Specifica tecnica

v0.3.0

Farmacode

farmacode.swe.unipd@gmail.com

Registro delle modifiche

Versione	Data	Scrittori	Revisori	Descrizione
0.3.0	2024-03-22	Bomben Filippo		Architettura Front-end
0.2.0	2024-03-20	Bomben Filippo		Tecnologie
0.1.0	2024-03-01	Favaron Riccardo	Bomben Filippo	Struttura iniziale del documento

Indice

1) Introduzione

1.1) Scopo del documento

Il documento riguardante l'analisi dei requisiti è un elemento di fondamentale importanza per i progetti di sviluppo software che voglio rispettare i massimi standard di qualità definiti dall'insegnamento dell'ingegneria del software.
Il presente documento ha lo scopo di fornire una descrizione dettagliata e più precisa possibile riguardanti le linee di massima del prodotto, che comprende i requisiti, così detti, obbligatori, desiderati e opzionali che vanno a rispondere alle necessità del proponente.
Si specializza sull'analisi dei bisogni dell'utente utilizzatore esaminati dallo studio del capitolato e durante i vari incontri con l'azienda proponente volti a tale scopo.
Le richieste del proponente sono, dunque, raccolte e ben identificate nel seguente documento; inoltre, sono classificate secondo le categorie standard di requisiti funzionali, di qualità e di vincolo.
L'analisi dei requisiti compone la pietra portante della progettazione di un sistema software, in quanto esplicita le funzionalità che il prodotto finale deve offrire. È essenziale per i programmatori usufruire di tale documento per assimilare a pieno le necessità dei proponenti di progetto per poi trovare la soluzione che più si sposa a soddisfare le esigenze proposte.
Il documento seguente deve essere il più completo e specifico possibile così da garantire requisiti corretti e che riscoprano tutti gli scenari plausibili per limitare i rischi di progetto ed evitare di inciampare in errori e ritardi che si traducono in costi maggiori.
È utile definire una precisa e formale rappresentazione grafica dei requisiti e degli attori in gioco grazie ai diagrammi dei casi d'uso, così da facilitare la comprensione a tutti.

1.1.1) Struttura logica casi d'uso

I casi d'uso descritti in questo documento hanno una precisa struttura logica descritta dal seguente modello:

- Titolo: Titolo del caso d'uso;
- Figura;
- Attori coinvolti: Il soggetto che esegue una determinata azione;
- Precondizioni: Lo stato del sistema prima del caso d'uso;
- Postcondizioni: Lo stato del sistema dopo l'esecuzione dello scenario descritto dal caso d'uso;
- Scenario principale: Descrizione dettagliata delle azioni svolte dall'attore durante il caso d'uso, intermedio tra le ipotesi e i risultati;
- Estensioni (se presenti): Possibili estensioni derivanti dal caso d'uso;
- Generalizzazioni (se presenti): Generalizzazioni di attori e casi d'uso.

1.2) Scopo del prodotto

Il progetto ha lo scopo di realizzare un *sistema di raccomandazione* con relativa interfaccia web che guidi le attività dell'azienda utilizzatrice del prodotto finale; suggerendo a quali clienti rivolgere le singole attività di marketing e commerciali.
Dall'interfaccia utente del sistema software sarà possibile selezionare uno specifico cliente e visualizzare i prodotti da lui acquistati e quelli che il sistema ha individuato come raccomandati. Inoltre selezionato un articolo il sistema suggerirà a quali clienti proporli, selezionandoli in base a quanto probabile siano interessati a quel determinato prodotto. I vari prodotti possono essere filtrati per categoria così da facilitarne la ricerca e restringere il campo di soluzione.
Ogni risultato restituito dal sistema di raccomandazione è classificabile tramite un feedback

così da poter eventualmente correggere il tiro dell'algoritmo che ha fornito l'esito della suggerimento.
L'utente amministratore avrà poi la possibilità di usufruire di altre funionalità dedicate, come ad esempio visualizzare la cronologia delle ricerche.

