Specifica tecnica

v0.5.0

Farmacode

farmacode.swe.unipd@gmail.com

Registro delle modifiche

Versione	Data	Scrittori	Revisori	Descrizione
0.5.0	2024-04-14	Passarella Alessandro		Stesura documentazione API
0.4.0	2024-03-23	Pandolfo Mattia		Prima stesura architettura Back-end
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	2.1.1

Surprise	Libreria per Python utilizzata per semplificare lo sviluppo di sistemi di raccomandazione e valutare le prestazioni di algoritmi di filtraggio collaborativo	1.1.3
React.js	Libreria JavaScript utilizzata per semplificare lo sviluppo front-end, consentendo una gestione modulare delle componenti grafiche.	18.2.0
PrimeReact	Suite per l'User Interface per React.js che utilizza componenti già definiti e ben strutturati.	10.5.1
Express	Libreria di JavaScript utilizzata per lo sviluppo back-end del sito	4.18.2
Flask	Framework per lo sviluppo di applicazioni web in Python che fornisce strumenti per la gestione delle richieste HTTP	3.0.x
NumPy	Libreria per Python utilizzata per la manipolazione di array e matrici multidimensionali.	1.26.0
PyTorch	Framework per l'apprendimento automatico basato su Python che offre tensori potenti, grafi computazionali dinamici e autograd.	2.2.2
Tailwind CSS	Framework per css utilizzato per lo sviluppo di interfacce utente.	3.4.1
Axios	Libreria JavaScript utilizzata per effettuare richieste HTTP sia lato client che lato server	1.6.8

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.	2.18.1
Node.js	Ambiente di runtime open-source per l'esecuzione di codice JavaScript lato server tramite appositi script.	18.16.1
NPM	Gestore di pacchetti (Node Package Manager) per JavaScript all'interno di Node.js.	9.5.1
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.	27.5.1
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 ogni diagramma ci sarà una breve spiegazione sulla funzionalità della pagina/componente. Inoltre abbiamo preso dalla libreria PrimeReact i seguenti componenti:

- Password;
- InputText;
- Button;
- Multiselect;
- DataTable;
- Column;
- Dialog;
- Divider;
- Dropdown;
- MenuBar;
- Rating.

Di tali componenti non ci saranno diagrammi delle classi poiche' importati da una librei esterna.

3.1.2) Pagine

3.1.2.1) Clienti

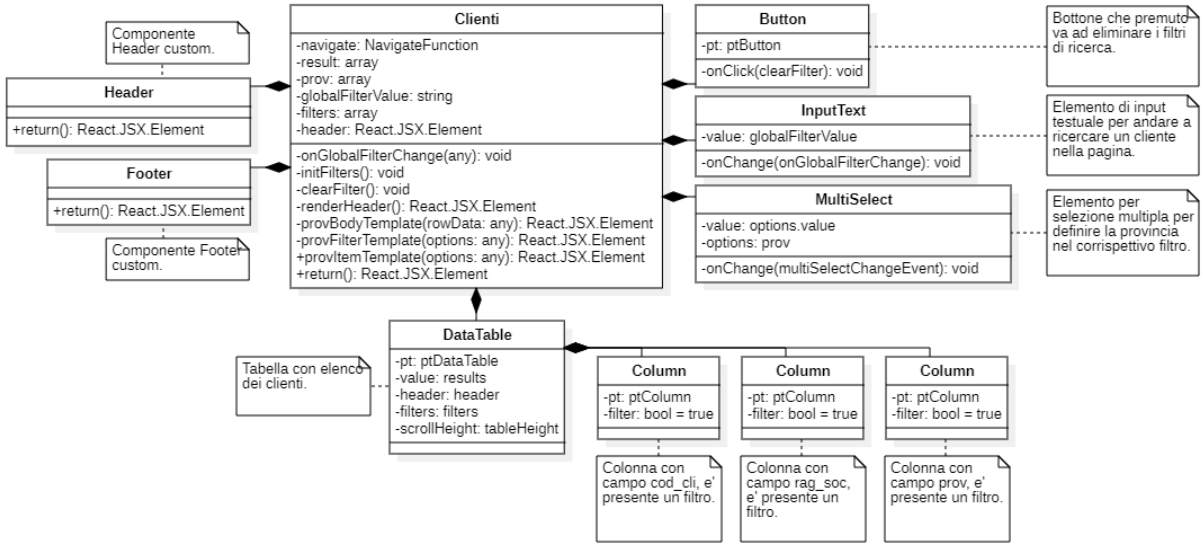

Figura 1: Clienti

Clienti è una pagina consultiva, dove vengono visualizzati tutti i clienti memorizzati nel database.
È composta da una sezione header dove è possibile filtrare i clienti e da una tabella in cui si possono colsultare i clienti.

