verrebbero esportate dal sistema solamente le ricerche per cui è stato selezionato quel valore nel campo omonimo della pagina *Ricerca*. Per tutti i campi in cui rimane selezionata l'opzione *Nessuna selezione* non viene invece effettuato alcun filtraggio e, lasciando selezionata questa opzione in tutti i campi, viene esportato l'intero contenuto della collezione. Alla pressione del bottone *Esporta CSV* viene infine inviata una richiesta al server con i parametri selezionati dall'operatore. Alla risposta del server viene visualizzato un prompt all'utente con cui gli viene chiesto di salvare il file. Un estratto del contenuto del file CSV ricevuto dal client è mostrato in Figura 5.7.

Back-End

Per l'implementazione di questa funzionalità lato server è stato sviluppato il servizio `flask_esporta` che risponde sulla porta 8104. I dettagli del servizio sono riportati in Tabella 5.3.

`flask_esporta`	
Porta	8104
Protocollo	HTTP
Metodo	GET

Request (Parametri GET)	`accreditamento: String`
	`alzheimer: String`
	`provincia: String`
	`residenzialita: String`
	`utenza: String`

Response (file CSV)	`log_ricerche_RSA.csv`

Tabella 5.3: Dettagli del servizio `flask_esporta`.

Descrizione dei parametri in ingresso

I parametri ricevuti in ingresso dal servizio come parametri della richiesta GET inviata dal client sono:

- **`accreditamento`**: parametro contenente il valore selezionato sul client per il campo *Accreditamento*.
- **`alzheimer`**: parametro contenente il valore selezionato sul client per il campo *Presidio alzheimer*.
- **`provincia`**: parametro contenente il valore selezionato sul client per il campo *Provincia*.
- **`residenzialita`**: parametro contenente il valore selezionato sul client per il campo *Residenzialità*.
- **`utenza`**: parametro contenente il valore selezionato sul client per il campo *Autosufficienza*.

Descrizione dei parametri in uscita

Questo servizio non presenta parametri in uscita in quanto la risposta non è in formato JSON ma consiste nel file CSV generato lato server.

Affinché la risposta sia riconosciuta dal client come un file da salvare, nell'header sono stati impostati i seguenti parametri:

- **`Content-Type`**: `"text/csv"`.
- **`Content-Disposition`**: `"attachment; filename=log_ricerche_rsa.csv"`.

Generazione del file CSV

Per la generazione del file CSV è stata utilizzata la struttura dati DataFrame della libreria Pandas [3], nella quale sono stati importati i dati da MongoDB. In particolare, di tale struttura è stato utilizzato il metodo `to_csv()` che permette di ottenere un file CSV a partire dai dati memorizzati al suo interno. Inoltre, poiché il formato CSV è un formato più adatto a contenere dati strutturati in forma tabellare rispetto ai dati semi-strutturati del formato JSON di MongoDB, si è proceduto con una normalizzazione della struttura dati prima della conversione in file, con il fine di rendere facilmente accessibili anche i dati contenuti nei documenti annidati. Tale operazione è stata realizzata usando la funzione `json_normalize()` della stessa libreria Pandas.

	A	B	C	D	E	F	G	
1	Accreditamento	Alzheimer	Utenza	Residenzialita	Provincia	Ricerca_avanzata	Servizi_RSA.Categorie	Servizi_RSA.Interessat
2	TUTTI	SI	AUTOSUFFICIENTI	RESIDENZIALE	NOVARA	FALSE		
3	TUTTI	SI	AUTOSUFFICIENTI	SEMIRESIDENZIALE	ASTI	TRUE	'Cura della persona'	'Estetista'
4	NON ACCREDITATO	SI	AUTOSUFFICIENTI	RESIDENZIALE	NOVARA	FALSE		
5	TUTTI	SI	AUTOSUFFICIENTI	RESIDENZIALE	ASTI	TRUE	'Servizi interni', 'Cura della persona', 'Carateristiche', 'Attività interne', 'Ristorazione', 'Attività fisica', 'Attività esterne'	'Stireria', 'Animazione', '
6	TUTTI	NO	AUTOSUFFICIENTI	RESIDENZIALE	ASTI	TRUE	'Attività interne', 'Servizi interni', 'Cura della persona', 'Carateristiche'	'Animazione', 'Giornali'
7	TUTTI	SI	AUTOSUFFICIENTI	RESIDENZIALE	NOVARA	FALSE		
8	TUTTI	SI	AUTOSUFFICIENTI	RESIDENZIALE	TUTTE	TRUE	'Attività fisica', 'Servizi interni', 'Cura della persona', 'Attività interne', 'Carateristiche', 'Ristorazione'	'Palestra', 'Estetista', 'C
9	TUTTI	NO	AUTOSUFFICIENTI	RESIDENZIALE	ASTI	TRUE	'Attività fisica', 'Carateristiche', 'Servizi interni', 'Attività interne', 'Cura della persona', 'Ristorazione'	'Palestra', 'Stireria', 'Film
10	ACCREDITATO	NO	AUTOSUFFICIENTI	RESIDENZIALE	TORINO	FALSE		
11	TUTTI	NO	AUTOSUFFICIENTI	RESIDENZIALE	NOVARA	TRUE	'Carateristiche', 'Attività interne', 'Servizi interni', 'Cura della persona', 'Attività esterne', 'Attività fisica'	'Camera singola', 'Giorn
12	TUTTI	SI	AUTOSUFFICIENTI	RESIDENZIALE	TUTTE	TRUE	'Attività interne', 'Carateristiche', 'Cura della persona'	'Laboratorio', 'Parrucchi
13	TUTTI	NO	AUTOSUFFICIENTI	RESIDENZIALE	VERCELLI	TRUE	'Cura della persona', 'Servizi interni', 'Carateristiche'	'Podologo'
14	TUTTI	SI	AUTOSUFFICIENTI	RESIDENZIALE	ASTI	TRUE	'Carateristiche', 'Servizi interni', 'Cura della persona', 'Attività fisica', 'Attività interne', 'Attività esterne'	'Podologo', 'Lavanderia'
15	TUTTI	NO	AUTOSUFFICIENTI	RESIDENZIALE	ASTI	TRUE	'Attività interne', 'Ristorazione', 'Attività fisica', 'Servizi interni', 'Carateristiche', 'Cura della persona'	'Laboratorio', 'Menu a s
16	TUTTI	SI	AUTOSUFFICIENTI	RESIDENZIALE	BIELLA	TRUE	'Carateristiche', 'Cura della persona'	'Camera singola', 'Parru
17	TUTTI	NO	AUTOSUFFICIENTI	RESIDENZIALE	VERCELLI	TRUE	'Servizi interni', 'Carateristiche', 'Attività interne', 'Attività esterne', 'Ristorazione'	'Stireria', 'Menu a scelta
18	ACCREDITATO	SI	AUTOSUFFICIENTI	SEMIRESIDENZIALE	ASTI	TRUE		
19	ACCREDITATO	NO	AUTOSUFFICIENTI	RESIDENZIALE	VERBANIA	TRUE	'Carateristiche', 'Servizi interni', 'Attività interne', 'Cura della persona', 'Attività esterne', 'Attività fisica', 'Ristorazione'	'Camera singola', 'Anim
20	TUTTI	NO	AUTOSUFFICIENTI	RESIDENZIALE	TORINO	TRUE	'Carateristiche', 'Attività interne', 'Servizi interni', 'Cura della persona', 'Attività fisica'	'Laboratorio', 'Estetista',
21	TUTTI	NO	AUTOSUFFICIENTI	RESIDENZIALE	BIELLA	FALSE		
22	TUTTI	NO	AUTOSUFFICIENTI	RESIDENZIALE	BIELLA	TRUE	'Carateristiche', 'Cura della persona', 'Attività interne'	'Camera doppia', 'Podol
23	ACCREDITATO	NO	AUTOSUFFICIENTI	RESIDENZIALE	ASTI	TRUE	'Cura della persona', 'Attività interne', 'Attività esterne', 'Carateristiche', 'Servizi interni'	'Podologo', 'Animazione
24	NON ACCREDITATO	SI	AUTOSUFFICIENTI	SEMIRESIDENZIALE	TUTTE	TRUE	'Servizi interni', 'Cura della persona', 'Attività interne', 'Carateristiche'	'Stireria', 'Parrucchiere/B
25	TUTTI	NO	AUTOSUFFICIENTI	RESIDENZIALE	NOVARA	TRUE		
26	ACCREDITATO	SI	AUTOSUFFICIENTI	RESIDENZIALE	VERCELLI	FALSE		
27	ACCREDITATO	NO	AUTOSUFFICIENTI	RESIDENZIALE	BIELLA	TRUE	'Cura della persona', 'Servizi interni', 'Carateristiche'	'Parrucchiere/Barbiere',
28	TUTTI	NO	AUTOSUFFICIENTI	RESIDENZIALE	VERBANIA	TRUE	'Carateristiche', 'Attività interne', 'Cura della persona'	'Estetista'
29	ACCREDITATO	SI	AUTOSUFFICIENTI	RESIDENZIALE	VERBANIA	TRUE		

Figura 5.7: Estratto del contenuto del file CSV ricevuto dal client.

Capitolo 6

Sistemi di raccomandazione

6.1 Introduzione

I sistemi di raccomandazioni sono algoritmi che hanno come obiettivo quello di identificare e suggerire un insieme di contenuti, prodotti o servizi (e.g. libri, film, ristoranti, hotel) che siano il più possibile in accordo con le preferenze o le necessità di un utente [28]. Da diversi anni i sistemi di raccomandazione giocano un ruolo centrale in ambito *e-commerce* ma sono stati impiegati anche in moltissimi altri ambiti, come ad esempio quello sanitario.

Gli attori principali all'interno di un sistema di raccomandazione sono due: gli *utenti* e gli *articoli*. Gli utenti sono le entità che interagiscono con il sistema e a cui vengono fornite le raccomandazioni, mentre gli articoli sono i prodotti da raccomandare. Il particolare utente a cui si vuole fornire una raccomandazione in un dato momento è definito *utente attivo*.

L'obiettivo principale di questi sistemi è quindi quello di predire il giudizio di un utente per quegli articoli con cui non ha ancora interagito, con il fine di raccomandare a tale utente l'insieme di articoli che sono stimati essere di maggior interesse per lui.

6.2 Categorie principali

Nei paragrafi seguenti sono descritte le principali categorie di sistemi di raccomandazioni esistenti [22], i.e. *collaborative filtering*, *content-based*, *knowledge-based* e *association rules-based*.

6.2.1 Collaborative Filtering

I sistemi di raccomandazione *collaborative filtering* si basano sulle valutazioni fornite agli articoli da parte degli utenti per generare raccomandazioni [22]. Il termine *collaborative* si riferisce proprio all'utilizzo delle valutazioni dei vari utenti in maniera collaborativa per predire le valutazioni mancanti.

All'interno di questo tipo di sistemi, ciascun utente è descritto come un vettore di valori corrispondenti alle valutazioni che ha fornito o che ha ricevuto. Tali valutazioni sono memorizzate nella cosiddetta *matrice delle valutazioni utente-articolo*. Un esempio

di matrice delle valutazioni è mostrato in Tabella 6.1. Poiché ogni utente interagisce (e.g. acquista, legge, vede) solitamente solo con una piccola parte degli articoli presenti nel sistema, tale matrice risulta spesso sparsa. L'obiettivo del *collaborative filtering* è quindi quello di predire le valutazioni mancanti considerando le valutazioni fornite da altri utenti simili. Tali predizioni vengono poi utilizzate per identificare quali articoli raccomandare o meno all'utente attivo.

	Inception	*Matrix*	*Interstellar*	*Gladiator*
Alice	5	4	?	1
Bob	?	?	2	5
Carl	5	4	5	?
Daniel	?	5	?	3

Tabella 6.1: Esempio di matrice delle valutazioni utente-articolo.

I sistemi di raccomandazioni *collaborative filtering* si suddividono in due categorie sulla base dell'approccio usato per fornire raccomandazioni:

- **Memory-Based**: l'approccio *memory-based*, anche detto *neighborhood-based*, impiega direttamente i valori delle valutazioni degli utenti senza la costruzione di modelli matematici. La predizione delle valutazioni mancanti è basata sul concetto di similarità, la quale viene applicata agli utenti (approccio *user-based*) o agli articoli (approccio *item-based*). Nell'approccio *user-based*, per fornire raccomandazioni si ricercano i k utenti più simili all'utente attivo, sulla base delle valutazioni fornite, e si predicono le valutazioni mancanti calcolando una media pesata delle valutazioni fornite dagli stessi. La similarità viene quindi calcolata tra le righe della matrice delle valutazioni per identificare gli utenti simili. Nell'approccio *item-based*, invece, per predire la valutazione dell'utente attivo per un determinato articolo di interesse si determina l'insieme S degli articoli maggiormente simili a tale articolo, sulla base delle valutazioni ricevute, e si predice la valutazione per tale articolo calcolando una media pesata delle valutazioni degli articoli simili. Nell'approccio *item-based* la similarità viene quindi calcolata tra le colonne della matrice delle valutazioni per identificare gli articoli simili. Il vantaggio principale dell'approccio *memory-based* consiste nella semplicità di implementazione degli algoritmi, rendendo anche le raccomandazioni risultanti spesso facili da spiegare. Questa categoria risulta però poco efficiente nel caso di matrici delle valutazioni particolarmente sparse.

- **Model-Based**: l'approccio *model-based* impiega invece la matrice delle valutazioni per l'addestramento di modelli predittivi tramite l'utilizzo di algoritmi di Data Mining e Machine Learning. Esempi di algoritmi utilizzati da questo approccio sono gli alberi di decisione e i modelli a fattori latenti. Contrariamente all'approccio *memory-based*, l'approccio *model-based* risulta particolarmente adatto nel caso di matrici delle valutazioni notevolmente sparse. Lo svantaggio principale di questo metodo consiste nella complessità degli algoritmi impiegati, i quali rendono spesso difficoltoso spiegare le raccomandazioni fornite.

Tra i vantaggi principali delle raccomandazioni *collaborative filtering* vi è il fatto che non richiedono alcuna informazione circa gli utenti e gli articoli e, di conseguenza, possono essere usate in molti contesti diversi, anche in quei contesti dove la descrizione di utenti e articoli può essere complessa. I sistemi di raccomandazione *collaborative filtering* sono inoltre in grado di fornire raccomandazioni fortuite, ossia raccomandazioni di articoli che sono di valore per l'utente ma che l'utente non si aspettava e che non avrebbe mai considerato altrimenti.

Lo svantaggio principale di questa categoria consiste invece nel problema del *cold-start*. Poiché questi sistemi considerano esclusivamente le interazioni passate degli utenti con gli articoli, risulta impossibile fornire raccomandazioni ai nuovi utenti in quanto, non avendo ancora interagito con alcun articolo, non possono essere confrontati con altri utenti. Inoltre anche i nuovi articoli sono affetti dallo stesso problema; fino a quando essi non verranno valutati dagli utenti, gli stessi non potranno mai essere raccomandati. Questi sistemi necessitano quindi di tempo prima che le raccomandazioni diventino sufficientemente accurate.

6.2.2 Content-Based

I sistemi di raccomandazione *content-based* si basano sugli attributi descrittivi degli articoli per la generazione delle raccomandazioni [22]. Il termine *content* si riferisce proprio alla descrizione degli articoli.