1.3) Glossario

Al fine di evitare eventuali equivoci o incomprensioni riguardo la terminologia utilizzata all'interno di questo documento, si è deciso di adottare un Glossario, con file apposito, in cui vengono riportate tutte le definizioni rigogliose delle parole ambigue utilizzate in ambito di questo progetto. Nel documento appena descritto verranno riportati tutti i termini definiti nel loro ambiente di utilizzo con annessa descrizione del loro significato.
La presenza di un termine all'interno del Glossario è evidenziata dal *colore blu*.

1.4) Maturità e miglioramenti

Questo documento è stato realizzato utilizzando un approccio incrementale, con lo scopo di semplificare i cambiamenti nel tempo in base alle reciproche esigenze decise da entrambi le parti, ovvero membri del gruppo di progetto e azienda proponente. Pertanto non può essere considerato esaustivo e completo, ma in costante miglioramento.

1.5) Riferimenti

1.5.1) Riferimenti normativi

- Norme di Progetto v.2.0.0;
- Capitolato C2: Sistemi di raccomandazione
  https://www.math.unipd.it/~tullio/IS-1/2023/Progetto/C2.pdf;
- Regolamento progetto ditattico
  https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/PD2.pdf.

1.5.2) Riferimenti informativi

- I diagrammi dei casi d'uso (UML) (slide del corso di Ingegneria del Software)
  https://www.math.unipd.it/~rcardin/swea/2022/Diagrammi%20Use%20Case.pdf.
- Progettazione: I pattern architetturali (slide del corso di Ingegneria del Software)
  https://www.math.unipd.it/~rcardin/swea/2022/Software%20Architecture%20Patterns.pdf
- Verifica e validazione: analisi statica (T10) (slide del corso di Ingegneria del Software)
  https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T10.pdf
- Verifica e validazione: analisi dinamica aka testing (T11) (slide del corso di Ingegneria del Software)
  https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T11.pdf
- Programmazione: SOLID programming (slide del corso di Ingegneria del software)
  https://www.math.unipd.it/~rcardin/swea/2021/SOLID%20Principles%20of%20Object-Oriented%20Design_4x4.pdf

2) Tecnologie

Questa sezione serve a fornire una panoramica generale sulle tecnologie adottate per il progetto. Vengono riportate sottoforma di tabelle le diverse tecnologie, sia per la codifica che per l'analisi e il test del codice. Ogni tabella è formata da tre colonne che riportanto:

- La tecnologia utilizzata (il nome del linguaggio/framework/strumento);
- La descrizione del ruolo che la tecnologia ha avuto all'interno del progetto;
- La versione della tecnologia usata.

2.1) Tecnologie per la codifica

Le tecnologie per la codifica del progetto riguardano i vari linguaggi utilizzati per la scrittura del codice, le librerie e framework adottate per facilitare l'implementazione delle funzionalità e gli strumenti utilizzati per la gestione della codifica del progetto.
La scelta di determinate tecnologie è il risultato di ricerche nelle quali abbiamo cercato di capire i vantaggi che avrebbero potuto portare al progetto.

2.1.1) Linguaggi

Tecnologia	Descrizione	Versione
Linguaggi		
HTML	Linguaggio di markup utilizzato per la creazione e gestione della struttura delle pagine web. La sua funzione è quella di "scheletro" delle pagine e del contenuto in esse.	5
CSS	Linguaggio per la formattazione dei documenti HTML, il suo scopo è di gestire lo stile e il design del sito.	3
JavaScript	Linguaggio di programmazione per la gestione degli eventi dell'utente e per la comunicazione con l'API.	TD
Python	Linguaggio di programmazione usato per la creazione del sistema di raccomandazione.	3.11.5
SQL	Linguaggio di interrogazione per la creazione e gestione del database.	TD

Tabella 1: Linguaggi

2.1.2) Librerie e framework

Tecnologia	Descrizione	Versione
Librerie e framework		
Pandas	Libreria per Python utilizzata per la manipolazione e l'analisi dei dati	TD