3.1.2.2) Cronologia

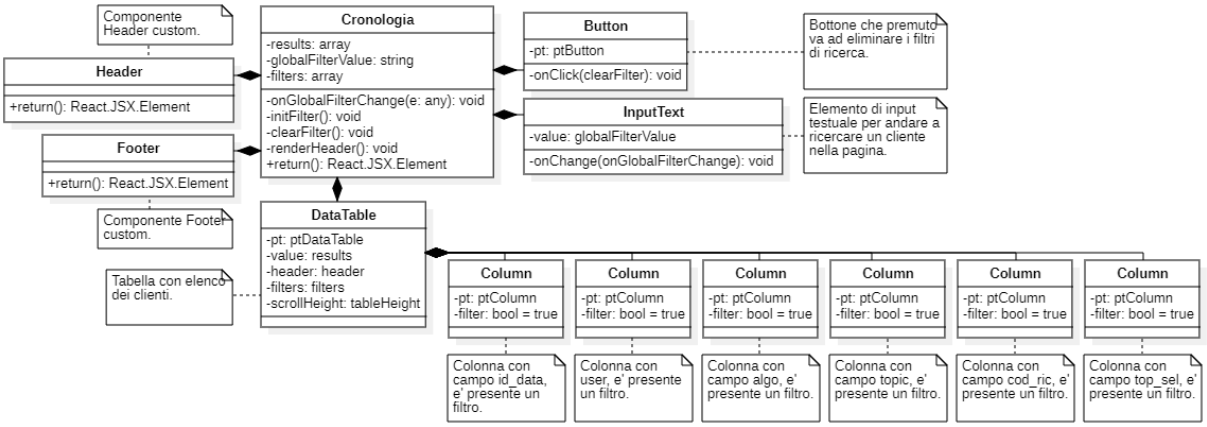

Figura 2: Cronologia

Cronologia è una pagina consultiva, dove viene visualizzata la cronologia delle ricerche o raccomandazioni.
È composta da una sezione header contentente un *InputText* per ricerche nella pagina e da una tabella in cui poter visualizzare e ulteriormente filtrare la ricerca.

3.1.2.3) Feedback

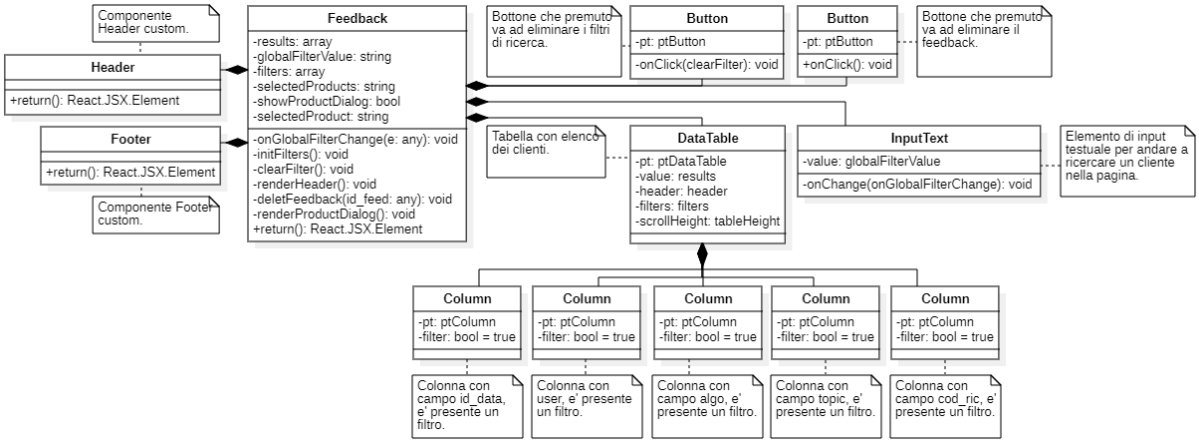

Figura 3: Feedback

Feedback è una pagina consultiva, dove vengono visualizzati i feedback.
È composta da una sezione header contentente un *InputText* per ricerche nella pagina, un *Button* per andare a cancella eventuali filtri e un *Button* per andare ad eliminare il feedback. Inoltre e' presente una tabella in cui poter visualizzare e ulteriormente filtrare i feedback.

3.1.2.4) Login

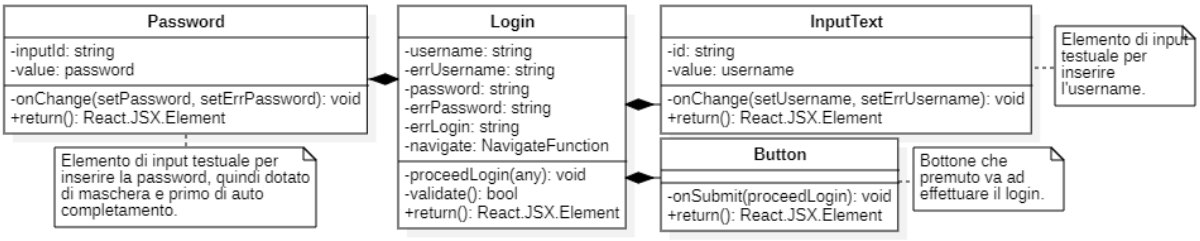

Figura 4: Login

login è la pagina di accesso al sito.
E' composta da un *InputText*, usato per ricevere il dato username per effettuare il login, il componente *Password* per l'immissione della password. Infine *Button*, utilizzato per inviare tutte le informazioni compilate nel form e effettuare concretamente il login per accedere alla pagina principale.
È l'unica pagina a non avere ne' header ne' footer che la compongano.

3.1.2.5) Pagina Non Trovata

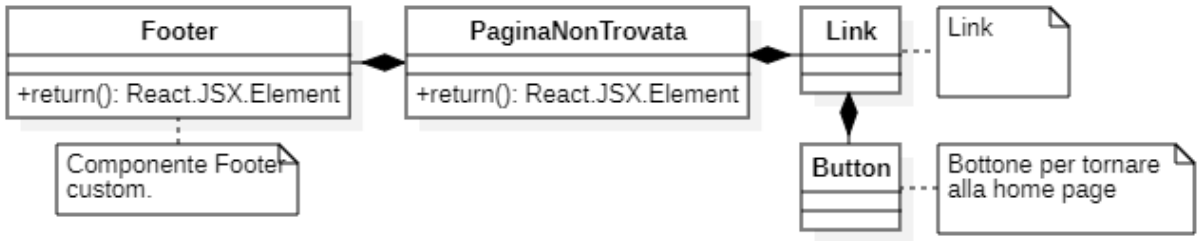

Figura 5: Pagina Non Trovata

PaginaNonTrovata e' la pagina visualizzata in caso di errore nella navigazione.
È composta da un *Button* che permette di tornare alla pagina Ricerca. In questa pagina è presente solo il footer.