Questa categoria differisce dalla categoria *collaborative filtering* in quanto le valutazioni degli altri utenti non contribuiscono in alcun modo alle raccomandazioni per l'utente attivo. Le valutazioni dell'utente attivo sono invece utilizzate in combinazione con le descrizioni degli articoli per l'addestramento di un modello specifico per tale utente. Tale modello viene poi impiegato per predire il grado di apprezzamento dell'utente nei confronti di articoli che non ha ancora valutato o acquistato.

Uno dei vantaggi principali dei sistemi di raccomandazione *content-based* è che riescono a raccomandare i nuovi articoli anche quando non sono disponibili valutazioni da parte degli utenti. L'utente attivo potrebbe infatti aver già valutato in precedenza articoli simili e, di conseguenza, anche i nuovi articoli possono essere raccomandati allo stesso anche in assenza di valutazioni.

Tra gli svantaggi di questa categoria di sistemi di raccomandazione vi è invece il fatto che spesso forniscono raccomandazioni ovvie per l'utente, in quanto possono raccomandare solo articoli che in qualche modo risultano simili ad altri articoli con cui l'utente ha già interagito in precedenza. Inoltre, anche se questi sistemi sono efficaci nel raccomandare i nuovi articoli, non lo sono invece per fornire raccomandazioni ai nuovi utenti. Questo perché il sistema necessita comunque dello storico delle valutazioni dell'utente attivo per valutare quali siano gli articoli che potrebbero essere di interesse per lui. Per quanto riguarda i nuovi utenti, anche questa categoria è quindi affetta dal problema del *cold-start*.

6.2.3 Knowledge-Based

I sistemi di raccomandazione *knowledge-based* forniscono raccomandazioni sulla base di vincoli o di requisiti specificati interattivamente dall'utente, in combinazione con la descrizione degli articoli e la conoscenza del dominio di applicazione [22]. Il termine *knowledge*

si riferisce all'importanza che tale conoscenza del dominio ricopre in questa categoria di sistemi.

Questa categoria differisce dalle categorie *collaborative filtering* e *content-based* in quanto non utilizza né le valutazioni degli altri utenti né lo storico delle valutazioni dell'utente attivo. Questi sistemi permettono invece all'utente di specificare direttamente ciò di cui ha bisogno, dandogli quindi un maggiore controllo nel processo di raccomandazione. Le informazioni fornite dall'utente vengono poi utilizzate in combinazione alla conoscenza di dominio per ricercare gli articoli maggiormente adatti alle esigenze espresse da raccomandare.

I sistemi di raccomandazione *knowledge-based* si dividono in due categorie sulla base del tipo di interfaccia fornita all'utente per la specifica dei requisiti:

- **Constraint-Based**: nei sistemi *constraint-based* all'utente viene fornita un'interfaccia in cui specificare vincoli o requisiti sugli attributi degli articoli. Attraverso la conoscenza di dominio tali vincoli vengono poi utilizzati per la raccomandazione dell'articolo più appropriato. In base al risultato fornito dal sistema l'utente può poi ripetere tale processo, rilassando o restringendo alcuni dei vincoli specificati, in modo da raffinare ulteriormente il risultato.

- **Case-Based**: nei sistemi *case-based* all'utente viene fornita un'interfaccia in cui specificare articoli obiettivo, i.e. articoli quanto più simili al tipo di articolo desiderato, da utilizzare come punto di partenza per l'esplorazione delle raccomandazioni. Vengono quindi applicate metriche di similarità tra gli attributi di tali articoli e gli altri articoli presenti nel sistema per identificare quelli più simili da raccomandare. In base al risultato fornito dal sistema l'utente può poi ripetere tale processo indicando gli articoli raccomandati come nuovi articoli obiettivo, procedendo quindi interattivamente verso raccomandazioni sempre più affini all'articolo desiderato.

Il vantaggio principale dei sistemi *knowledge-based* consiste nel fatto che, non servendosi delle valutazioni degli utenti, essi non sono affetti dal problema del *cold-start* né per quanto riguarda i nuovi utenti né per quanto riguarda i nuovi articoli.

Lo svantaggio principale di questi sistemi consiste invece nel fatto che, dipendendo fortemente dagli attributi degli articoli, le raccomandazioni fornite possono risultare a volte ovvie per l'utente, come accade anche nel caso dei sistemi di raccomandazione *content-based*.

6.2.4 Association Rules-Based

I sistemi di raccomandazione *association rules-based* si basano sulla tecnica esplorativa di Data Mining delle regole di associazione per fornire raccomandazioni. Tale tecnica permette di estrarre relazioni nascoste tra i dati a partire da un dataset composto da una lista di transazioni [23]. Una transazione consiste in un insieme non ordinato di elementi, chiamati *item*, che appaiono o che vengono scelti insieme.

In quanto tecnica esplorativa, l'analisi tramite regole di associazione non nasce con lo scopo di fornire raccomandazioni; tuttavia le informazioni ricavate dal suo impiego possono essere sfruttate allo scopo. Una delle sue applicazioni più note è infatti la *Market Basket Analysis* che consiste nell'analizzare i carrelli della spesa con il fine di identificare

relazioni nascoste tra i prodotti che vengono acquistati dai clienti. A seguito dell'analisi, alcuni dei prodotti potrebbero risultare spesso acquistati insieme; il commerciante può quindi utilizzare tale informazione per realizzare attività di marketing mirate o gestire il posizionamento di questi prodotti sugli scaffali in modo tale da incentivarne l'acquisto ed incrementarne le vendite.

Nell'ambito dei sistemi di raccomandazione, questa tecnica differisce da altri metodi di raccomandazione, come ad esempio il *collaborative filtering*, in quanto non tiene in considerazione lo storico delle interazioni con gli *item* del singolo utente. L'obiettivo delle regole di associazione infatti non è quello di trovare possibili relazioni di preferenza per un individuo in particolare quanto piuttosto di trovare relazioni tra insiemi di elementi indipendentemente dall'utente a cui si desidera fornire raccomandazioni e considerando indistintamente tutte le transazioni di tutti gli utenti.

Matematicamente, una regola di associazione è composta da due insiemi di elementi, denominati *antecedente* e *conseguente*, ed esprime una relazione di co-occorrenza tra questi due insiemi. L'insieme congiunto degli elementi presenti nell'antecedente e nel conseguente prende il nome di *itemset*. Ad esempio, considerando un *itemset* composto da due insiemi X e Y, una possibile regola di associazione tra questi insiemi è indicata dalla formula seguente:

$$X \Rightarrow Y$$

nella quale X costituisce l'antecedente mentre Y il conseguente.

Questa regola indica che quando in una transazione sono presenti gli elementi di X, con una certa probabilità anche gli elementi di Y saranno presenti all'interno della stessa.

Per valutare statisticamente la qualità di una regola di associazione si usano principalmente le seguenti misure statistiche: *supporto*, *confidenza* e *lift*. Definendo l'insieme $T = \{t_1, t_2, ..., t_N\}$ delle transazioni e definendo l'operatore $\sigma(X) = |\{t_i | X \subseteq t_i, t_i \in T\}|$, le tre misure sono definite nel modo seguente [23]:

- **Supporto**: è un'indicazione della frequenza con cui un determinato *itemset* compare all'interno delle transazioni della base dati. Questo valore aiuta ad identificare quali regole possono contenere informazioni statisticamente rilevanti. In fase di analisi infatti si considerano solo le regole basate su *itemset* con valore di supporto superiore ad una soglia minima. Matematicamente, il supporto di un *itemset* è la frazione di transazioni che lo contengono rispetto al numero totale di transazioni:

$$support(X \cup Y) = \frac{\sigma(X \cup Y)}{|T|} \tag{6.1}$$

- **Confidenza**: è un'indicazione di quanto spesso la relazione indicata dalla regola di associazione è stata riscontrata effettivamente nelle transazioni della base dati. Matematicamente, la confidenza corrisponde alla probabilità condizionata di occorrenza del conseguente data la presenza dell'antecedente:

$$confidence(X \Rightarrow Y) = \frac{support(X \cup Y)}{support(X)} \tag{6.2}$$

- **Lift**: è una misura di correlazione tra la presenza dell'antecedente in una transazione e la probabilità di co-occorrenza del conseguente nella stessa. Essa fornisce un'indicazione di quanto spesso antecedente e conseguente appaiano insieme rispetto al caso di indipendenza statistica. Matematicamente, il lift è il rapporto tra la confidenza della regola ed il supporto del conseguente:

$$lift(X \Rightarrow Y) = \frac{support(X \cup Y)}{support(X) \cdot support(Y)} \tag{6.3}$$

  Il suo valore è interpretabile come segue:

$$lift(X \Rightarrow Y) = \begin{cases} = 1 & \text{se } X \text{ e } Y \text{ non sono correlati} \\ > 1 & \text{se } X \text{ e } Y \text{ sono correlati positivamente} \\ < 1 & \text{se } X \text{ e } Y \text{ sono correlati negativamente} \end{cases}$$

Attraverso una selezione appropriata dei valori di queste metriche è possibile estrarre da un dataset transazionale le regole di associazione statisticamente più rilevanti. Conoscendo quali sono gli articoli di interesse per l'utente è poi possibile ricercare tra le regole estratte quelle aventi nell'antecedente tali articoli e raccomandare all'utente gli articoli presenti nel conseguente delle stesse.

Il vantaggio principale dei sistemi di raccomandazione *association rules-based* consiste nel fatto che non richiedono la creazione di un profilo utente per poter fornire raccomandazioni. È sufficiente infatti che l'utente interagisca con alcuni degli articoli presenti nel sistema, ad esempio inserendo dei prodotti nel carrello di un sito *e-commerce*, per poter raccomandare allo stesso gli articoli correlati. Questa categoria di sistemi di raccomandazione non è quindi affetta dal problema del *cold-start* in riferimento ai nuovi utenti.

Lo svantaggio principale che comporta questa categoria consiste invece nel fatto che, non fornendo raccomandazioni personalizzate, le raccomandazioni risultanti possono risultare ovvie per l'utente.

Capitolo 7

Progettazione e sviluppo del sistema di raccomandazione

In questo capitolo viene descritta la fase di progettazione e di sviluppo del sistema di raccomandazione proposto.

Il capitolo è suddiviso in quattro parti:

- nella prima parte vengono riportate le valutazioni di applicabilità al contesto in analisi delle categorie principali dei sistemi di raccomandazione, con il fine di identificare la categoria più appropriata al raggiungimento dell'obiettivo prefissato, i.e. coadiuvare l'utente finale verso la struttura RSA maggiormente adatta alle sue esigenze, sfruttando le informazioni derivanti dalle ricerche utenti passate. A seguito di tali valutazioni e constatato che, con i dati a disposizione, la raccomandazione diretta di una strutture RSA non sembrerebbe possibile, è stato quindi valutato lo sviluppo di un sistema di raccomandazione per coadiuvare l'utente, durante la fase di ricerca stessa, nella selezione dei parametri di ricerca più appropriati da utilizzare;

- nella seconda parte viene descritta la soluzione proposta per la realizzazione di tale sistema di raccomandazione;

- nella terza parte viene descritta nel dettaglio la fase di progettazione del sistema di raccomandazione proposto;

- nella quarta parte viene descritta nel dettaglio la fase di sviluppo e di implementazione dello stesso all'interno dell'applicazione.

7.1 Valutazioni di applicabilità

In questa sezione vengono descritte le valutazioni di applicabilità al contesto in analisi delle principali categorie di sistemi di raccomandazione presentate nel Capitolo 6, con l'obiettivo di identificare l'approccio più consono per coadiuvare l'utente finale verso la struttura sanitaria più adatta alle sue esigenze.

Per quanto riguarda i sistemi di raccomandazione *collaborative filtering*, essi si basano sulle valutazioni degli utenti per fornire raccomandazioni. Nell'applicazione in oggetto, però, le valutazioni circa le strutture RSA selezionate non sono disponibili. L'applicazione infatti consente esclusivamente di esplorare le strutture presenti sul territorio, senza poi permettere di indicare la struttura effettivamente selezionata e, successivamente, di fornirne una valutazione. In mancanza di queste informazioni, un'applicazione tradizionale dell'approccio *collaborative filtering* non sembrerebbe possibile.

In ambito sanitario sono state però proposte alcune soluzioni alternative di sistemi di raccomandazione *collaborative filtering* che non necessitano di valutazioni da parte degli utenti, come quelle proposti da Kulev *et al.* [27] e Han *et al.* [29]. Nel sistema proposto da Kulev *et al.* gli utenti sono confrontati sulla base dei parametri di salute; questo sembrerebbe un approccio adatto per l'applicazione in oggetto in quanto sarebbe possibile caratterizzare un utente sulla base dei parametri di ricerca base selezionati in fase di ricerca. Le raccomandazioni fornite dal loro sistema si basano però poi su dati che evidenziano un qualche miglioramento nelle condizioni di salute; nell'applicazione in oggetto invece non è disponibile alcuna relazione tra una ricerca e il grado di apprezzamento verso la struttura finale scelta dall'utente. Han *et al.* invece, per ovviare alla mancanza di feedback espliciti da parte degli utenti, hanno sfruttato i dati sulle visite ripetute verso lo stesso medico per dedurne il grado di apprezzamento. Questo sistema non sembrerebbe però applicabile all'applicazione in oggetto per due ragioni: come menzionato in precedenza non sono disponibili riferimenti alla struttura finale selezionata dall'utente e, anche nel caso questa informazione fosse stata disponibile, la possibilità che un paziente possa trovarsi a visitare più volte la stessa struttura RSA sembrerebbe piuttosto rara, in quanto le strutture di questo genere sono utilizzate solo in situazioni con esigenze sanitarie molto specifiche. Nonostante non si basino su valutazioni esplicite degli utenti, anche i due approcci citati non sembrerebbero quindi comunque applicabili al contesto in analisi.

I sistemi di raccomandazione *content-based* si basano invece sulla descrizione degli articoli e sullo storico delle interazioni dell'utente con essi. Per quanto riguarda la descrizione degli articoli, nel contesto in analisi è disponibile una descrizione delle strutture RSA derivabile dalle informazioni circa i servizi medicali e non medicali offerti dalle stesse. Per quanto riguarda lo storico delle interazioni dell'utente con le strutture, valgono invece le stesse considerazioni fatte per i sistemi di raccomandazione *collaborative filtering*: non sono infatti disponibili né le valutazioni degli utenti per le strutture, né è possibile sapere le strutture selezionate dagli stessi. In mancanza di queste informazioni non sembrerebbe quindi possibile costruire un modello specifico per un utente per la generazione di raccomandazioni. Inoltre, anche nel caso queste informazioni fossero state disponibili, un sistema di raccomandazione *content-based* probabilmente non sarebbe stato un sistema adatto al contesto: il fatto di poter raccomandare una struttura RSA sulla base delle strutture RSA apprezzate in precedenza dall'utente potrebbe infatti non essere significativo. In momenti temporali diversi, l'utente potrebbe avere infatti esigenze di servizi medicali diverse e quindi, una struttura perfettamente adatta per un certo tipo di problematica potrebbe risultare invece poco adatta per un altro tipo di problematica.