Surprise	Libreria per Python utilizzata per semplificare lo sviluppo di sistemi di raccomandazione e valutare le prestazioni di algoritmi di filtraggio collaborativo	TD
React.js	Libreria JavaScript utilizzata per semplificare lo sviluppo front-end, consentendo una gestione modulare delle componenti grafiche.	TD
PrimeReact	Suite per l'User Interface per React.js che utilizza componenti già definiti e ben strutturati.	TD
Express	Libreria di JavaScript utilizzata per lo sviluppo back-end del sito	TD
NumPy	Libreria per Python utilizzata per la manipolazione di array e matrici multidimensionali.	TD
Tailwind CSS	Framework per css utilizzato per lo sviluppo di interfacce utente.	3

Tabella 2: Librerie e framework

2.1.3) Strumenti e servizi

Tecnologia	**Descrizione**	**Versione**
Strumenti e servizi		
MySQL	RDBMS per la creazione e gestione dei database in SQL.	TD
Node.js	Ambiente di runtime open-source per l'esecuzione di codice JavaScript lato server tramite appositi script.	TD
NPM	Gestore di pacchetti (Node Package Manager) per JavaScript all'interno di Node.js.	TD
VS Code	IDE di programmazione gratuito ricco estensioni esterne.	TD
Docker	Creatore di ambienti di sviluppo tramite container per la gestione delle dipendenze.	TD
Git	Sistema di controllo e versionamento utilizzato per la gestione del codice.	TD
Anaconda	Gestore e distributore per Python dei pacchetti per la gestione delle versioni.	TD

Tabella 3: Strumenti e servizi

2.2) Tecnologie per l'analisi del codice

2.2.1) Analisi statica

Tecnologia	Descrizione	Versione
Analisi statica		
Ruff	Strumento per l'analisi statica del codice Python, individua errori, violazioni delle convenzioni di codifica e altri problemi nel codice sorgente.	0.3.3
ESLint	Strumento utilizzato per l'analisi statica del codice JavaScript e TypeScript, che aiuta a individuare gli errori di codice e le pratiche non ottimali.	8.57.0

Tabella 4: Analisi statica

2.2.2) Analisi dinamica

Tecnologia	Descrizione	Versione
Analisi dinamica		
Pytest	Framework di test open-source per Python. Offre un'ampia gamma di funzionalità per la scrittura e l'esecuzione di test unitari, di integrazione funzionali	8.0.x
Jest	Framework di test basato su JavaScript con funzionalità di creazione di mock e il testing del codice in modo asincrono.	29.7.x
GitHub Action	Servizio di CI/CD per automatizzare il processo di build, test e deploy del progetto software.	/

Tabella 5: Analisi dinamica

3) Architettura

Il gruppo, durante la fase di progettazione, ha decisio di adottare un'architettura a microservizi. La scelta di questa precisa architettura è ricaduta per la natura ben separata dei ruoli delle varie componenti del progetto. Per questo motivo la comunicazione tra i vari servizi avviene tramite API Rest, sviluppate sia in Python con Flask che in JavaScript attraverso la libreria Express. Abbiamo quindi deciso di dividere nel seguente modo il sistema:

- Fron-end, la parte di presentazione del sistema, che rappresenta l'interfaccia utente a cui l'utente può accedervi attraverso il browser. La parte client è stata sviluppata tramite HTML, CSS e JavaScript con la libreria React;
- Back-end, la parte logica del progetto che sfruttando API Rest, creano interazione tra il Database e l'algoritmo in maniera che comunichino correttamente e riescano a recuperare tutti i dati necessari.

3.1) Architettura Front-end

L'architettura front-end del prodotto sfrutta alcuni dei design pattern più comuni della libreria React, rimodellati in base alle esigenze della specifica situazione e del progetto.
Abbiamo cercato, per quanto possibile, di separare il più possibile i compiti tra le varie componenti, per semplificare e gestire al meglio i vari stati dell'applicazione.

3.1.1) Diagramma delle classi

In questa sezione vengono descritte le varie pagine attraverso la convenzione UML per la rappresenta delle classi.
Alcune di queste pagine avranno delle componenti usate all'interno di esse, per evitare ridondanza, la descrizione delle pagine e la descrizione delle componenti saranno separate, in questo modo il diagramma sarà più semplice da leggere evitando un eccessivo caos.
Allo scopo di rendere il tutto più chiaro possibile, a seguito di ongi diagramma ci sarà una breve spiegazione sulla funzionalità della pagina/componente.