3.1.2.6) Prodotti

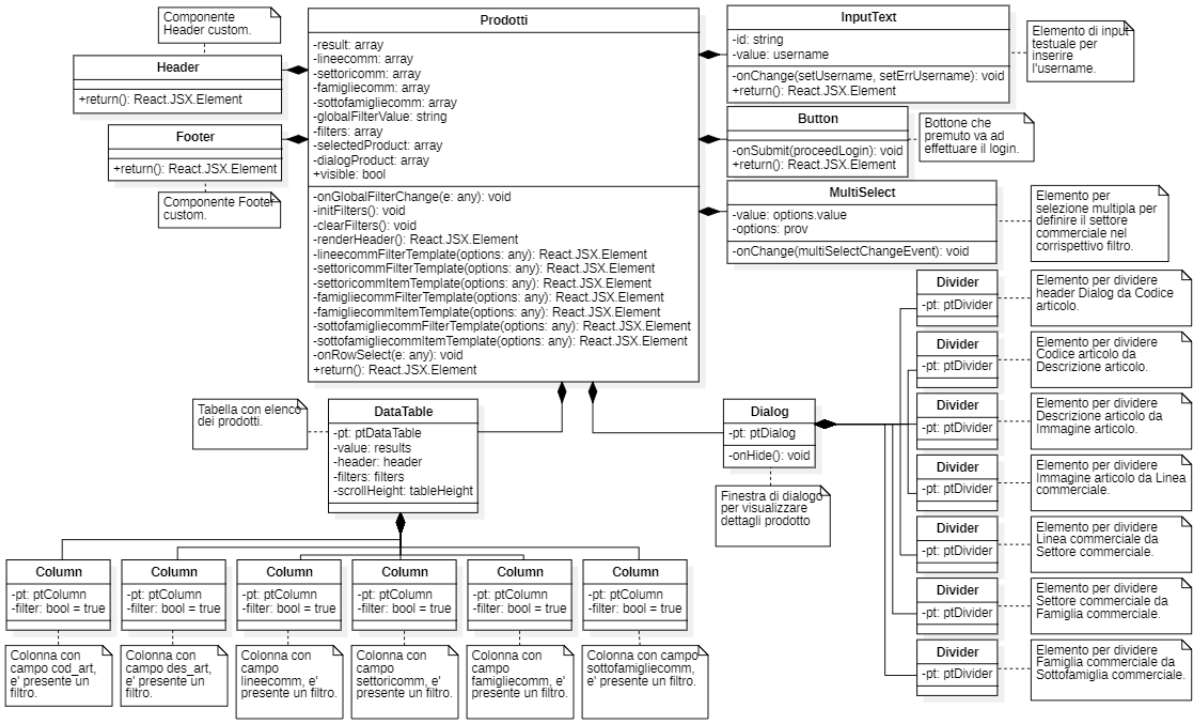

Figura 6: Prodotti

Prodotti è una pagina molto analoga a Clienti, dove vengono visualizzati tutti i prodotti presenti nel Database.
È composta da un header con un *InputText* per effettuare una ricerca e un *Button* per cancellare filtri applicati. Nella parte sottostante è possibile visualizzare, ed ulteriormente filtrare, tutti i prodotti grazie ad una tabella.
Inoltre premendo sopra un prodotto grazie al componente *Dialog* si potranno visualizzare i dettagli in una finestra separata.