I sistemi di raccomandazione *knowledge-based* si basano sulle informazioni specificate dall'utente per fornire raccomandazioni. Nonostante questi sistemi non utilizzino le valutazioni degli utenti, risolvendo quindi il problema di tale mancanza, essi non sono stati

tenuti in considerazione nel presente lavoro di tesi in quanto un'istanza di questa categoria di sistemi è già presente nell'applicazione, come descritto nel Paragrafo 2.1, la quale consiste nella modalità di ricerca ottimizzata. L'obiettivo di questo lavoro di tesi è infatti quello di andare oltre la conoscenza del singolo utente e di utilizzare la conoscenza di tutti gli utenti che interagiscono con il sistema per fornire raccomandazioni sempre più raffinate.

I sistemi di raccomandazione *association rules-based* si basano sull'estrazione di regole di associazione a partire da un dataset transazionale. Una possibile applicazione al contesto in analisi di questa categoria di sistemi potrebbe essere quella di considerare come transazioni i parametri di ricerca selezionati dagli utenti e, attraverso la creazione di *pseudo-item* [22], di correlare gli stessi con il grado di apprezzamento verso la struttura RSA selezionata. Un esempio di regola di associazione estratta con questo approccio potrebbe essere:

$$(\textit{Alzheimer=Si, Utenza=Non autosufficienti}) \Rightarrow (\textit{Struttura=NomeStruttura, Rating=Like})$$

Come per gli altri sistemi descritti in precedenza, l'applicazione di questo approccio necessita però di una qualche correlazione tra la ricerca effettuata dall'utente e la struttura poi effettivamente selezionata. In mancanza di tale informazione, anche questo approccio non sembrerebbe quindi applicabile.

Sulla base delle valutazioni effettuate si è quindi concluso che, in mancanza di una qualche indicazione da parte degli utenti circa la struttura selezionata e circa il grado di apprezzamento della stessa, la raccomandazione diretta di una struttura RSA non sembrerebbe possibile. Si è quindi deciso di spostare l'obiettivo del sistema di raccomandazione dal raccomandare direttamente una struttura RSA al supportare l'utente durante la stessa fase di ricerca nella selezione dei parametri di ricerca più consoni da utilizzare. Tale approccio risulta applicabile al contesto in analisi in quanto non richiede alcuna informazione sulle strutture RSA visitate e si focalizza esclusivamente sui parametri di ricerca selezionati dagli utenti. Per la realizzazione di tale sistema, tra gli approcci valutati, sono stati selezionati gli approcci *collaborative filtering* e *association rules-based* che consentono di sfruttare la conoscenza condivisa degli utenti, sulla base delle passate ricerche memorizzate, per la generazione di raccomandazioni. La soluzione proposta è descritta nel dettaglio nella sezione seguente.

7.2 Soluzione proposta

La soluzione proposta consiste in un sistema di raccomandazione ibrido per supportare l'utente nella scelta dei parametri di ricerca ottimizzata da impiegare all'interno della sezione corrispondente della pagina *Ricerca*.

Il sistema è composto da due componenti:

- una componente basata sull'approccio *collaborative filtering* per la raccomandazione di una configurazione iniziale dei valori di rilevanza, dei servizi RSA e dei servizi al contorno;
- una componente basata sull'approccio *association rules-based* per la raccomandazione progressiva dei servizi RSA e dei servizi al contorno richiesti spesso congiuntamente.

Di seguito sono descritte nel dettaglio le due componenti.

Raccomandazioni collaborative

La componente del sistema di raccomandazione basata sul *collaborative filtering* riceve in ingresso la configurazione dei parametri di ricerca base selezionati dall'utente durante la fase di ricerca (i.e. *Autosufficienza, Residenzialità, Presidio Alzheimer, Accreditamento* e *Provincia*) e produce in uscita una configurazione iniziale di parametri di ricerca ottimizzata consigliati (i.e. i valori di rilevanza selezionabili tramite gli slider *Distanza, Rating, Tempo di attesa, Servizi RSA e Servizi al contorno* e i servizi selezionabili nelle sottosezioni *Servizi RSA* e *Servizi al contorno*). Una rappresentazione di questa componente del sistema di raccomandazione è mostrata in Figura 7.1.

In particolare, per la generazione di raccomandazioni questa componente si basa sull'approccio *neighborhood-based* considerando il profilo di ciascun utente sulla base dei parametri di ricerca base selezionati. Il profilo dell'utente (o, in questo contesto, la sua configurazione di ricerca base) è quindi confrontato con le configurazioni di ricerca base utilizzate nelle ricerche passate dagli altri utenti per selezionare, tra le ricerche memorizzate, le k ricerche più affini alle esigenze di servizi medicali espresse dall'utente attivo. L'algoritmo utilizzato per la selezione delle ricerche affini è l'algoritmo *k-nearest neighbors*. Per consentire a tale algoritmo di calcolare la similarità tra le ricerche, i parametri di ricerca base sono prima codificati tramite *one-hot encoding* e successivamente confrontati utilizzando una versione pesata della metrica della *similarità del coseno*. Le raccomandazioni generate sono poi composte da una media pesata dei valori di rilevanza espressi all'interno delle ricerche affini e da una classifica pesata dei servizi maggiormente richiesti all'interno delle stesse.

La fase di progettazione di questa componente del sistema di raccomandazione è descritta nel dettaglio nel Paragrafo 7.3.1, mentre la fase di sviluppo e di implementazione della stessa all'interno dell'applicazione è descritta nel Paragrafo 7.4.1.

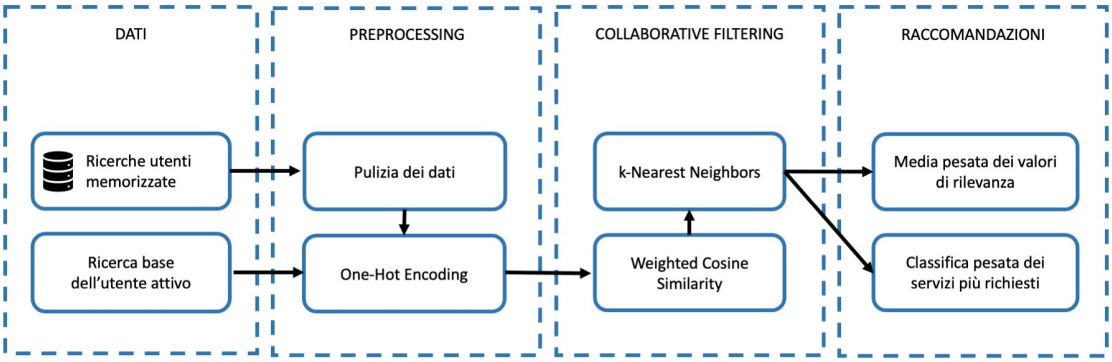


Figura 7.1: Rappresentazione della componente *collaborative filtering* del sistema di raccomandazione.

Raccomandazioni mediante regole di associazione

La componente *association rules-based* del sistema di raccomandazione si pone come un'estensione progressiva delle raccomandazioni di servizi RSA e di servizi al contorno fornite

inizialmente dalla componente collaborativa. In particolare, le raccomandazioni di questa componente vengono fornite ogniqualvolta l'utente specifica l'interessamento per un servizio nelle sottosezioni *Servizi RSA* e *Servizi al contorno*; il sistema genera quindi una nuova classifica di servizi da raccomandare, ricercando quei servizi che, nelle ricerche utenti passate, sono stati richiesti spesso congiuntamente a tale servizio. La componente è unica per entrambe le sottosezioni e, a seconda della sottosezione con cui l'utente attivo sta interagendo in un dato momento, fornisce raccomandazioni di servizi RSA o di servizi al contorno rispettivamente. Una rappresentazione di questa componente del sistema di raccomandazione è mostrata in Figura 7.2.

Questa componente riceve in ingresso l'insieme dei servizi inseriti dall'utente nella sottosezione *Servizi RSA*, o rispettivamente *Servizi al contorno*, congiuntamente alla configurazione dei parametri di ricerca base selezionati durante la fase di ricerca e produce in uscita una lista di servizi raccomandati in ordine di rilevanza. In particolare, la generazione delle raccomandazioni è basata sulle informazioni derivanti dall'estrazione di regole di associazione, le quali sono estratte a partire dai servizi richiesti congiuntamente nelle ricerche utenti passate. Per consentire tale operazione, questa componente prevede una prima fase di preprocessing del dataset per la trasformazione dello stesso in formato transazionale: prima dalle ricerche sono rimossi tutti gli attributi non riguardanti i servizi in oggetto e successivamente le stesse sono convertite in liste di servizi. Inoltre, per garantire una maggiore affinità dei servizi raccomandati con le esigenze di servizi medicali richiesti dall'utente, l'estrazione delle regole avviene esclusivamente a partire dalle ricerche memorizzate aventi la stessa configurazione di ricerca base selezionata dall'utente. Tutte le ricerche con una configurazione di parametri diversa sono quindi rimosse dal dataset prima dell'estrazione. L'estrazione delle regole di associazione è poi effettuata in tempo reale utilizzando l'algoritmo FP-growth con una configurazione predefinita dei valori di supporto e confidenza. Le raccomandazioni generate sono poi composte dai servizi presenti nel conseguente delle regole estratte, ordinati in ordine decrescente in base al valore di confidenza.

La fase di progettazione di questa componente del sistema di raccomandazione è descritta nel dettaglio nel Paragrafo 7.3.2, mentre la fase di sviluppo e di implementazione della stessa all'interno dell'applicazione è descritta nel Paragrafo 7.4.2.

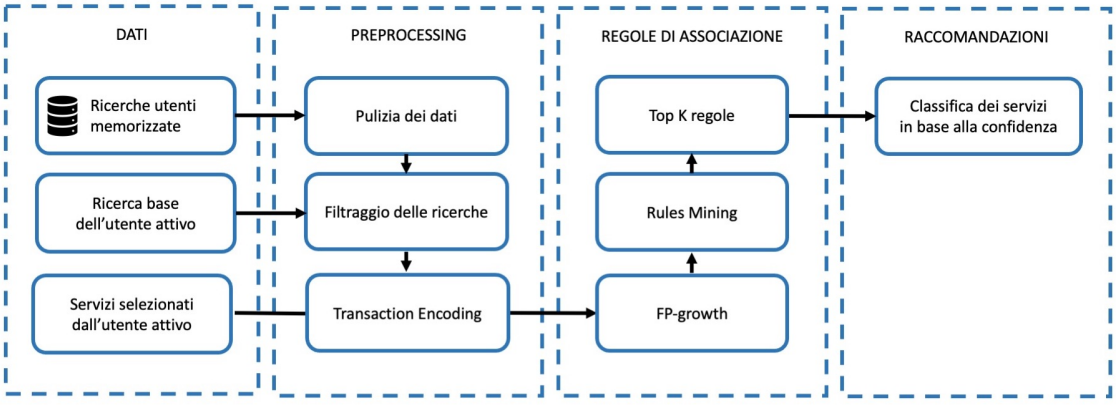


Figura 7.2: Rappresentazione della componente *association rules-based* del sistema di raccomandazione.

7.3 Progettazione del sistema

In questa sezione viene descritto come sono state progettate le due componenti del sistema di raccomandazione proposto. La sezione è quindi suddivisa in due parti: la prima parte è dedicata alla descrizione della progettazione della componente basata sul *collaborative filtering*, mentre la seconda parte è dedicata alla descrizione della progettazione della componente *association rules-based*.

7.3.1 Raccomandazioni collaborative

Di seguito sono descritte le fasi della progettazione della componente del sistema di raccomandazione basata sul *collaborative filtering*: selezione del metodo di raccomandazione, preprocessing dei dati, selezione della metrica di similarità, selezione delle ricerche *neighbors* e calcolo delle raccomandazioni.

Metodo di raccomandazione

Il primo passo nella progettazione della componente collaborativa è consistito nella selezione del metodo di raccomandazione *collaborative filtering* da utilizzare tra *neighborhood-based* e *model-based*.

Per poter selezionare il metodo più appropriato al contesto è stato necessario definire quali fossero i dati da considerare per la definizione del profilo di un utente, essenziale nell'approccio *collaborative filtering*. Tradizionalmente, come descritto nel Paragrafo 6.2.1, gli utenti sono descritti sulla base delle valutazioni fornite agli articoli. Non essendo disponibili queste valutazioni, si è quindi deciso di definire il profilo dell'utente sulla base delle condizioni di salute specificate dall'utente attraverso la richiesta dei servizi medicali indicati durante la fase di ricerca. In particolare, il profilo è definito dai cinque parametri di ricerca base selezionati dall'utente nella sezione *Ricerca Base*, i.e. *Autosufficienza, Residenzialità, Presidio Alzheimer, Accreditamento* e *Provincia*. Definito il profilo dell'utente si è quindi proceduto con la selezione dell'approccio da utilizzare.

Come descritto nel Paragrafo 6.2.1, l'approccio *neighborhood-based* è un approccio semplice e accurato nel fornire raccomandazioni. La sua accuratezza è dovuta al fatto che l'intero dataset è tenuto in considerazione per il calcolo delle raccomandazioni. Questa caratteristica, però, è anche causa di un degrado delle prestazioni dell'algoritmo all'aumentare delle dimensioni e della sparsità della matrice delle valutazioni. L'approccio *model-based*, d'altra parte, grazie alla costruzione di un modello predittivo risulta efficiente nel caso di una matrice sparsa e di dimensioni elevate; tale approccio richiede però un determinato tempo prima del suo utilizzo, necessario per l'addestramento del modello, e comporta una maggiore complessità degli algoritmi da utilizzare, rendendo anche le raccomandazioni risultanti difficili da spiegare.

Considerando che, nel contesto in analisi, la matrice su cui vengono applicati questi algoritmi è di dimensioni ridotte (in quanto contenente solamente i parametri di ricerca base selezionati dagli utenti), si è optato per l'approccio *neighborhood-based* che garantisce accuratezza e semplicità di implementazione, consentendo allo stesso tempo di spiegare con facilità le raccomandazioni risultanti.

Preprocessing dei dati

Il preprocessing dei dati è una delle fasi principali in molte delle applicazioni di Data Mining e Machine Learning e consiste in una fase di preparazione del *dataset*, i.e. l'insieme dei dati utilizzati, al fine di renderlo adatto ad essere fornito in ingresso all'algoritmo da utilizzare [23]. Questa fase, a seconda del caso specifico, può essere necessaria e consistere di molteplici operazioni o non essere necessaria, nel caso in cui i dati siano già in un formato adeguato. Per la componente collaborativa sono state valutate le seguenti operazioni di preprocessing: pulizia dei dati, codifica degli attributi e normalizzazione degli attributi.

Pulizia dei dati

L'operazione di pulizia dei dati consiste nel ricercare all'interno del dataset se sono presenti ricerche memorizzate con dati mancanti o errati con il fine di eliminarli o di ripulirli. Gli attributi su cui è stato effettuato il controllo sono quelli considerati dall'algoritmo *k-nearest neighbors* per il calcolo della similarità e il calcolo delle raccomandazioni: `Provincia`, `Residenzialita`, `Alzheimer`, `Accreditamento` e `Utenza` per il calcolo della similarità e `Rilevanza`, `Servizi_RSA` e `Servizi_contorno` per il calcolo delle raccomandazioni.

Tra gli attributi per il calcolo della similarità, all'interno del dataset sintetico utilizzato in questo lavoro di tesi, non sono presenti dati errati o incompleti. Si assume inoltre che la stessa situazione si verifichi con i dati di ricerche utenti reali, in quanto tali attributi sono gli attributi corrispondenti ai campi della sezione *Ricerca Base*, i quali sono inseriti obbligatoriamente dall'utente in fase di ricerca e selezionati tra un insieme predefinito di opzioni possibili. Per questi attributi non sono quindi necessarie operazioni di pulizia.