3.1.2) Pagine

3.1.2.1) Login

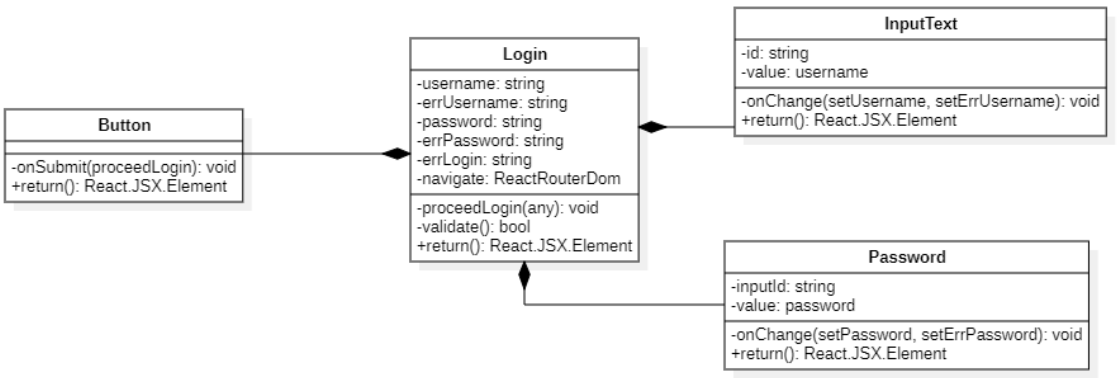

Figura 1: Login

3.1.2.2) Ricerca

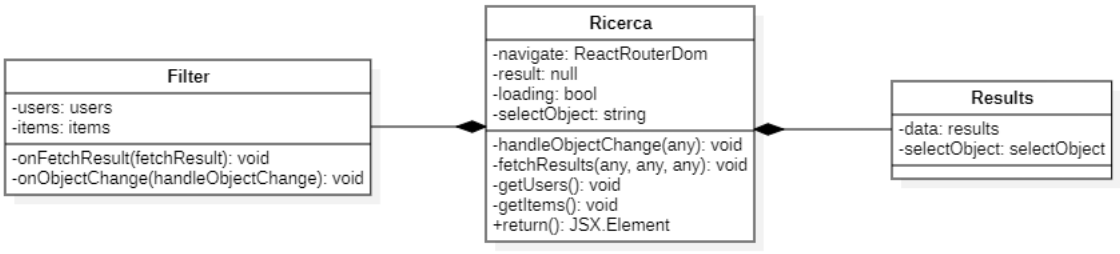

Figura 2: Ricerca

3.1.2.3) Clienti

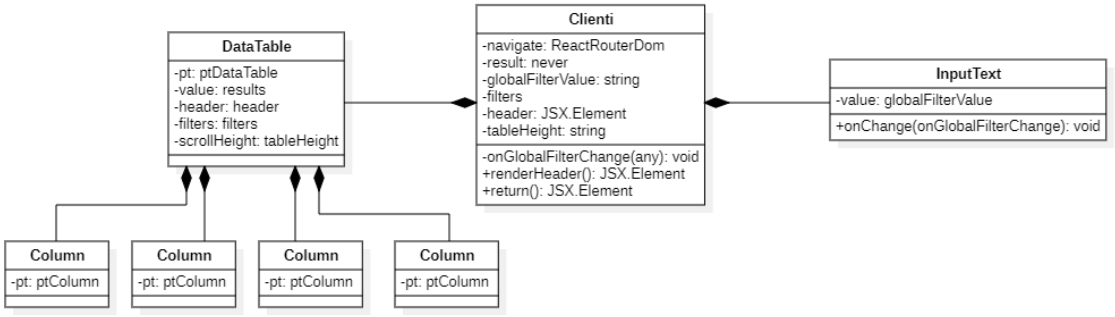

Figura 3: Clienti

3.1.2.4) Prodotti