3.1.2.7) Profilo

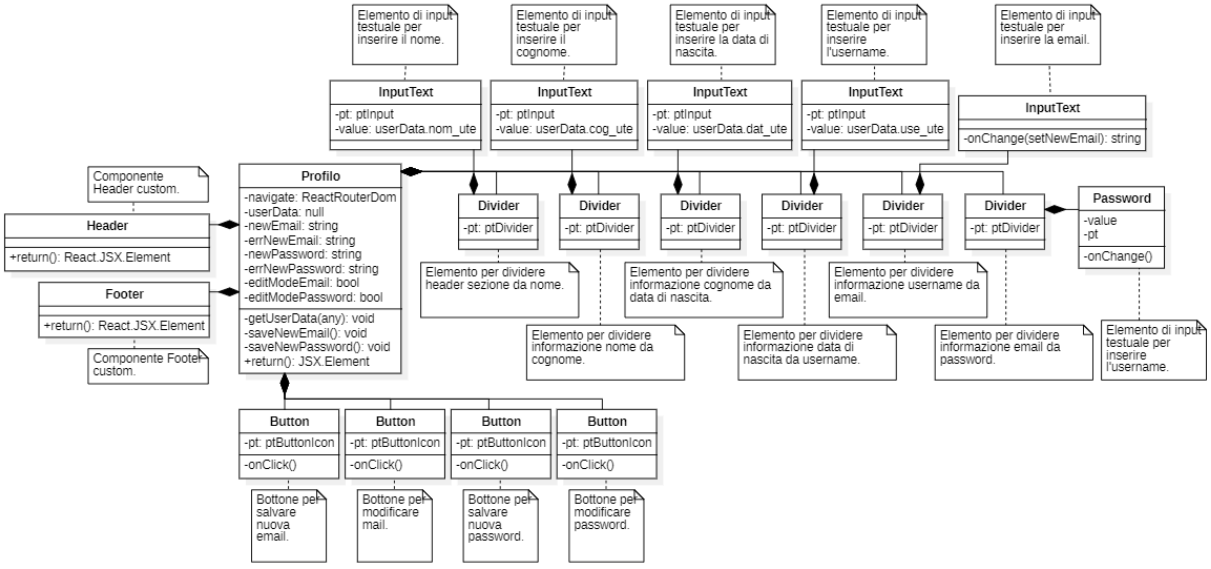

Figura 7: Profilo

Profilo è la pagina in cui visualizzare e modificare le informazioni dell'utente.
È composta da una serie di *InputText* in cui visualizzare e modificare le informazioni oltre a *Button* per confermare o meno le modifiche.

3.1.2.8) Ricerca

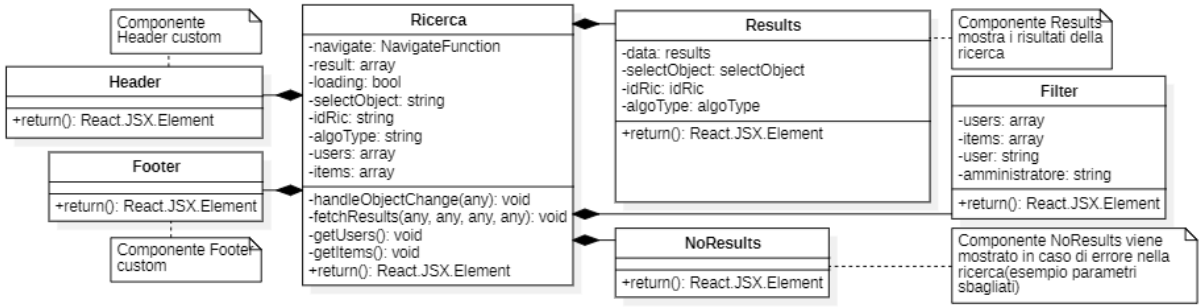

Figura 8: Ricerca

La pagina di ricerca è il core del progetto, in questa pagina si fanno le ricerche per le raccomandazioni in base ai criteri di scelta (prodotto per clienti o cliente per prodotti).
La componente *Filter* è quella che va a impostare la query per il recupero, una volta impostata correttamente, dei risultati di raccomandazione del modello. Quindi in base a come viene impostati i vari criteri del filtro, cambia anche la query al database.
Results invece si occupa di mostrare i dati recuperati, renderizzandoli a schermo all'interno di una tabella. La grandezza della tabella e direttamente correlata al numero di risultati che l'utente ha impostato in *Filters*.

3.1.3) Componenti

3.1.3.1) Filtri

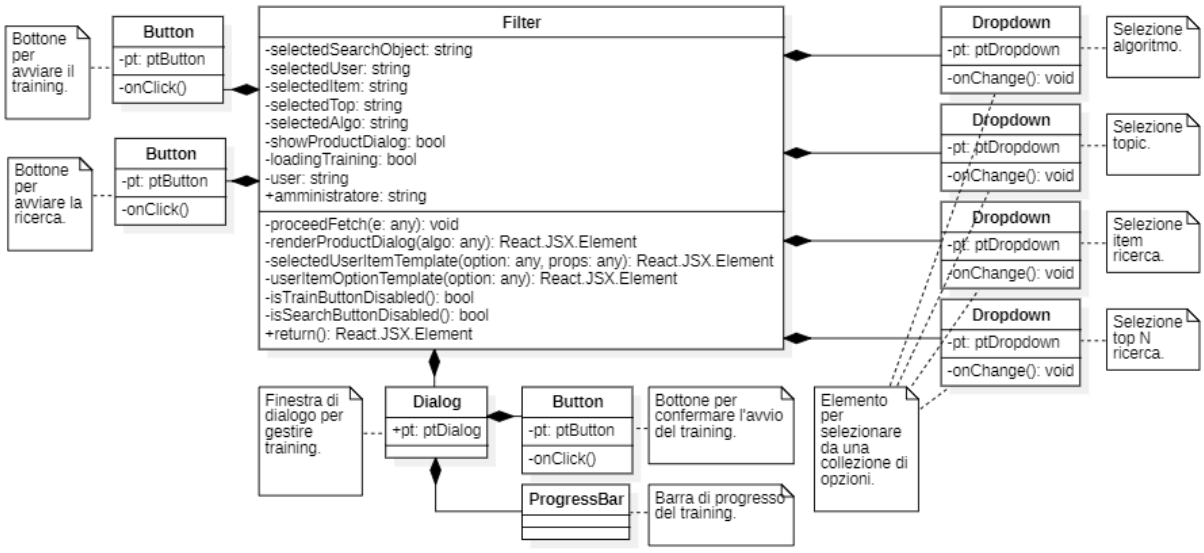

Figura 9: Filtri

Filtri è la componente per gestire il filtri nella pagina di Riceca.
È composta da una serie di *Dropdown* per selezionare opzioni di ricerca e *Button* per confermare le scelte. Inoltre presenta una finestra di Dialogo per gestire il training.