Gli attributi per il calcolo delle raccomandazioni presentano invece dei dati mancanti, in quanto tali attributi sono presenti esclusivamente nelle ricerche corrispondenti a ricerche ottimizzate. Poiché l'obiettivo dell'algoritmo è quello di raccomandare i parametri di ricerca avanzata, non sarebbe significativo considerare ricerche che non contengono tali parametri nel calcolo delle raccomandazioni. Tutte le ricerche corrispondenti a ricerche base sono quindi state filtrate dal dataset e sono state mantenute le ricerche corrispondenti a ricerche ottimizzate. Nelle ricerche rimaste all'interno del dataset sintetico, gli attributi considerati non presentano dati mancanti o incompleti; si assume inoltre che la stessa situazione si verifichi con i dati di ricerche utenti reali per le stesse considerazioni fatte in precedenza con gli attributi per il calcolo della similarità.

Codifica degli attributi

La codifica degli attributi consiste nel modificare la tipologia di dato di alcuni attributi con il fine di renderli adatti ad essere forniti in ingresso all'algoritmo da utilizzare. Nella componente collaborativa del sistema di raccomandazione proposto, questa operazione risulta necessaria in quanto l'algoritmo *k-nearest neighbors* richiede che tutti i dati in ingresso siano di tipo numerico per il calcolo della similarità, mentre gli attributi su cui viene effettuato tale calcolo, i.e. quelli della sezione *Ricerca Base*, sono di tipo categorico, come mostrato in Tabella 7.1a.

L'approccio selezionato per la codifica dei dati è la codifica *one-hot encoding* che è l'approccio maggiormente utilizzato per la conversione di dati categorici in dati numerici [30]. Tale approccio consiste nel convertire ciascun campo categorico in un vettore binario

n-dimensionale, in cui gli attributi binari rappresentano gli n valori categorici assumibili dal campo. Così, ad esempio, i valori del campo `Accreditamento` si potrebbero convertire in (1,0,0), (0,1,0) e (0,0,1). Come alternativa alla codifica *one-hot encoding* è stata valutata la codifica *label encoding* che consiste nell'assegnare a ciascun valore di un campo categorico un valore numerico. Così, ad esempio, ai valori del campo `Accreditamento` si potrebbero assegnare i seguenti valori numerici: `TUTTI` = 0, `ACCREDITATO` = 1, `NON ACCREDITATO` = 2. Nella Tabella 7.1b e nella Tabella 7.1c sono mostrate, come esempio, le ricerche della Tabella 7.1a codificate con i due approcci valutati.

A seguito del confronto tra i due approcci, la codifica *label encoding* è stata ritenuta meno adatta al contesto in quanto potrebbe assegnare impropriamente un ordine di grandezza ad attributi che in origine non l'avevano. I valori numerici assegnati in precedenza, ad esempio, potrebbero portare l'algoritmo a considerare che `TUTTI` < `ACCREDITATO` < `NON ACCREDITATO`, inducendolo a conclusioni errate. Poiché tutti i campi categorici della *Ricerca Base* non hanno un ordinamento, questo approccio non è stato ritenuto adatto. Con la codifica *one-hot encoding*, invece, il problema dell'ordinamento non si verifica. Lo svantaggio principale di questo approccio è però quello di aumentare, anche considerevolmente, il numero di attributi. Nel caso in analisi, il numero di attributi incrementa da 5 a 18, come mostrato nella Tabella 7.1c. Poiché l'incremento è comunque contenuto, si è ritenuto questo metodo più appropriato per il sistema di raccomandazione proposto.

Normalizzazione degli attributi

Il processo di normalizzazione [23] consiste nell'applicare una trasformazione agli attributi numerici con il fine di portarli sulla medesima scala di valori. Questa operazione è spesso necessaria nella preparazione dei dati per gli algoritmi basati sul calcolo di distanze, come ad esempio il *k-nearest neighbors*, affinché nessuno degli attributi domini gli altri sulla base dell'ordine di grandezza dei valori assunti. Le trasformazioni applicate generalmente consistono nel portare i valori di tutti gli attributi nell'intervallo tra 0 e 1 o nell'intervallo tra -1 e +1 con media nulla e varianza unitaria. Poiché tutti gli attributi su cui viene effettuato il calcolo della similarità dall'algoritmo *k-nearest neighbors*, a seguito della codifica *one-hot encoding*, assumono già valori nell'intervallo tra 0 e 1, questo processo non è stato ritenuto necessario e non è stato applicato.

Metrica di similarità

La metrica di similarità utilizzata per calcolare l'affinità tra gli utenti è uno dei parametri determinanti nella progettazione di un sistema di raccomandazione *neighborhood-based* [32]. Metriche diverse portano infatti a valori di similarità diversi e, di conseguenza, a raccomandazioni diverse. La scelta della metrica più adatta dipende fortemente dal contesto in cui essa è applicata.

Nei sistemi *neighborhood-based*, la *similarità del coseno* e il *coefficiente di correlazione di Pearson* sono le metriche utilizzate tradizionalmente [31]. Queste metriche sono però generalmente utilizzate in contesti in cui i dati non sono di tipo binario, come nel caso in cui gli utenti sono rappresentati in base alle valutazioni numeriche che hanno fornito agli articoli. Tra le metriche proposte nel contesto di dati di tipo binario ci sono invece il *coefficiente Simple Matching* e il *coefficiente Jaccard*.

	Utenza	Residenzialita	Accreditamento	Alzheimer	Provincia
R_0	NON AUTOSUFFICIENTI	SEMIRESIDENZIALE	ACCREDITATO	NO	TORINO
R_1	AUTOSUFFICIENTI	SEMIRESIDENZIALE	ACCREDITATO	NO	TORINO
R_2	AUTOSUFFICIENTI	RESIDENZIALE	ACCREDITATO	NO	TORINO
R_3	AUTOSUFFICIENTI	RESIDENZIALE	NON ACCREDITATO	NO	TORINO
R_4	AUTOSUFFICIENTI	RESIDENZIALE	NON ACCREDITATO	SI	TORINO
R_5	AUTOSUFFICIENTI	RESIDENZIALE	NON ACCREDITATO	SI	CUNEO

(a) Ricerche con attributi categorici.

	Utenza	Residenzialita	Accreditamento	Alzheimer	Provincia
R_0	0	0	1	0	5
R_1	1	0	1	0	5
R_2	1	1	1	0	5
R_3	1	1	2	0	5
R_4	1	1	2	1	5
R_5	1	1	2	1	3

(b) Ricerche con attributi codificati tramite *label encoding*.

	U_A	U_NA	R_R	R_S	AC_AC	AC_NA	AC_T	A_N	A_S	P_AL	P_AS	P_B	P_C	P_N	P_TO	P_TU	P_VERB	P_VERC
R_0	0	1	0	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0
R_1	1	0	0	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0
R_2	1	0	1	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0
R_3	1	0	1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0
R_4	1	0	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0
R_5	1	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0

(c) Ricerche con attributi codificati tramite *one-hot encoding*.

Tabella 7.1: Confronto tra codifiche di dati.

Tra le due metriche utilizzate tradizionalmente, la *similarità del coseno* sembrerebbe però anche utilizzabile in un contesto di dati binari come quello in analisi. Spertus *et al.* [33], ad esempio, hanno condotto uno studio per confrontare le prestazioni di un sistema di raccomandazione collaborativo nel contesto di un social network, applicando sei metriche di similarità differenti. Tra le metriche, hanno utilizzato la *similarità del coseno* mentre hanno scartato il *coefficiente di correlazione di Pearson* in quanto considerato generalmente inappropriato per dati binari. Anche se i risultati potrebbero essere rilevanti per le particolari caratteristiche con cui è stata condotta l'analisi, è interessante notare che la metrica che ha prodotto i risultati migliori è la *similarità del coseno*.

Le metriche che sono quindi state valutate per il sistema di raccomandazione proposto sono il *coefficiente Simple Matching*, il *coefficiente Jaccard* e la *similarità del coseno*. Le tre metriche sono descritte di seguito [23]:

- **Coefficiente Simple Matching (SMC)**: questo coefficiente è una misura statistica utilizzata per calcolare la similarità tra due vettori binari, contenenti quindi esclusivamente i valori 0 o 1 per ciascuno degli attributi. Matematicamente esso consiste nel rapporto tra il numero di attributi in comune tra i due vettori e il numero totale di attributi che li compongono:

$$SMC = \frac{n_{1,1} + n_{0,0}}{n_{1,1} + n_{0,1} + n_{1,0} + n_{0,0}} \tag{7.1}$$

  in cui:

  - $n_{0,0}$: numero di attributi in cui entrambi i vettori presentano il valore 0;
  - $n_{0,1}$: numero di attributi in cui il primo vettore presenta il valore 0 mentre il secondo vettore presenta il valore 1;
  - $n_{1,0}$: numero di attributi in cui il primo vettore presenta il valore 1 mentre il secondo vettore presenta il valore 0;
  - $n_{1,1}$: numero di attributi in cui entrambi i vettori presentano il valore 1.

  Il valore di questo coefficiente ricade sempre tra 0 e 1, dove lo 0 indica una dissimilarità totale tra i due vettori mentre l'1 indica che i due vettori sono identici. L'implementazione e l'interpretazione di questo coefficiente sono molto semplici ma, affinché il suo valore sia significativo, gli attributi che compongono i vettori su cui viene calcolato devono essere attributi binari simmetrici, i cui valori di 0 e 1 rappresentano stati ugualmente significanti. Un esempio di attributo binario simmetrico potrebbe essere un attributo rappresentante il genere maschile o femminile di una persona, indicati dai valori 0 e 1, in cui entrambi gli stati definiscono una determinata caratteristica.

- **Coefficiente Jaccard**: questo coefficiente è una misura statistica che permette di valutare la similarità tra due insiemi finiti di elementi. Dati due insiemi A e B, il *coefficiente Jaccard* è definito come il rapporto tra la cardinalità dell'intersezione dei due insiemi e la cardinalità dell'unione dei due insiemi:

$$J = \frac{|A \cap B|}{|A \cup B|} \tag{7.2}$$

Il valore di questo coefficiente ricade sempre tra 0 e 1. Nel caso in cui gli insiemi siano rappresentati da due vettori binari, il coefficiente è riformulato nel modo seguente:

$$J = \frac{n_{1,1}}{n_{1,1} + n_{0,1} + n_{1,0}} \tag{7.3}$$

dove $n_{1,1}$, $n_{0,1}$ e $n_{1,0}$ assumono gli stessi valori indicati precedentemente per il *coefficiente Simple Matching*. Quando utilizzato tra vettori binari, questo coefficiente risulta essere molto simile al *coefficiente Simple Matching*. La differenza principale è costituita dal termine $n_{0,0}$ che nel *coefficiente Simple Matching* è preso in considerazione mentre nel *coefficiente Jaccard* è assente. Questo rende il *coefficiente Jaccard* appropriato per calcolare la similarità tra vettori binari asimmetrici.

- **Similarità del coseno**: questa misura consente di calcolare la similarità tra due vettori composti da attributi di tipo numerico reale. È calcolata determinando il coseno dell'angolo tra i due vettori, il quale permette di identificare se i due vettori sono diretti o meno nella stessa direzione. Matematicamente, il coseno tra due vettori è definito come il rapporto tra il prodotto scalare e il prodotto delle norme quadrate dei vettori:

$$\cos(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{||\boldsymbol{u}||\ ||\boldsymbol{v}||} \tag{7.4}$$

  Il valore di questa metrica è sempre compresa nell'intervallo -1 e +1: se l'angolo tra i due vettori è di 0° allora il coseno assume valore 1; se i vettori sono ortogonali il coseno assume valore 0; se i vettori puntano in direzioni opposte il coseno assume valore -1. Minore è l'angolo tra i due vettori, maggiore è quindi la similarità tra di essi.

Per valutare la misura più adatta al contesto in analisi è stata calcolata la similarità tra le ricerche di esempio mostrate in Tabella 7.1c. In particolare, è stata presa come riferimento la ricerca R_0 ed è stata calcolata la sua similarità con le altre ricerche presenti in tabella usando le tre metriche valutate. Sono state scelte queste ricerche perché rappresentano un buon esempio dei diversi livelli di similarità che si possono verificare:

- la ricerca R_1 condivide con R_0 tutti i campi eccetto `Utenza`;
- la ricerca R_2 condivide con R_0 tutti i campi eccetto `Utenza` e `Residenzialita`;
- la ricerca R_3 condivide con R_0 i soli campi `Alzheimer` e `Provincia`;
- la ricerca R_4 condivide con R_0 il solo campo `Provincia`;
- la ricerca R_5 non condivide campi con R_0.

Il calcolo delle similarità e il confronto tra le metriche è riportato in Tabella 7.2.

Dal confronto risulta che la metrica del *coefficiente Simple Matching* non è adatta al contesto in analisi in quanto il valore di similarità tra R_0 e R_5, due ricerche con parametri completamente diversi, è superiore a 0 e anche di molto. La ragione di questo è dovuta

	Similarità con la ricerca R_0		
Ricerca R_i *confrontata*	*SMC*	*J*	*cos*
R_0 (tutti i parametri uguali)	1,000	1,000	1,000
R_1 (1 parametro diverso)	0,888	0,666	0,800
R_2 (2 parametri diversi)	0,777	0,428	0,600
R_3 (3 parametri diversi)	0,666	0,250	0,400
R_4 (4 parametri diversi)	0,555	0,111	0,200
R_5 (tutti i parametri diversi)	0,444	0,000	0,000

Tabella 7.2: Confronto tra le metriche di similarità valutate.

al fatto che gli attributi delle ricerche sono di tipo binario asimmetrico; il coefficiente quindi, calcolando la similarità anche sulla base degli 0 in comune, produce un risultato non significativo.

D'altra parte il *coefficiente Jaccard*, calcolando la similarità esclusivamente in base agli 1 in comune, produce un risultato significativo. Il valore di similarità tra ricerche con sempre meno parametri in comune è progressivamente decrescente fino ad arrivare a 0.

Anche la *similarità del coseno* produce un risultato ugualmente significativo; il suo valore è inoltre interpretabile come la percentuale dei parametri in comune tra le due ricerche e questo, considerando tutti i parametri di uguale importanza, è il comportamento desiderato.

Tra le tre metriche valutate, la *similarità del coseno* è quindi quella ritenuta più adatta per il sistema di raccomandazione proposto. Per questa metrica è stata inoltre valutata una versione pesata per poter assegnare più o meno rilevanza ai diversi campi nel calcolo della similarità. Ad esempio, nel contesto in analisi, si ritiene rilevante che due utenti che effettuano ricerche che condividono tutte le esigenze di servizi medicali e differiscono per la sola provincia selezionata, possano essere considerati più simili rispetto a due utenti che selezionano la stessa provincia ma differiscono nella richiesta del servizio *Presidio Alzheimer*. La versione pesata della metrica è riportata di seguito:

$$wcos(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) = \frac{\sum_i w_i u_i v_i}{\sqrt{\sum_i w_i u_i^2}\sqrt{\sum_i w_i v_i^2}} \tag{7.5}$$

Nella scelta dei pesi da assegnare ai parametri di ricerca sono state fatte le seguenti considerazioni:

- si ritiene che il campo `Provincia` possa essere considerato meno rilevante, ai fini del calcolo della similarità, rispetto agli altri campi in quanto non definisce un'esigenza di servizi medicali per il paziente;
- si ritiene che, in mancanza di dati reali che dimostrino il contrario, tutti gli altri campi che descrivono le esigenze sanitarie del paziente debbano essere considerati della stessa rilevanza.