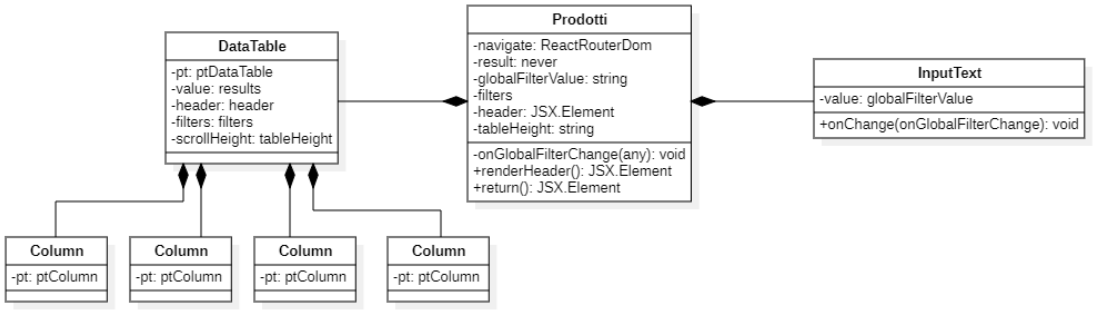

Figura 4: Prodotti

3.1.2.5) Profilo

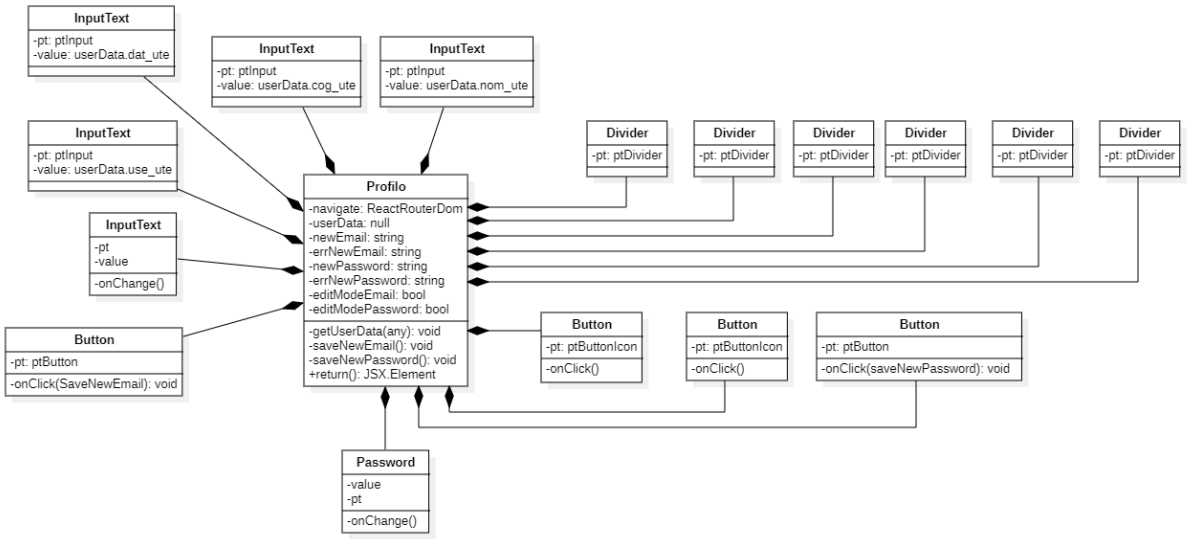

Figura 5: Profilo

3.1.3) Componenti

3.1.3.1) Filtri

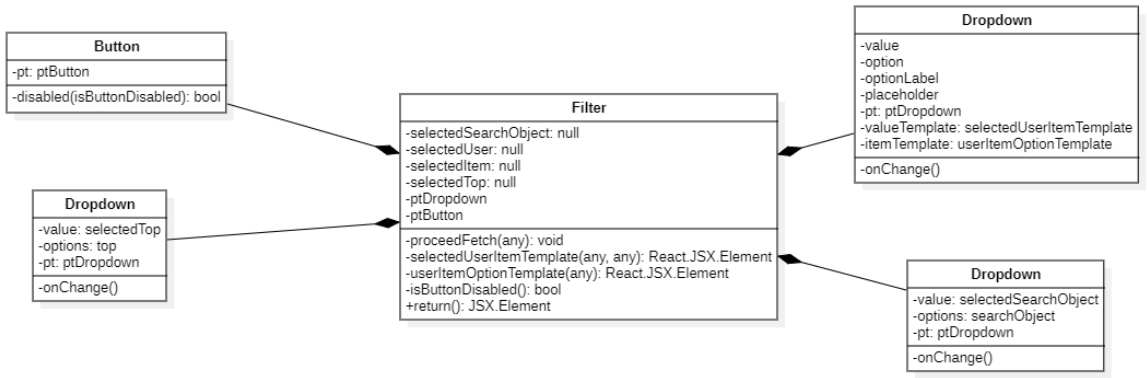

Figura 6: Filtri

3.1.3.2) Header

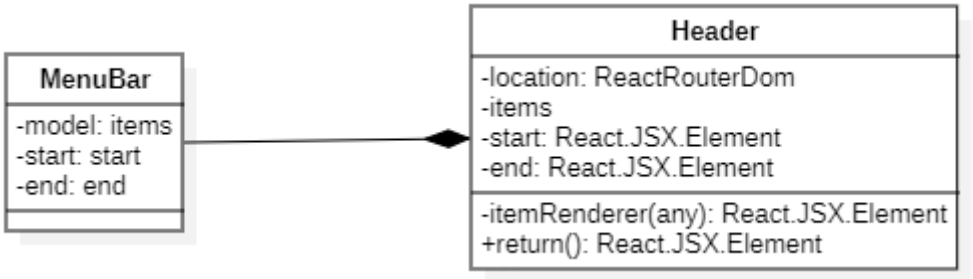