3.1.3.2) Footer

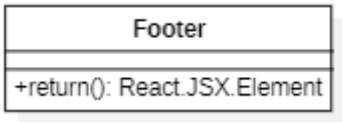

Figura 10: Footer

Footer è la componente che contiene informazioni sul progetto: l'anno di sviluppo, il proponente e un riferimento al gruppo di lavoro.

3.1.3.3) Header

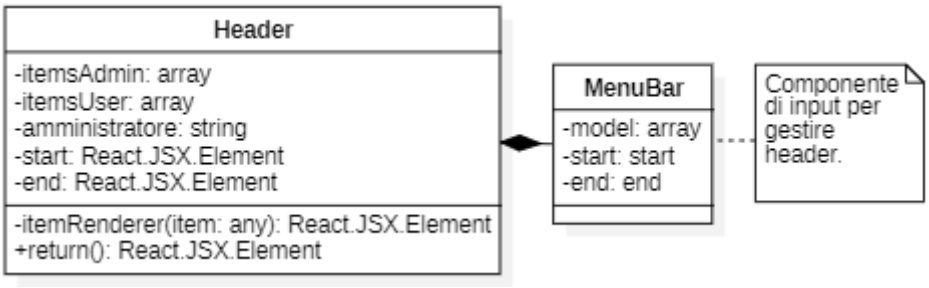

Figura 11: Header

La componente Header contiene il menù di navigazione.
È semplicemente composta da *MenuBar*, importato da PrimeReact, per visualizzare tutte le voci del menù.

3.1.3.4) No Results

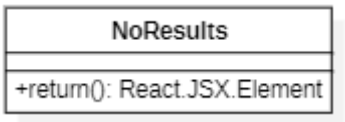

Figura 12: No Results

NoResults è una semplice componente richiamata in caso di ricerca senza risultati o ricerca non effettuata.

3.1.3.5) Results

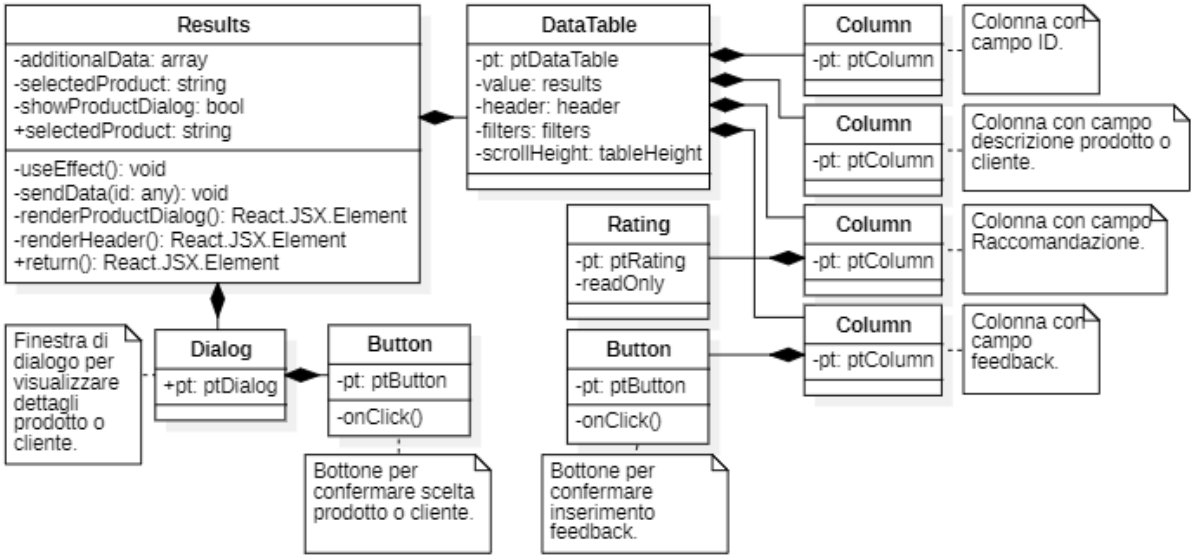

Figura 13: Results

La componente Results viene utilizzata per visualizzare i sultati della raccomandazione.
È composta da una tabella in cui visualizzare e filtrare il risultato della raccomandazione, inoltre di ogni elemente è possibile visualizzare ulteriori dettagli grazie ad una finestra di dialogo visualizzabile con la pressione su uno specifico elemento.

3.2) Architettura Back-end

3.2.1) Introduzione

3.2.2) Schema base di dati (AGGIUNGERE TABELLA FEEDBACK)

In questa sezione, viene presentato lo schema di base di dati realizzato con MySQL, relativo all'architettura back-end del servizio descritto. Descriviamo più nel dettaglio questa composizione:

- **ute**, rappresentante i singoli utenti, comprensiva di:
  1. un username univoco;
  2. il nome dell'utente;
  3. il cognome;
  4. la data di nascita;
  5. una mail univoca;
  6. una password;
  7. l'informazione sull'essere un amministratore o meno.
- **prov**, rappresentante le provincie italiane, comprensiva di:
  1. il codice identificativo univoco della provincia;
  2. il nome della provincia.
- **anacli**, rappresentante i clienti, comprensiva di:
  1. un codice identificativo univoco;
  2. la ragione sociale;
  3. il codice della provincia di appartenenza (chiave esterna). ??????
- **linee_comm**, rappresentante la linea dei prodotti, comprensiva di:
  1. un codice identificativo univoco;

2. la linea del prodotto.