Si è quindi scelto di assegnare al campo `Provincia` un peso dimezzato rispetto al suo valore nella versione della metrica non pesata, e di distribuire uniformemente a tutti gli

altri campi il valore sottratto a tale campo in modo da mantenere la somma complessiva dei pesi uguale ad 1. I valori dei pesi proposti sono riportati in Tabella 7.3.

Pesi	*Valore*
w_{1-2} (`Utenza`)	0,225
w_{3-4} (`Residenzialita`)	0,225
w_{5-7} (`Accreditamento`)	0,225
w_{8-9} (`Alzheimer`)	0,225
w_{10-18} (`Provincia`)	0,100

Tabella 7.3: Valori dei pesi proposti per la versione pesata della *similarità del coseno*.

Per valutare l'efficacia della metrica pesata e per confrontarla con la sua versione non pesata sono stati calcolati nuovamente i valori di similarità tra la ricerca R_0 e le altre ricerche presenti in Tabella 7.1c. I risultati sono riportati in Tabella 7.4.

	Similarità con la ricerca R_0	
Ricerca R_i *confrontata*	*cos*	*wcos*
R_0 (tutti i parametri uguali)	1,000	1,000
R_1 (1 parametro diverso)	0,800	0,775
R_2 (2 parametri diversi)	0,600	0,550
R_3 (3 parametri diversi)	0,400	0,325
R_4 (solo `Provincia` in comune)	0,200	0,100
R_5 (tutti i parametri diversi)	0,000	0,000

Tabella 7.4: Confronto tra i valori di similarità calcolati con la metrica *similarità del coseno* e la versione pesata proposta.

Dai risultati mostrati nella Tabella 7.4 si evince che la metrica proposta conduce al risultato desiderato. Le ricerche che differiscono nei campi riferiti ai servizi medicali differiscono per un valore di similarità di 0,225 punti, mentre due ricerche che differiscono nella provincia selezionata, come R_4 e R_5, differiscono per un valore di similarità inferiore, di soli 0,1 punti.

La metrica selezionata per calcolare la similarità tra le ricerche nel sistema di raccomandazione proposto è quindi la versione pesata della metrica *similarità del coseno* mostrata nell'Equazione (7.5), con i valori dei pesi proposti in Tabella 7.3.

Nel contesto del sistema di raccomandazione in analisi, tale metrica può poi essere riformulata nel modo seguente:

$$Sim(r_1, r_2) = wcos(\boldsymbol{b}_{r_1}, \boldsymbol{b}_{r_2}, \boldsymbol{w}) = \frac{\sum_i w_i b_{r_1,i} b_{r_2,i}}{\sqrt{\sum_i w_i b_{r_1,i}^2}\sqrt{\sum_i w_i b_{r_2,i}^2}} \tag{7.6}$$

in cui:

- $Sim(r_1, r_2)$ rappresenta il valore di similarità tra la ricerca r_1 e la ricerca r_2;

- $\boldsymbol{b}_{r_1}$ rappresenta il vettore dei parametri di ricerca base codificati *one-hot* della ricerca r_1;
- $\boldsymbol{b}_{r_2}$ rappresenta il vettore dei parametri di ricerca base codificati *one-hot* della ricerca r_2;
- $\boldsymbol{w}$ rappresenta il vettore dei pesi dei parametri di ricerca base codificati *one-hot* mostrati in Tabella 7.3.

Selezione delle ricerche Neighbors

Nei sistemi di raccomandazione *neighborhood-based*, il passo successivo al calcolo della similarità tra l'utente attivo e tutti gli altri utenti presenti nel sistema consiste nella selezione dell'insieme degli utenti più simili, i.e. i cosiddetti *neighbors*, da tenere in considerazione per il calcolo delle raccomandazioni. Nel sistema di raccomandazione proposto questo passo consiste nel selezionare le ricerche memorizzate più simili alla ricerca dell'utente attivo, i.e. le ricerche *neighbors*, su cui basare le raccomandazioni dei parametri di ricerca avanzata.

Nell'analisi condotta da Herlocker *et al.* [34] sulle scelte progettuali adottate in letteratura nei sistemi di raccomandazione *neighborhood-based*, sono stati rilevati e confrontati i seguenti tre approcci per la selezione dei *neighbors*:

- il primo approccio consiste nel considerare tutti gli utenti del sistema come utenti *neighbors*, pesando il contributo di ciascuno in base alla similarità con l'utente attivo;
- il secondo approccio consiste nell'ordinare gli utenti in ordine decrescente sulla base del valore di similarità con l'utente attivo e selezionare i *top k* utenti più simili. Questo approccio è chiamato *k-nearest neighbors*;
- il terzo approccio consiste nel definire una soglia minima di similarità e selezionare come *neighbors* tutti gli utenti con valore di similarità superiore alla soglia. Questo approccio è chiamato *threshold-based neighbors*.

Il primo approccio è stato considerato inefficiente da Herlocker *et al.* poiché il fatto di tenere in considerazione tutti gli utenti del sistema, che possono essere milioni, rende difficile poter calcolare raccomandazioni in tempo reale. Nel sistema di raccomandazione proposto questo problema potrebbe non essere rilevante in quanto, essendo l'applicazione in oggetto un'applicazione utilizzabile solo per casi molto specifici e limitata al solo territorio regionale, non si prevede la presenza nella base dati di un numero non gestibile di ricerche. Inoltre, si potrebbe aggirare il problema precalcolando *offline* le raccomandazioni da fornire per ciascuna configurazione di ricerca base. Il problema più rilevante di questo approccio, nel contesto in analisi, consiste invece nel fatto di considerare ricerche di utenti con esigenze di servizi medicali anche molto diverse da quella dell'utente attivo, incrementando quindi la probabilità di raccomandare servizi poco affini alle esigenze dell'utente. Questo approccio non è quindi stato ritenuto adatto nel contesto in analisi.

Nel confronto tra l'approccio *k-nearest neighbors* e l'approccio *threshold-based neighbors*, Herlocker *et al.* hanno concluso che l'approccio *k-nearest neighbors* è quello in grado di fornire un'accuratezza delle raccomandazioni più elevata. Sebbene l'approccio *threshold-based*

neighbors consenta di avere un livello minimo di similarità garantito, lo svantaggio principale di questo approccio è quello di ridurre notevolmente il numero di utenti considerati, portando ad una minore consistenza delle raccomandazioni fornite. Per ovviare a questo problema sarebbe necessario selezionare valori di similarità ridotti ma questo porterebbe nuovamente ad un incremento della probabilità di fornire raccomandazioni poco affini alle esigenze dell'utente. Questo approccio, considerando l'attuale mancanza totale di dati reali di ricerche utenti, non è quindi stato ritenuto adatto per il sistema di raccomandazione proposto. Sarebbe invece possibile riconsiderare lo stesso approccio nel momento in cui nel sistema sarà presente un numero elevato di ricerche tale da poter consentire la generazione di raccomandazioni consistenti.

Per quanto riguarda l'approccio *k-nearest neighbors*, Herlocker *et al.* hanno identificato come ottimali i valori del parametro k all'interno dell'intervallo tra 20 e 50. Per identificare tale intervallo, hanno calcolato l'accuratezza nella predizione delle valutazioni degli utenti con diversi valori di k e selezionato i valori che hanno portato alle predizioni maggiormente accurate. Questo processo di selezione del valore ottimale del parametro k non sembrerebbe però applicabile nel sistema di raccomandazione proposto in quanto, non disponendo delle valutazioni finali da parte degli utenti, non è possibile identificare se una data selezione di parametri in fase di ricerca ha poi portato l'utente verso una struttura adeguata alle sue esigenze o meno. In mancanza di tale informazione e considerando che valori di k più elevati comportano una maggiore probabilità di fornire raccomandazioni poco adatte all'utente, il valore selezionato per il sistema di raccomandazione proposto è 20. Inoltre, tenendo in considerazione il fatto che anche all'interno delle 20 ricerche più simili, almeno nella condizione iniziale del sistema, potrebbero essere presenti ricerche con parametri distanti dai parametri di ricerca selezionati dall'utente attivo, si è optato per l'adozione di una soglia minima di similarità necessaria per la considerazione di una ricerca nel calcolo delle raccomandazioni. Il valore minimo di similarità proposto è di 0,45, il quale garantisce, utilizzando l'Equazione (7.6) con i pesi mostrati in Tabella 7.3, che le ricerche *neighbors* considerate nel calcolo delle raccomandazioni condividano con la ricerca dell'utente attivo almeno due dei parametri riferiti ai servizi medicali richiesti.

Calcolo delle raccomandazioni

Nell'approccio *neighborhood-based*, e in generale per tutta la categoria di algoritmi *collaborative filtering*, il calcolo delle raccomandazioni può essere formulato in due modi [22]:

i. predire la valutazione di un utente per un articolo con cui non ha ancora interagito;

ii. generare una classifica dei k articoli con cui l'utente non ha ancora interagito più rilevanti per l'utente stesso.

Non potendo applicare nessuno dei due approcci per la raccomandazione diretta delle strutture RSA finali, si è optato per l'utilizzo di entrambi gli approcci, in contesti diversi, per la predizione del set di parametri di ricerca avanzata da raccomandare all'utente:

- l'approccio (i) è stato utilizzato per la stima dei cinque valori di rilevanza esprimibili dall'utente attraverso gli slider, mostrati in Figura 4.2, sulla base dei valori di rilevanza espressi nelle ricerche selezionate come *neighbors*;

- l'approccio (ii) è stato utilizzato per la generazione di una classifica dei servizi di cui l'utente può indicare un interessamento, come mostrato in Figura 4.4, sulla base dei servizi presenti nelle ricerche selezionate come *neighbors* e sulla base del grado di interesse specificato per ciascuno di essi.

Per quanto riguarda la stima dei valori di rilevanza, si è scelto di adattare l'approccio comunemente utilizzato nei sistemi *neighborhood-based* per la predizione delle valutazioni al contesto in analisi. In particolare, mentre nei sistemi di raccomandazione *neighborhood-based* tradizionali si usa la media pesata delle valutazioni degli utenti *neighbors* per predire la valutazione per un determinato articolo [35, 36], nel sistema proposto viene calcolata la media pesata dei valori di rilevanza espressi nelle ricerche *neighbors* per stimare i valori di rilevanza per l'utente attivo. La formula utilizzata per la stima di questi valori è la seguente:

$$\widehat{r}_{u,p} = \frac{\sum_{n \in N} r_{n,p} \cdot Sim(u,n)}{\sum_{n \in N} Sim(u,n)} \tag{7.7}$$

in cui:

- $\widehat{r}_{u,p}$ rappresenta il valore di rilevanza stimato per il parametro p per l'utente attivo u;
- N rappresenta l'insieme delle ricerche *neighbors*;
- n rappresenta l'istanza di una delle ricerche *neighbors*;
- $r_{n,p}$ rappresenta il valore di rilevanza espresso per il parametro p nella ricerca n;
- $Sim(u,n)$ rappresenta il valore di similarità tra la ricerca dell'utente attivo u e la ricerca n, calcolato con l'Equazione (7.6).

In letteratura, dell'Equazione (7.7) è stata proposta anche una versione *adjusted* [35] per tenere in considerazione il fatto che utenti diversi possono avere scale di valutazioni diverse. In tale equazione, al posto del calcolo della media pesata delle valutazioni, viene calcolata la media pesata delle deviazioni dalla media delle valutazioni di tale utente, a cui viene aggiunta la media delle valutazioni dell'utente attivo. Questo approccio non è stato però ritenuto adatto al sistema di raccomandazione proposto in quanto, non essendo disponibile un profilo utente, non è possibile ricavare la media dei valori di rilevanza dell'utente attivo.

Per quanto riguarda la generazione della classifica di servizi, si è scelto di utilizzare un metodo simile a quello utilizzato da Narducci *et al.* [28] per la generazione della classifica di medici per l'utente attivo. Allo stesso modo con cui nel loro sistema viene calcolato un punteggio per ciascun medico sulla base della media pesata delle valutazioni degli utenti, nel sistema proposto si è deciso di assegnare un punteggio a ciascun servizio sulla base della media pesata del grado di interessamento espresso nelle ricerche *neighbors*. La formula per il calcolo dei punteggi è la seguente:

$$score_{u,s} = \sum_{n \in N} i_{n,s} \cdot Sim(u,n) \tag{7.8}$$

in cui:

- $score_{u,s}$ rappresenta il punteggio assegnato al servizio s per l'utente attivo u;
- N rappresenta l'insieme delle ricerche *neighbors*;
- n rappresenta un'istanza di una delle ricerche *neighbors*;
- $i_{n,s}$ rappresenta il valore di interessamento per il servizio s indicato nella ricerca n. Il valore di interessamento è calcolato come segue:

$$i_{n,s} = \begin{cases} 0 & \text{se non è stato indicato un interessamento per il servizio} \\ 1 & \text{se è stato indicato un interessamento con opzione } \textit{Interessato} \\ 2 & \text{se è stato indicato un interessamento con opzione } \textit{Molto interessato} \end{cases}$$

- $Sim(u, n)$ rappresenta il valore di similarità tra la ricerca dell'utente attivo u e la ricerca n, calcolato con l'Equazione (7.6).

Raccomandazione progressiva dei servizi

Utilizzando la stessa metodologia impiegata per la generazione della classifica di servizi iniziale, è stata inoltre valutata la possibilità di una raccomandazione progressiva dei servizi, sulla base dei servizi inseriti di volta in volta dall'utente. L'obiettivo di questo approccio è quello di poter raffinare dinamicamente la classifica in base alle preferenze sui servizi espresse direttamente dall'utente in fase di ricerca.

L'approccio considerato consiste nella generazione di una nuova classifica ogniqualvolta l'utente esplicita l'interessamento per un servizio. La classifica è quindi ricalcolata considerando le sole ricerche *neighbors* nelle quali è presente, nella lista di servizi, l'insieme di servizi inserito dall'utente nella fase di ricerca. In questo modo la classifica di servizi proposti è progressivamente raffinata considerando, di volta in volta, le ricerche sempre più simili alle esigenze dell'utente.

Come alternativa all'approccio basato sul *collaborative filtering* appena descritto, per la raccomandazione progressiva dei servizi è stata valutata la tecnica di Data Mining delle regole di associazione. Questo secondo approccio, basandosi sul calcolo di misure statistiche, risulta più complesso ma anche più completo del primo in quanto permette di stimare la probabilità con la quale un determinato servizio può essere d'interesse sulla base dei servizi già inseriti. Inoltre, l'utilizzo di una seconda tecnica di raccomandazione garantisce una minor probabilità di ridondanza e polarizzazione delle raccomandazioni proposte.

L'approccio basato sulle regole di associazione è quindi quello selezionato per la raccomandazione progressiva dei servizi. Poiché basato su un approccio diverso dal *collaborative filtering*, tale metodo di raccomandazione costituisce una componente separata del sistema di raccomandazione, i.e. la seconda, descritta nel Paragrafo 7.3.2.