Figura 7: Header

3.1.3.3) Results

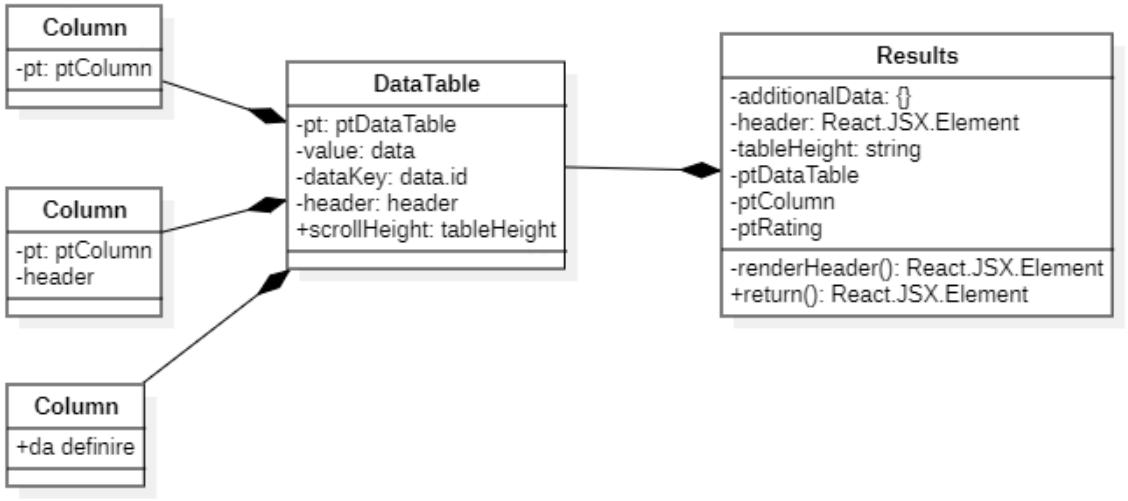

Figura 8: Results

3.2) Architettura Back-end

4) Stato requisiti funzionali

4.1) Tabella requisiti funzionali

4.2) Grafico requisiti funzionali

5) Elenco delle immagini

6) Elenco delle tabelle