- **settori_comm**, rappresentante il settore dei prodotti, comprensiva di:
  1. un codice identificativo univoco;
  2. il settore del prodotto.
- **famiglie_comm**, rappresentante le famiglie dei prodotti, comprensiva di:
  1. un codice identificativo univoco;
  2. la famiglia del prodotto.
- **sottofamiglie_comm**, rappresentante la sottofamiglia dei prodotti, comprensiva di:
  1. un codice identificativo univoco;
  2. la sottofamiglia del prodotto.
- **anaart**, rappresentante i singoli prodotti, comprensiva di:
  1. un codice identificativo univoco;
  2. la descrizione dell'articolo;
  3. la linea del prodotto (chiave esterna);
  4. il settore del prodotto (chiave esterna);
  5. la famiglia del prodotto (chiave esterna);
  6. la sottofamiglia del prodotto (chiave esterna).

3.2.3) Algoritmo di raccomandazione

3.2.3.1) Diagramma delle classi

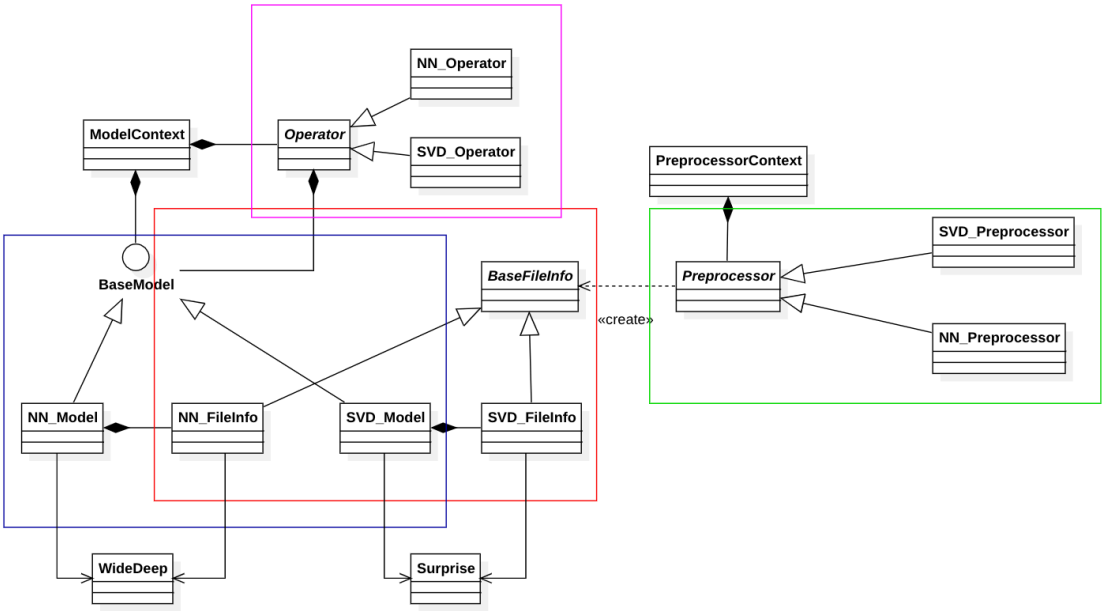

Figura 14: Diagramma algoritmo (totale)

Descrizione:

Pattern:

3.2.3.2) Componenti:

3.2.3.2.1) Preprocessor

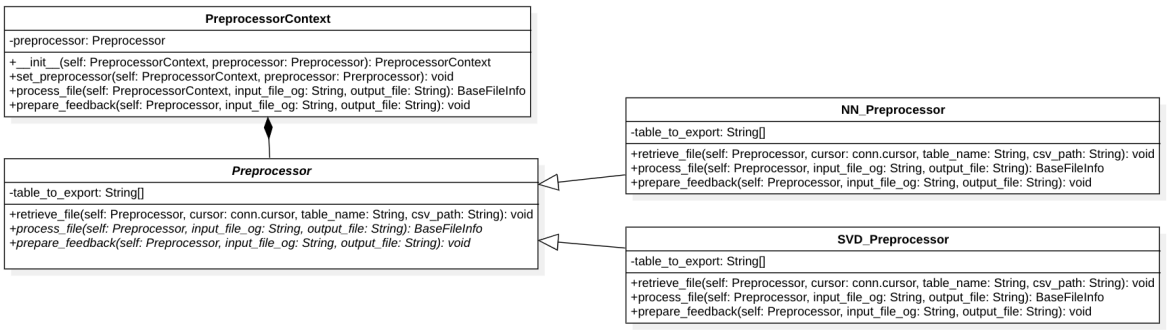

Figura 15: Results

Descrizione:
La classe Preprocessor è una classe astratta che fornisce un'interfaccia per processare dati in diversi modi. È progettata per essere una classe base da cui ereditano altre classi che implementano metodi specifici di preprocessing. La struttura di base della classe è progettata per essere flessibile e consentire l'estensione per gestire diversi tipi di dati e metodi di preprocessing. Le classi SVD_Preprocessor e NN_Preprocessor estendono la classe Preprocessor, ereditano infatti le funzionalità di base e forniscono implementazioni specifiche per il preprocessing dei dati utilizzando due diversi approcci, rispettivamente il metodo Singular Value Decomposition (SVD) e neural network (NN). La classe PreprocessorContext infine utilizza Preprocessor come parte del suo funzionamento. Essa fornisce un "contesto" per il preprocessing dei dati, consentendo di cambiare facilmente il tipo di preprocessing senza dover modificare il codice che lo utilizza.