7.3.2 Raccomandazioni mediante regole di associazione

Di seguito sono descritte le fasi della progettazione della componente *association rules-based* del sistema di raccomandazione proposto: selezione della modalità di calcolo delle regole, preprocessing dei dati, estrazione delle regole di associazione e calcolo delle raccomandazioni.

Modalità di calcolo delle regole

Il primo passo nella progettazione della componente *association rules-based* è consistito nella selezione della modalità di calcolo delle regole di associazione tra *online* e *offline*. La modalità di calcolo *online* consiste nell'estrazione delle regole in tempo reale nel momento in cui si devono fornire le raccomandazioni, mentre la modalità di calcolo *offline* consiste in una fase di precalcolo delle regole antecedente al momento di calcolo delle raccomandazioni, durante il quale è sufficiente solo analizzare le regole estratte in precedenza.

Per poter selezionare la modalità più appropriata al contesto è stato necessario definire quali fossero i dati su cui basare l'estrazione delle regole di associazione. Nell'ambito dei sistemi di raccomandazione *association rules-based* sono infatti possibili due opzioni: estrarre le regole di associazione e, di conseguenza, fornire raccomandazioni tenendo in considerazione indistintamente tutte le transazioni del dataset [38] oppure restringere le transazioni considerando solamente quelle affini all'utente a cui si devono fornire raccomandazioni [39]. Nel contesto in analisi questo si traduce nella scelta tra l'utilizzare tutte le ricerche memorizzate nel dataset, indistintamente dalla tipologia di utente che l'ha effettuata, oppure considerare solo le ricerche affini all'utente attivo sulla base dei parametri di ricerca base selezionati. Tenendo in considerazione che, nell'ambito del sistema di raccomandazione proposto, alcune tipologie di servizi che possono essere rilevanti per un certo tipo di utenti potrebbero invece non esserlo per altri, poiché adatte solo per determinate condizioni sanitarie, si è ritenuto più appropriato il secondo approccio, il quale estrae associazioni a partire esclusivamente dalle ricerche di utenti con le stesse necessità di servizi medicali.

Tra i parametri di ricerca base da tenere in considerazione per la selezione delle ricerche da utilizzare si è scelto di considerare i quattro parametri riferiti alla richiesta di servizi medicali, i.e. *Accreditamento, Presidio Alzheimer, Residenzialità* e *Autosufficienza*, mentre si è scelto di non considerare il parametro *Provincia* in quanto non definisce una caratteristica sanitaria del paziente.

Avendo scelto di estrarre le regole di associazione sulla base della tipologia di utente che effettua la ricerca, non è quindi stato possibile effettuare un'estrazione univoca delle regole in modalità *offline* da utilizzare successivamente per tutti gli utenti. Sono quindi state valutate le due opzioni seguenti:

i. effettuare in modalità *offline* un'estrazione multipla delle regole di associazione, una per ciascuna combinazione possibile dei parametri di ricerca base selezionabili;

ii. effettuare in modalità *online* l'estrazione delle regole in tempo reale, filtrando le ricerche in base ai parametri di ricerca base indicati dall'utente.

Considerando che, nel contesto in analisi, le dimensioni del dataset reale si stimano essere contenute e che a seguito del filtraggio delle ricerche sulla base dei parametri di ricerca base si stima siano ancora inferiori, si è optato per l'estrazione in modalità *online*, in quanto le dimensioni contenute del dataset consentono un'estrazione delle regole in un tempo ragionevole.

Preprocessing dei dati

Per la componente *association rules-based* sono state valutate le seguenti operazioni di preprocessing: pulizia dei dati e filtraggio e conversione delle ricerche.

Pulizia dei dati

L'operazione di pulizia dei dati per la componente in oggetto è consistita nel verificare la presenza dati errati o mancanti all'interno degli attributi considerati per l'estrazione delle regole di associazione, i.e. `Servizi_RSA` o `Servizi_contorno` a seconda del caso.

All'interno del dataset sintetico tali attributi presentano dei dati mancanti in quanto, non solo essi sono presenti esclusivamente all'interno delle ricerche ottimizzate, ma anche all'interno di queste può accadere che tali campi risultino vuoti poiché l'inserimento dei servizi è opzionale per l'utente. Si assume che la stessa situazione si verifichi anche con i dati di ricerche utenti reali mentre si esclude la presenza di dati errati all'interno di questi campi poiché i servizi inseribili dall'utente sono selezionati da un insieme predefinito di opzioni possibili. Poiché le ricerche in cui non sono specificati servizi non sarebbero significative ai fini dell'estrazione delle regole di associazione, tutte le ricerche che presentano il campo `Servizi_RSA`, o `Servizi_contorno` rispettivamente, mancante o vuoto sono state filtrate dal dataset.

Filtraggio e conversione delle ricerche

Poiché per l'estrazione delle regole di associazione si è scelto di considerare le sole ricerche affini alle esigenze di servizi medicali espresse dall'utente, tutte le ricerche con i valori dei campi `Utenza`, `Alzheimer`, `Residenzialita` e `Accreditamento` diversi dai parametri di ricerca base selezionati dall'utente attivo sono filtrate dal dataset. Non essendo noti a priori i parametri selezionati dall'utente, questa operazione è eseguita in tempo reale nel momento in cui l'utente effettua la ricerca.

Inoltre, poiché gli algoritmi di estrazione delle regole di associazione richiedono in ingresso un dataset transazionale, composto quindi da liste di *item*, all'operazione di filtraggio delle ricerche segue un'operazione di rimozione e conversione degli attributi con il fine di trasformare il dataset in tale formato. Tutti gli attributi diversi da `Servizi_RSA`, o rispettivamente `Servizi_contorno`, sono quindi eliminati. La struttura del campo rimanente è infine appiattita in modo da contenere la sola lista di servizi selezionati. La struttura di tale campo è infatti originariamente composta da tre sottocampi, come mostrato nelle Tabelle 4.5 e 4.6; di tali sottocampi, `Categorie` è rimosso mentre `Interessato` e `Molto_interessato` sono uniti. Il risultato di questa operazione è un dataset composto da una lista di transazioni di servizi nel formato richiesto per l'estrazione delle regole di associazione. Un estratto di esempio del dataset in formato transazionale è mostrato in Tabella 7.5.

Estrazione delle regole di associazione

L'operazione di estrazione delle regole di associazione è il processo che consente, a partire da un dataset composto da transazioni, di trovare tutte le regole che soddisfano particolari soglie di supporto e confidenza [23].

	`Servizi_RSA`
R_0	*Lavanderia, Film, Palestra*
R_1	*Podologo, Camera singola*
R_2	*Palestra, Animazione, Gite, Laboratorio*
R_3	*Camera con tv, Giornali, Estetista*
R_4	*Parrucchiere/Barbiere, Estetista*
R_5	*Animazione, Palestra, Gite, Lavanderia, Stireria*

Tabella 7.5: Esempi di ricerche convertite in transazioni di servizi.

Il processo consiste nei passi seguenti:

1. definizione delle soglie minime dei valori di supporto e confidenza;
2. estrazione degli *itemset frequenti*, i.e. gli *itemset* con valore di supporto superiore alla soglia definita;
3. generazione delle regole di associazione, a partire dagli *itemset frequenti* trovati nel punto precedente, che soddisfano il valore minimo di confidenza selezionato.

Di seguito è descritto come è stato affrontato il problema dell'estrazione degli *itemset frequenti* e come sono state selezionate le soglie di supporto e confidenza per l'estrazione delle regole.

Estrazione degli itemset frequenti

L'operazione di estrazione degli *itemset frequenti* è l'operazione computazionalmente più onerosa del processo di estrazione delle regole di associazione in quanto consiste nella ricerca di tutte le possibili combinazioni di *item* che appaiono nelle transazioni della base dati.

Per eseguire tale operazione sono quindi stati valutati due degli algoritmi più noti per l'estrazione efficiente degli *itemset frequenti*, l'algoritmo *Apriori* e l'algoritmo *FP-growth* [23]. I due algoritmi sono descritti brevemente di seguito.

Algoritmo Apriori L'algoritmo Apriori si basa sulla proprietà di *anti-monotonicità* del supporto: *se un itemset è frequente, allora anche tutti i suoi sotto-insiemi lo sono.* In formule, dati due *itemset* X e Y:

$$(X \subseteq Y) \implies support(X) \geq support(Y)$$

Questo algoritmo sfrutta una strategia efficiente di ricerca in ampiezza dei possibili *itemset* candidati attraverso un filtraggio degli *itemset* che non soddisfano la soglia minima di supporto selezionata. Questa strategia permette di ridurre in modo considerevole lo spazio di ricerca di tutti i possibili *itemset*. L'algoritmo, inizialmente, genera tutti i possibili candidati *itemset frequenti* composti da un singolo elemento. Successivamente, procede con il calcolo del supporto di ciascuno di essi e con il filtraggio di quelli con valore di supporto superiore alla soglia minima. L'algoritmo procede

poi iterativamente eseguendo i seguenti passi fino a quando non sono più disponibili *itemset* candidati:

1. generazione di tutti gli *itemset* di dimensione $n + 1$ attraverso tutte le possibili combinazioni degli *itemset* non filtrati di dimensione n;
2. calcolo del supporto degli *itemset* generati al punto precedente;
3. filtraggio degli *itemset* con supporto maggiore della soglia minima definita.

Algotimo FP-growth L'algoritmo FP-growth (acronimo di *Frequent Pattern growth*) ricerca gli *itemset frequenti* senza la generazione iterativa dei possibili candidati. Esso consiste principalmente in due passi:

1. costruzione di una struttura dati compatta chiamata *FP-Tree*, contenente una rappresentazione compressa del dataset;
2. estrazione degli *itemset frequenti* in modo efficiente direttamente dalla struttura dati FP-Tree.

L'operazione più onerosa di questo algoritmo consiste nella creazione iniziale della struttura dati FP-Tree che, a sua volta, consiste in due passi:

1. l'algoritmo effettua una prima scansione del database per contare le occorrenze dei singoli *item* presenti nelle transazioni e ne memorizza i risultati in una struttura dati chiamata *header table*. Gli *item* con supporto inferiore alla soglia minima sono scartati mentre quelli rimanenti sono ordinati in ordine decrescente di supporto;
2. l'algoritmo effettua una seconda scansione del database inserendo le transazioni nella struttura dati FP-Tree: gli *item* all'interno delle transazioni vengono prima ordinati in ordine decrescente di supporto, come riportato nella *header table*, e successivamente vengono inseriti nella struttura dati, memorizzando i prefissi in comune una sola volta.

Nel selezionare tra i due algoritmi quello più adatto al contesto in analisi si è tenuto particolarmente in considerazione il fattore tempo di esecuzione, in modo da consentire un'estrazione delle regole quanto più possibile in tempo reale.

In letteratura vari studi sono stati condotti per confrontare le prestazioni dei due algoritmi. Mythili *et al.* [37], ad esempio, hanno confrontato i due algoritmi applicandoli a diverse tipologie di dataset transazionali concludendo che l'algoritmo FP-growth risulta generalmente più efficiente e performante dell'algoritmo Apriori. Questo sarebbe principalmente dovuto al fatto che l'algoritmo FP-growth, non prevedendo la fase di generazione degli *itemset* candidati, necessita di sole due scansioni del dataset. L'algoritmo Apriori richiede invece una scansione dell'intero dataset ad ogni iterazione per conteggiare il supporto degli *itemset* candidati. Il confronto tra i due algoritmi è riassunto in Tabella 7.6.

Tra i due algoritmi è stato quindi selezionato l'algoritmo FP-growth in quanto comporta generalmente un tempo di esecuzione inferiore.

	Apriori	*FP-growth*
Velocità di esecuzione	Più lento	Più veloce
Memoria utilizzata	Più memoria (memorizzazione di tutti gli itemset frequenti candidati)	Meno memoria (memorizzazione della sola rappresentazione compatta del dataset)
Scansioni del dataset	Una per ogni iterazione dell'algoritmo	Due

Tabella 7.6: Confronto tra l'algoritmo Apriori e l'algoritmo FP-growth.

Generazione delle regole

Il passo successivo per la generazione delle regole di associazione consiste nella selezione dei valori di soglia minimi dei parametri di supporto e confidenza. Una selezione appropriata di tali valori è determinante per le prestazioni del sistema di raccomandazione [40]: selezionando un valore di supporto troppo elevato il numero di *itemset frequenti* estratti potrebbe infatti essere troppo basso, con la conseguenza di non poter estrarre sufficienti regole e di non riuscire a fornire raccomandazioni; selezionando un valore di supporto troppo basso invece il tempo di esecuzione richiesto per l'estrazione degli *itemset frequenti* potrebbe risultare troppo elevato con la conseguenza di non riuscire a fornire raccomandazioni in tempo reale. In modo analogo, selezionando un valore di confidenza troppo basso le raccomandazioni risultanti potrebbero risultare poco utili per l'utente, in quanto basate su associazioni apparse troppo raramente nelle ricerche passate per poter essere considerate rilevanti; con una selezione di un valore troppo elevato invece il numero di regole estratte potrebbe risultare troppo scarso per poter fornire raccomandazioni.

Per la selezione ottimale di questi valori sono quindi stati valutati alcuni degli approcci proposti in letteratura per la selezione di tali parametri nell'ambito dei sistemi di raccomandazione.

M. S. Reddy *et al.* [38], ad esempio, hanno proposto un sistema di raccomandazione basato su regole di associazione per un sito di *e-commerce* in un contesto di Big Data. Nel loro sistema, i parametri di supporto e confidenza sono stati selezionati staticamente con valore 0,5. Tale valore non sembrerebbe però adeguato al contesto in analisi; un valore così elevato di supporto rischierebbe infatti di portare all'estrazione di troppe poche regole, se non di estrarne nessuna, in un contesto come quello in analisi in cui il dataset è stimato essere di dimensioni contenute e che partirebbe da una condizione iniziale di totale assenza di dati.

Bendakir *et al.* [39] hanno utilizzato invece un approccio differente all'interno di un sistema di raccomandazione per corsi universitari. Per selezionare i valori di supporto e confidenza più adeguati, nel loro sistema hanno utilizzato un processo di ottimizzazione basato sull'accuratezza delle regole generate, integrato con un sistema di valutazione delle raccomandazioni risultanti da parte degli studenti. A seguito di tale processo, hanno concluso che i risultati migliori sono ottenuti con un valore di supporto intorno a 0,1 e un valore di confidenza superiore a 0,5. Un simile approccio non sembrerebbe però

attualmente applicabile al sistema in analisi in quanto, né si dispone di dati reali su cui valutare l'accuratezza delle raccomandazioni, né si dispone di un feedback da parte degli utenti. Inoltre, poiché le regole estratte variano sulla base della configurazione di ricerca base selezionata dall'utente, sarebbe necessario eseguire tale processo per tutte le possibili configurazioni selezionabili.

Un ulteriore approccio nella selezione dei valori di supporto e confidenza è quello proposto da Lin *et al.* [40]. Focalizzandosi sull'estrazione delle regole di associazione con il fine di fornire raccomandazioni loro hanno proposto un algoritmo per l'adattamento dinamico del valore del supporto, con il fine di ovviare al problema della valorizzazione predefinita di tale valore. Il loro algoritmo si pone come alternativa ai tradizionali algoritmi di estrazione degli *itemset frequenti* in quanto, al posto di richiedere in ingresso un valore di soglia minima per il supporto, richiede invece di specificare l'intervallo del numero di regole da estrarre desiderato. L'algoritmo procede quindi con l'estrazione degli *itemset frequenti* e la generazione delle regole in modo iterativo, selezionando valori di supporto progressivamente decrescenti fino all'ottenimento del numero di regole desiderato. Nonostante questo approccio risulti particolarmente interessante in un contesto dove il numero di raccomandazioni che si desidera fornire è limitato, come quello in analisi, non sembrerebbe però adatto al fornire le stesse in tempo reale. Il fatto di dover ripetere molteplici volte il processo di estrazione degli *itemset frequenti* e delle regole risultanti comporterebbe infatti un tempo di esecuzione troppo elevato.