Metodi:

- Preprocessor:
  1. 'retrieve_file' : Un metodo astratto che prende un cursore SQL, il nome di una tabella e un percorso per un file CSV. Esegue una query SQL per estrarre i dati dalla tabella e scrive i risultati in un file CSV;
  2. 'process_file' : Un metodo astratto che prende un percorso del file di input e un percorso del file di output. È responsabile di processare il file di input in base alle esigenze specifiche dell'algoritmo e salvarlo nel file di output;
  3. 'prepare_feedback' : Un metodo astratto simile a process_file, ma specificamente progettato per preparare i dati di feedback.
- PreprocessorContext:
  1. 'set_preprocessor' : Imposta il preprocessor da utilizzare;
  2. 'process_file' : Prende un percorso del file di input e un percorso del file di output. Utilizza il preprocessor impostato per elaborare il file di input e salvarlo nel file di output;
  3. 'prepare_feedback' : Simile a process_file, ma specifico per preparare i dati di feedback.

  I metodi di SVD_Preprocessor e NN_Preprocessor sono semplicemente delle implementazione dei metodi di Preprocessor, rispettivamente per Singular Value Decomposition (SVD) e neural network (NN).

3.2.3.2.2) FileInfo

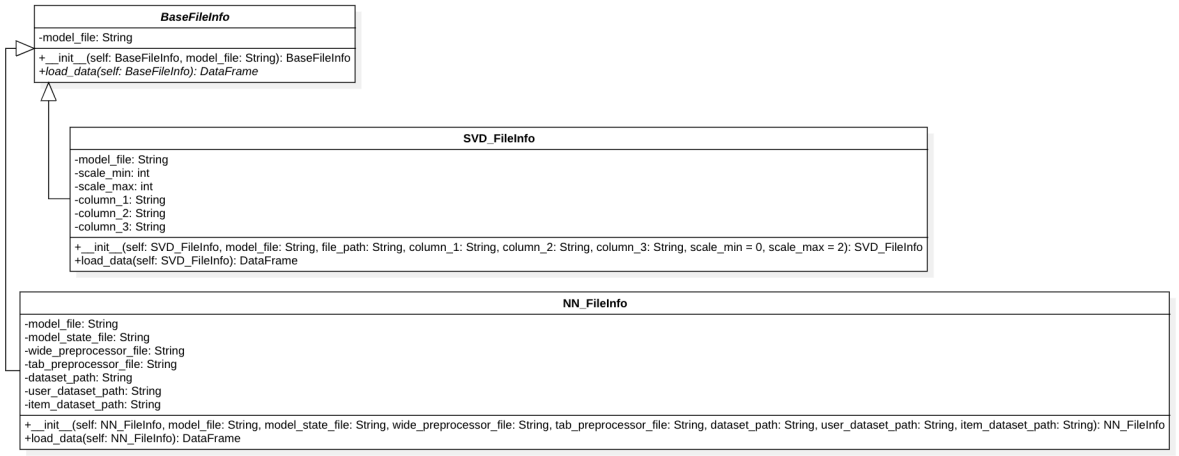

Figura 16: FileInfo

Descrizione:
La classe FileInfo fornisce un'astrazione di base per caricare dati da file, indipendentemente dal loro scopo specifico, offrendo funzionalità di base come l'apertura e la lettura dei file. Le sottoclassi SVD_FileInfo e NN_FileInfo ereditano dalla classe astratta, fornendo implementazioni specifiche per il loro scopo, preparando, organizzando ed interpretando i dati, rispettivamente, per l'analisi SVD o per l'addestramento di neural network.

Metodi:

- BaseFileInfo:
  1. 'load_data' : Un metodo astratto per il caricamento dei dati da file.
- 'NN_FileInfo' :
  1. 'load_data' : Implementazione di 'load_data' di BaseFileInfo, carica i dati dal dataset generale specificato nel percorso dataset_path utilizzando pandas, restituisce i dati sotto forma di DataFrame, utile per modelli di rete neurale che richiedono dati in formato tabellare per l'addestramento.
- 'SVD_FileInfo' :
  1. 'load_data' : Implementazione di 'load_data' di BaseFileInfo, carica i dati dal file di dati specificato nel percorso file_path utilizzando pandas, definisce una scala di valutazione dei dati utilizzando il modulo Reader e carica i dati in un oggetto Dataset, selezionando solo le colonne specificate, questo è utile per modelli basati su decomposizione singolare che operano su dati in formato tabellare.