A seguito delle valutazioni di applicabilità degli approcci citati si è optato per una valorizzazione predefinita dei parametri di supporto e confidenza, selezionandoli con l'obiettivo di ottenere, di volta in volta, un insieme di regole il più rilevanti possibile e nel minor tempo possibile. I valori di supporto e confidenza proposti sono rispettivamente di 0,2 e 0,7. Il valore di supporto di 0,2 si stima possa essere un buon candidato in quanto valore non troppo elevato, tale da consentire la generazione di un numero sufficiente di regole, e non troppo basso, tale da consentire un tempo di esecuzione accettabile. Il valore di confidenza di 0,7 è stato invece selezionato con l'obiettivo di fornire raccomandazioni sui servizi il più rilevanti possibile. Essendo un valore piuttosto elevato questo valore potrebbe portare all'estrazione di poche regole ma, considerando l'ambito di applicazione del sistema di raccomandazione proposto, i.e. l'ambito sanitario, dove una raccomandazione errata può portare ad una scelta inadeguata alle condizioni di salute del paziente, tra il rischio di fornire poche raccomandazioni e quello di fornire raccomandazioni poco adatte si è ritenuto meno problematico il primo. Per una selezione ancora più accurata dei servizi di raccomandare si è inoltre scelto di filtrare ulteriormente le regole estrarre sulla base del valore di lift; in particolare, le uniche regole considerate per la raccomandazione dei servizi sono quelle aventi un valore di lift maggiore di 1, in modo da garantire una correlazione positiva tra i servizi selezionati dall'utente e quelli raccomandati.

Calcolo delle raccomandazioni

Nei sistemi di raccomandazione *association rules-based* tradizionalmente le raccomandazioni sono calcolate sulla base degli *item* presenti nel conseguente delle regole estratte [38, 39, 40]. Nel sistema di raccomandazione proposto si è scelto di usare lo stesso approccio selezionando i servizi da raccomandare tra quelli presenti nel conseguente delle regole estratte. Di tali regole sono poi considerate le sole regole aventi il conseguente con

cardinalità unitaria e aventi come antecedente l'insieme dei servizi selezionati dall'utente. Le regole restanti sono quindi ordinate in ordine decrescente di confidenza e, di queste, sono selezionate le *top k* regole. I servizi presenti nel conseguente delle regole selezionate costituiscono i servizi raccomandati all'utente in ordine di rilevanza.

7.4 Sviluppo del sistema

In questa sezione viene descritto come è stato sviluppato e implementato all'interno dell'applicazione il sistema di raccomandazione proposto. La sezione è suddivisa in due parti: la prima parte descrive lo sviluppo delle raccomandazioni collaborative mentre la seconda parte descrive lo sviluppo delle raccomandazioni mediante regole di associazione.

7.4.1 Raccomandazioni collaborative

Di seguito viene descritto come sono stati sviluppati il front-end e il back-end per l'implementazione delle raccomandazioni collaborative.

Front-End

La visualizzazione delle raccomandazioni collaborative lato client avviene all'interno della pagina *Ricerca* della sezione *RSA*. In particolare, le raccomandazioni sono state integrate all'interno della sezione *Ricerca Ottimizzata* di tale pagina. Il calcolo delle raccomandazioni è effettuato lato server al momento dell'accesso dell'utente in tale sezione.

Al fine di consentire all'utente di scegliere se ricevere raccomandazioni o meno, è stato inserito il bottone *Abilita Raccomandazioni*, mostrato in Figura 7.3, che permette di abilitarne o disabilitarne la visualizzazione. Alla pressione del bottone *Abilita Raccomandazioni*, le raccomandazioni vengono abilitate ed il testo del bottone viene convertito in *Disabilita Raccomandazioni*.

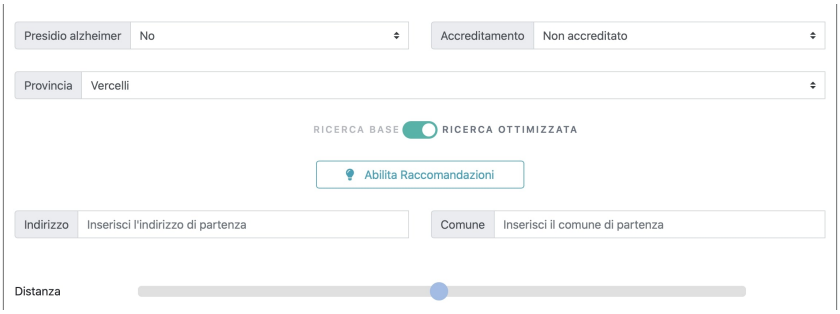


Figura 7.3: Sezione *Ricerca Ottimizzata* con il bottone *Abilita Raccomandazioni*.

Le raccomandazioni riguardanti i valori di rilevanza stimati sono visualizzate spostando dinamicamente la posizione dei cursori degli slider ai valori corrispondenti. Un esempio di raccomandazione dei valori di rilevanza è mostrato in Figura 7.4.

La raccomandazione dei servizi RSA maggiormente richiesti nelle ricerche *neighbors* è visualizzabile invece all'interno della sottosezione *Servizi RSA*. Espandendo la sottosezione,

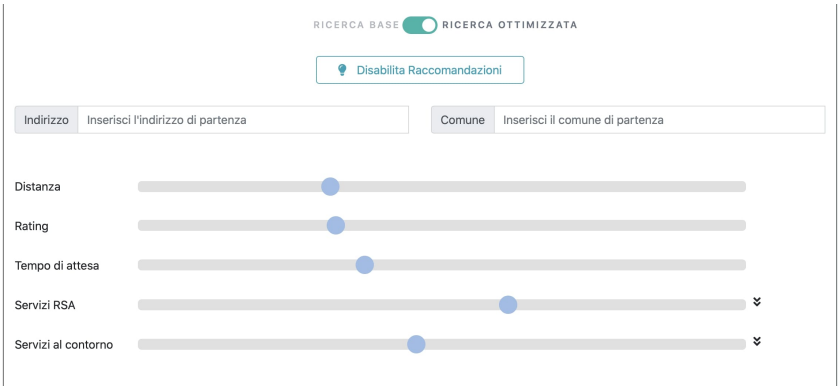


Figura 7.4: Raccomandazione dei valori di rilevanza.

come mostrato in Figura 7.5, viene visualizzato un riquadro in cui è mostrato l'elenco dei primi sei servizi presenti nella classifica di servizi RSA calcolata. Nel caso i servizi in classifica siano meno di sei, sono mostrati i soli servizi presenti.

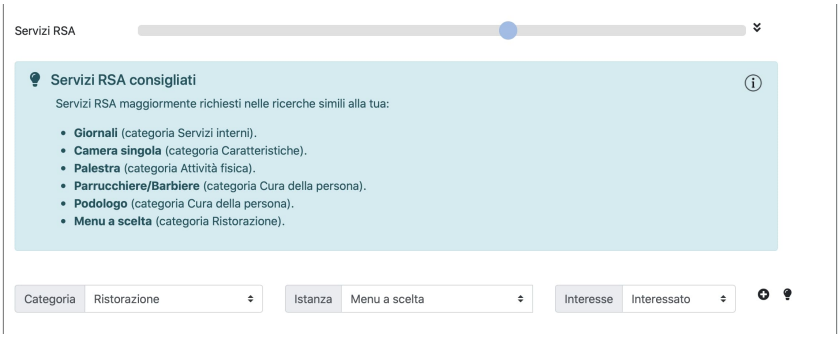


Figura 7.5: Raccomandazione dei servizi RSA maggiormente richiesti.

In modo analogo, la raccomandazione dei servizi al contorno maggiormente richiesti nelle ricerche *neighbors* è visualizzabile all'interno della sottosezione *Servizi al contorno*. Espandendo la sottosezione, come mostrato in Figura 7.6, viene visualizzato un riquadro in cui è mostrato l'elenco dei primi sei servizi presenti nella classifica di servizi al contorno calcolata. Nel caso i servizi in classifica siano meno di sei, come nella Figura 7.6, sono mostrati i soli servizi presenti.

Back-End

Per l'implementazione delle raccomandazioni collaborative lato server è stato sviluppato il servizio `flask_raccomandazioni` che risponde sulla porta 8105. I dettagli del servizio sono riportati in Tabella 7.7.

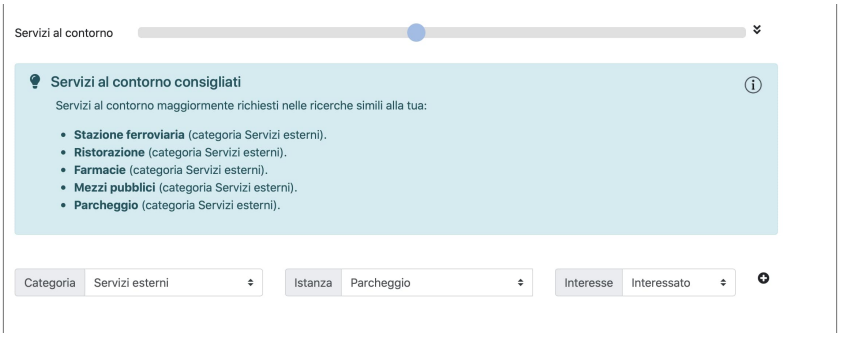


Figura 7.6: Raccomandazione dei servizi al contorno maggiormente richiesti.

`flask_raccomandazioni`	
Porta	8105
Protocollo	HTTP
Metodo	GET

Request (Parametri GET)	`provincia: String`
	`utenza: String`
	`residenzialita: String`
	`alzheimer: String`
	`accreditamento: String`

Response (JSON)	`rilevanza: Object`
	`servizi_rsa: Object`
	`servizi_esterni: Object`

Tabella 7.7: Dettagli del servizio `flask_raccomandazioni`.

Descrizione dei parametri in ingresso

I parametri ricevuti in ingresso dal servizio come parametri della richiesta GET inviata dal client sono:

- **`accreditamento`**: parametro contenente il valore di *Accreditamento* selezionato dall'utente nella sezione *Ricerca Base*.
- **`alzheimer`**: parametro contenente il valore di *Presidio Alzheimer* selezionato dall'utente nella sezione *Ricerca Base*.
- **`provincia`**: parametro contenente il valore di *Provincia* selezionato dall'utente nella sezione *Ricerca Base*.
- **`residenzialita`**: parametro contenente il valore di *Residenzialità* selezionato dall'utente nella sezione *Ricerca Base*.
- **`utenza`**: parametro contenente il valore di *Autosufficienza* selezionato dall'utente nella sezione *Ricerca Base*.

Descrizione dei parametri in uscita

La risposta del server è una risposta in formato JSON contenente le raccomandazioni calcolate. I campi presenti nella risposta sono:

- **`rilevanza`**: parametro contenente i valori di rilevanza raccomandati. I dati di questo parametro sono utilizzati lato client per impostare i valori iniziali degli slider, come mostrato in Figura 7.4.
- **`servizi_rsa`**: parametro contenente la classifica dei servizi RSA più richiesti. I dati di questo parametro sono utilizzati lato client per popolare l'elenco dei servizi RSA raccomandati, come mostrato in Figura 7.5.
- **`servizi_esterni`**: parametro contenente la classifica dei servizi al contorno più richiesti. I dati di questo parametro sono utilizzati lato client per popolare l'elenco dei servizi al contorno raccomandati, come mostrato in Figura 7.6.

Calcolo delle raccomandazioni

Per il calcolo delle raccomandazioni collaborative sono stati utilizzati i moduli `preprocessing` e `neighbors` della libreria Scikit-learn [6] ed il modulo `spatial.distance` della libreria SciPy [5]. In particolare, del modulo `preprocessing` è stata utilizzata la classe `OneHotEncoder` con cui sono stati codificati gli attributi di ricerca base delle ricerche tramite *one-hot encoding*. Del modulo `neighbors` è stata invece utilizzata la classe `NearestNeighbors` che fornisce un'implementazione dell'algoritmo *k-nearest neighbors*. Attraverso il metodo `kneighbors()` di tale classe sono state quindi ricercate, di volta in volta, le k ricerche *neighbors* su cui basare le raccomandazioni. Alla classe è stata inoltre fornita in ingresso la funzione `cosine` del modulo `spatial.distance` che implementa la versione pesata della metrica *similarità del coseno* mostrata nell'Equazione (7.5).

7.4.2 Raccomandazioni mediante regole di associazione

Di seguito viene descritto come sono stati sviluppati il front-end e il back-end per l'implementazione delle raccomandazioni *association rules-based*.

Front-End

La visualizzazione delle raccomandazioni mediante regole di associazione lato client avviene all'interno della pagina *Ricerca* della sezione *RSA*. In particolare, le raccomandazioni sono state integrate all'interno delle sottosezioni *Servizi RSA* e *Servizi al contorno* della sezione *Ricerca Ottimizzata* di tale pagina. Il calcolo delle raccomandazioni è effettuato lato server al momento dell'inserimento di servizi da parte dell'utente.

Come per le raccomandazioni collaborative, per permettere all'utente di scegliere se ricevere raccomandazioni o meno, anche le raccomandazioni mediante regole di associazione sono visualizzate solo nel caso in cui le raccomandazioni siano state abilitate tramite il bottone *Abilita Raccomandazioni*.

Le raccomandazioni di servizi RSA vengono calcolate lato server al momento dell'inserimento da parte dell'utente di un determinato servizio all'interno della sottosezione *Servizi*

RSA. Alla risposta del server, se sono disponibili raccomandazioni, viene visualizzata una finestra a scomparsa in cui vengono raccomandati i primi sei servizi in ordine decrescente di confidenza. Oltre all'indicazione del servizio raccomandato, nella finestra a scomparsa viene anche fornita l'indicazione della confidenza con cui viene raccomandato tale servizio. Un esempio di raccomandazione a seguito dell'inserimento del servizio *Podologo* è mostrato in Figura 7.7.

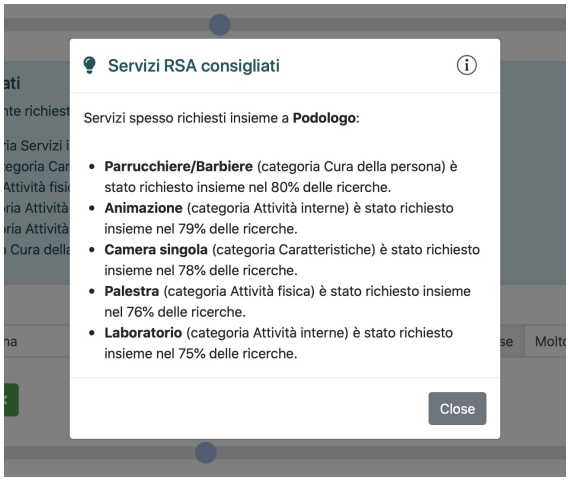


Figura 7.7: Raccomandazione di servizi RSA mediante regole di associazione.

Le raccomandazioni sono poi ulteriormente raffinate all'inserimento di ogni nuovo servizio da parte dell'utente: esse vengono ricalcolate considerando l'insieme di servizi inserito dall'utente e, se disponibili, vengono utilizzate per aggiornare i dati all'interno della finestra a scomparsa. Un esempio di raccomandazione a seguito dell'inserimento del servizio *Podologo*, seguito dall'inserimento del servizio *Animazione* è mostrato in Figura 7.8.

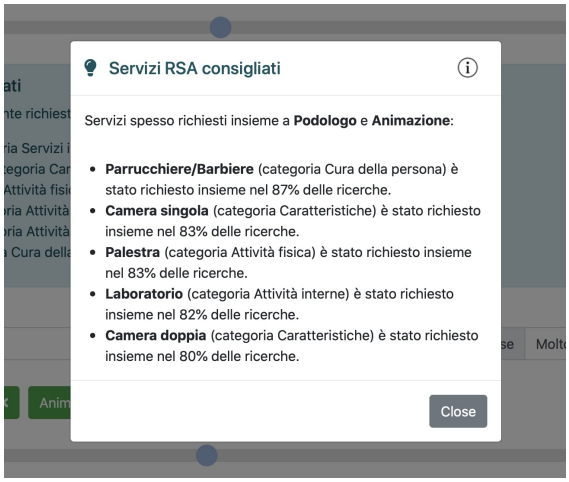


Figura 7.8: Raccomandazione progressiva di servizi RSA mediante regole di associazione.

In modo analogo, le raccomandazioni per i servizi al contorno sono calcolate al momento dell'inserimento di servizi da parte dell'utente all'interno della sottosezione *Servizi al contorno*. Nel caso siano disponibili raccomandazioni, esse sono visualizzate, come nel caso dei servizi RSA, all'interno di una finestra a scomparsa. Un esempio di raccomandazione di servizi al contorno mediante regole di associazione è mostrato in Figura 7.9.

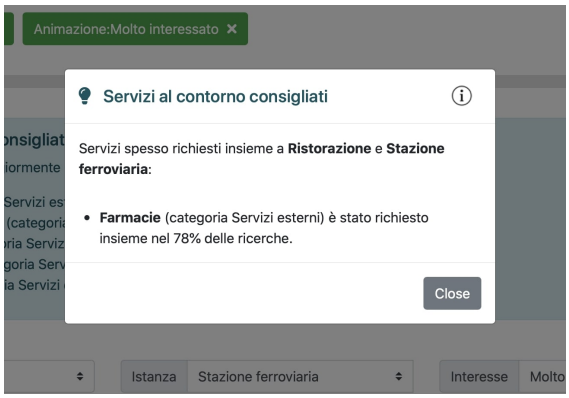


Figura 7.9: Raccomandazione di servizi al contorno mediante regole di associazione.

Back-End

Per l'implementazione delle raccomandazioni mediante regole di associazione lato server è stato sviluppato il servizio `flask_regole_associazione` che risponde sulla porta 8106. I dettagli del servizio sono riportati in Tabella 7.8.

`flask_regole_associazione`	
Porta	8106
Protocollo	HTTP
Metodo	GET

Request (Parametri GET)	`tipo: String`
	`servizi[]: List<String>`
	`utenza: String`
	`residenzialita: String`
	`alzheimer: String`
	`accreditamento: String`

Response (JSON)	`servizi_raccomandati: List<Object>`

Tabella 7.8: Dettagli del servizio `flask_regole_associazione`.

Descrizione dei parametri in ingresso

I parametri ricevuti in ingresso dal servizio come parametri della richiesta GET inviata dal client sono:

- **`tipo`**: parametro contenente il valore per selezionare il tipo di servizi di cui calcolare le raccomandazioni, i.e. `RSA` o `CONTORNO` a seconda del caso.
- **`servizi[]`**: parametro contenente la lista di servizi di cui l'utente ha espresso interessamento.
- **`accreditamento`**: parametro contenente il valore di *Accreditamento* selezionato dall'utente nella sezione *Ricerca Base*.
- **`alzheimer`**: parametro contenente il valore di *Presidio Alzheimer* selezionato dall'utente nella sezione *Ricerca Base*.
- **`residenzialita`**: parametro contenente il valore di *Residenzialità* selezionato dall'utente nella sezione *Ricerca Base*.
- **`utenza`**: parametro contenente il valore di *Autosufficienza* selezionato dall'utente nella sezione *Ricerca Base*.

Descrizione dei parametri in uscita

La risposta del server è una risposta in formato JSON contenente la lista di servizi raccomandati. Il campo presente nella risposta è:

- **`servizi_raccomandati`**: parametro contenente la lista di servizi raccomandati. Esso consiste in una lista di `Object` in cui ciascun oggetto è composto da una coppia *nome servizio - valore di confidenza*. I valori di questo parametro sono utilizzati per popolare la finestra a scomparsa mostrata nelle Figure 7.7, 7.8 e 7.9.

Estrazione delle regole di associazione

Per l'estrazione delle regole di associazione sono stati utilizzati i moduli `preprocessing` e `frequent_patterns` della libreria MLxtend [7]. In particolare, del modulo `preprocessing` è stata utilizzata la classe `TransactionEncoder` con cui è stata effettuata l'operazione di conversione del dataset in formato transazionale. Del modulo `frequent_patterns` sono state invece utilizzate le funzioni `fpgrowth` e `association_rules`. La funzione `fpgrowth` fornisce l'implementazione dell'algoritmo FP-growth ed è stata utilizzata per l'estrazione degli *itemset frequenti*. Tale funzione richiede in ingresso un dataset transazionale e un valore minimo di supporto e produce in uscita un DataFrame composto dalle colonne `support` e `itemsets`, contenenti gli *itemset frequenti* estratti ed i corrispondenti valori di supporto. La funzione `association_rules` è stata invece utilizzata per la generazione delle regole di associazione a partire dagli *itemset frequenti* estratti dalla funzione `fpgrowth`. Tale funzione richiede in ingresso il DataFrame generato dalla funzione `fpgrowth` ed un valore minimo di confidenza. La funzione restituisce a sua volta un DataFrame composto dalle colonne `antecedents`, `consequents`, `antecedent support`, `consequent support`,

`support`, `confidence` e `lift`: le colonne `antecedents` e `consequents` contengono rispettivamente l'antecedente ed il conseguente della regola mentre le restanti colonne riportano i valori delle rispettive metriche della regola di associazione estratta.

Capitolo 8

Conclusioni e sviluppi futuri

Negli ultimi anni la quantità considerevole di dati clinici raccolta quotidianamente ha sollevato il bisogno di utilizzare in ambito sanitario sistemi in grado di analizzare, visualizzare ed impiegare i dati raccolti per supportare sia i pazienti sia i professionisti del settore nel prendere decisioni più efficienti ed accurate per quanto riguarda la salute.

Il presente lavoro di tesi si è inserito all'interno dello sviluppo di un'applicazione per l'esplorazione di strutture assistenziali post-degenza regionali ed ha dimostrato come a partire dalla memorizzazione dei dati delle ricerche utenti per tali strutture sia stato possibile realizzare soluzioni per supportare l'utente nel processo di ricerca. Le soluzioni proposte consistono (i) in un cruscotto informativo per l'analisi delle caratteristiche delle ricerche utenti e (ii) in un sistema di raccomandazione per coadiuvare l'utente, durante la fase di ricerca, nella selezione dei parametri di ricerca da utilizzare.

La memorizzazione delle ricerche utenti è stata realizzata con il database NoSQL MongoDB di cui è stato definito il modello dei dati corrispondente.

Il cruscotto proposto consente di visualizzare statistiche riguardanti la distribuzione dei valori dei parametri di ricerca selezionati dagli utenti nelle ricerche passate. Esso prevede inoltre due modalità di filtraggio dei dati per consentire la visualizzazione di statistiche sulla base delle specifiche esigenze sanitarie espresse dagli utenti.

Il sistema di raccomandazione proposto consiste invece in un sistema ibrido composto da due componenti basate sugli approcci *collaborative filtering* e *association rules-based*. Tale sistema riceve in ingresso la configurazione di ricerca base selezionata dall'utente nella pagina di ricerca, contenente le esigenze di servizi medicali del paziente, e produce in uscita una configurazione di parametri di ricerca avanzata consigliati.

Nonostante raggiungano gli obiettivi prefissati, sia il cruscotto che il sistema di raccomandazione proposti presentano aspetti estendibili o migliorabili negli sviluppi futuri. Le funzionalità del cruscotto potrebbero ad esempio essere estese attraverso l'utilizzo dell'informazione sulla posizione geografica del paziente, memorizzata nel campo `Coordinate_partenza` del modello dei dati proposto; si potrebbero quindi analizzare le ricerche effettuate all'interno di un determinato raggio dalle coordinate geografiche di una zona di interesse. Si potrebbe inoltre introdurre una sezione dedicata alla visualizzazione di informazioni derivanti dall'analisi tramite di regole di associazione, attualmente utilizzata esclusivamente all'interno del sistema di raccomandazione. Importanti miglioramenti si potrebbero invece apportare al sistema di raccomandazione nel caso venisse introdotta

nelle evolutive future dell'applicazione la possibilità di valutare le strutture ospitanti. Si potrebbero quindi utilizzare tali informazioni per raccomandare direttamente una struttura sanitaria all'utente sulla base delle valutazioni fornite dagli altri utenti con esigenze sanitarie similari, come avviene tradizionalmente nell'approccio *collaborative filtering*. Tali valutazioni potrebbero essere inoltre impiegate per migliorare la qualità delle raccomandazioni del sistema proposto, utilizzandole ad esempio in un processo di calibrazione ottimale dei parametri del sistema, quali il valore del parametro k per la selezione delle ricerche *neighbors* o i valori di supporto e confidenza per l'estrazione delle regole di associazione.

Bibliografia

[1] Docker, https://www.docker.com/.
[2] Flask, https://flask.palletsprojects.com/.
[3] Pandas, https://pandas.pydata.org/.
[4] NumPy, https://numpy.org/.
[5] SciPy, https://www.scipy.org/scipylib/index.html/.
[6] Scikit-learn, https://scikit-learn.org/stable/.
[7] MLxtend, https://rasbt.github.io/mlxtend/.
[8] P. Atzeni, S. Ceri, P. Fraternali, S. Paraboschi, R. Torlone, *Basi di dati: modelli e linguaggi di interrogazione*. McGraw-Hill, Milano, Quarta Edizione, 2013.
[9] E. F. Codd, *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM, Volume 13, Numero 6, 1970.
[10] D. Reinsel, J. Gantz, J. Rydning, *The Digitization of the World - From Edge to Core*. IDC Report, Seagate, 2018.
[11] E. Brewer, *Towards Robust Distributed Systems*. Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing, 2000.
[12] MongoDB, Types of NoSQL Databases, https://www.mongodb.com/scale/types-of-nosql-databases/.
[13] MongoDB, https://www.mongodb.com/.
[14] BSON, https://bsonspec.org/.
[15] MongoDB Drivers, https://docs.mongodb.com/drivers/.
[16] MongoDB Atlas, https://www.mongodb.com/cloud/atlas/.
[17] PyMongo, https://pymongo.readthedocs.io/.
[18] Chart.js, https://www.chartjs.org/.
[19] FusionCharts, https://www.fusioncharts.com/fusioncharts/.
[20] D. Dowding, R. Randell, P. Gardner, G. Fitzpatrick, *Dashboards for Improving Patient Care: Review of the Literature*. International Journal of Medical Informatics, Volume 84, 2014.
[21] M. Isazad, A. Ojo, F. Sullivan, *Towards a theoretical model of dashboard acceptance and use in healthcare domain*, 2021.
[22] C. C. Aggarwal, *Recommender Systems: The Textbook*. Springer, First Edition, 2016.
[23] P. Tan, M. Steinbach, A. Karpatne, V. Kumar, *Introduction to Data Mining*. Pearson Education, New York, Second Edition, 2019.
[24] T. N. T. Tran, A. Felfernig, C. Trattner, A. Holzinger, *Recommender systems in the healthcare domain: state-of-the-art and research issues*. Journal of Intelligent Information Systems, 2020.

[25] P. Achananuparp, I. Weber, *Extracting Food Substitutes From Food Diary via Distributional Similarity*, 2016.

[26] S. Bankhele, A. Mhaske, S. Bhat, S. V. Shinde, *A Diabetic Healthcare Recommendation System*. International Journal of Computer Applications, Volume 167, Numero 5, 2017.

[27] I. Kulev, E. Vlahu-Gjorgievska, V. Trajkovik, S. Koceski, *Recommendation Algorithm Based on Collaborative Filtering and its Application in Health Care*. The 10th Conference for Informatics and Information Technology (CIIT 2013), 2013.

[28] F. Narducci, C. Musto, M. Polignano, M. Gemmis, P. Lops, G. Semeraro, *A Recommender System for Connecting Patients to the Right Doctors in the HealthNet Social Network*. Proceedings of the 24th International Conference on World Wide Web (WWW '15 Companion), 2015.

[29] Q. Han, J. Mengxin, I. Martinez de Rituerto de Troya, M. Gaur, L. Zejnilovic, *A Hybrid Recommender System for Patient-Doctor Matchmaking in Primary Care*. IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), 2018.

[30] W. Chen, C. Hsu, Y. Lai, V. Liu, M. Yeh, S. Lin, *Attribute-Aware Recommender System Based on Collaborative Filtering: Survey and Classification*. Frontiers in Big Data, Volume 2, 2020.

[31] X. Amatriain, A. Jaimes, N. Oliver, J. M. Pujol, *Data Mining Methods for Recommender Systems*. Recommender Systems Handbook, Capitolo 2, Springer, Boston, MA, 2011.

[32] C. Desrosiers, G. Karypis, *A Comprehensive Survey of Neighborhood-based Recommendation Methods*. Recommender Systems Handbook, Capitolo 4, Springer, Boston, MA, 2011.

[33] E. Spertus, M. Sahami, O. Buyukkokten, *Evaluating similarity measures: A large-scale study in the Orkut Social Network*. Proceedings of the 2005 International Conference on Knowledge Discovery and Data Mining (KDD-05), 2005.

[34] J. Herlocker, J. A. Konstan, J. Riedl, *An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms*. Information Retrieval Journal, Volume 5, 2002.

[35] G. Adomavicius, A. Tuzhilin, *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions*. IEEE Transactions on Knowledge and Data Engineering, Volume 17, Numero 6, 2005.

[36] Y. Shi, M. Larson, A. Hanjalic, *Collaborative Filtering beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges*. ACM Computing Surveys, Volume 47, Numero 1, 2014.

[37] M. S. Mythili, A. R. Mohamed Shanavas, *Performance Evaluation of Apriori and FP-Growth Algorithms*. International Journal of Computer Applications, Volume 79, Numero 10, 2013.

[38] M. Sumanth Reddy, Ranjith, K. V. Divya, *Association Rule based Recommendation system using Big Data*. International Research Journal of Engineering and Technology (IRJET), Volume 4, Numero 6, 2017.

[39] N. Bendakir, A. Esma, *Using Association Rules for Course Recommendation*. 2006.

[40] W. Lin, S. A. Alvarez, C. Ruiz, *Efficient Adaptive-Support Association Rule Mining for Recommender Systems*. Data Mining and Knowledge Discovery, Volume 6, 2002.