3.2.3.2.3) Model

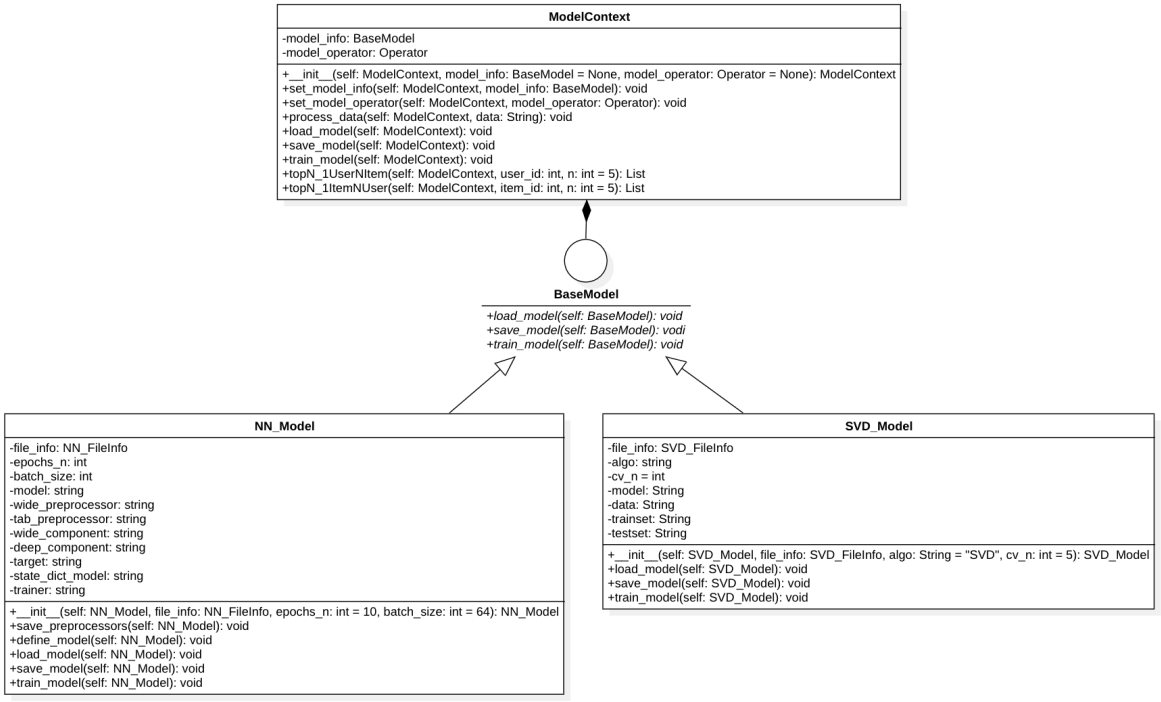

Figura 17: Model

Descrizione:
La classe BaseModel è una classe astratta che definisce i metodi per caricare, salvare e allenare i modelli. SVD_Model è una classe che estende BaseModel, è progettata specificamente per modelli basati su decomposizione ai valori singoli (SVD) e gestisce il caricamento, il salvataggio e l'addestramento di un modello SVD utilizzando la libreria Surprise in Python. NN_Model, invece, è un'altra classe che estende BaseModel, ma è orientata verso modelli di raccomandazione basati su reti neurali. Questa classe utilizza PyTorch per definire, addestrare e salvare modelli neurali; essa implementa la logica per gestire il caricamento e il salvataggio dei pre-elaboratori, la definizione dell'architettura del modello, il caricamento e il salvataggio del modello stesso e il suo addestramento. Infine, ModelContext agisce come un mediatore che connette queste classi di modelli con il mondo esterno. Astrae i dettagli specifici del trattamento del modello e fornisce un'interfaccia unificata per il caricamento, il salvataggio e l'addestramento dei modelli e delega le operazioni effettive alla classe di modello appropriata (SVD_Model o NN_Model) in base all'operatore di modello fornito.

Metodi:

- BaseModel:
  1. 'load_model' : Metodo astratto responsabile del caricamento di un modello;
  2. 'save_model' : Metodo astratto responsabile del salvataggio di un modello;
  3. 'train_model' : Metodo astratto responsabile dell'addestramento di un modello.
- SVD_Model:
  1. 'load_model' : Implementazione di 'load_model' di BaseModel, carica un modello da un file se esiste, altrimenti inizializza un modello SVD;
  2. 'save_model' : Implementazione di 'load_model' di BaseModel, salva il modello in un file;

3. ‘train_model’ : Implementazione di ‘load_model’ di BaseModel, carica i dati, carica o inizializza il modello SVD e, se il file del modello non esiste, esegue il training del modello sui dati caricati. Successivamente, salva il modello addestrato.

- NN_Model:
  1. ‘save_preprocessors’: Salva i preprocessori in file;
  2. ‘define_model’ : Definisce l’architettura del modello di rete neurale;
  3. ‘load_model’ : Implementazione di ‘load_model’ di BaseModel, carica un modello pre-addestrato e i preprocessori se esistono, altrimenti definisce il modello;
  4. ‘save_model’: Implementazione di ‘load_model’ di BaseModel, salva il modello e il suo dizionario di stato in file;
  5. ‘train_model’: Implementazione di ‘load_model’ di BaseModel, carica o definisce il modello NN, e se il file del modello non esiste, esegue il training del modello utilizzando i dati preprocessati. Successivamente, salva il modello addestrato.

- ModelContext:
  1. ‘set_model_info’ : Questo metodo consente di impostare le informazioni sul modello (model_info) dell’oggetto ModelContext. Accetta un argomento model_info, che viene quindi assegnato all’attributo model_info;
  2. ‘set_model_operator’ : Simile a set_model_info, questo metodo consente di impostare l’operatore di modello (model_operator) dell’oggetto ModelContext. Accetta un argomento model_operator, che viene quindi assegnato all’attributo model_operator;
  3. ‘process_data’ : Questo metodo permette di elaborare i dati attraverso le informazioni sul modello. Accetta un argomento data rappresentante i dati da elaborare. Utilizza l’attributo model_info per chiamare il metodo load_data, per caricare i dati nel modello;
  4. ‘load_model’ : Metodo responsabile del caricamento di un modello;
  5. ‘save_model’ : Metodo responsabile del salvataggio di un modello;
  6. ‘train_model’ : Metodo responsabile dell’addestramento di un modello;
  7. ‘topN_1UserNItem’ : Questo metodo restituisce i migliori N elementi per un dato utente in base alle previsioni del modello;
  8. ‘topN_1ItemNUser’ : Simile a topN_1UserNItem, ma restituisce i migliori N utenti per un dato elemento in base alle previsioni del modello.

3.2.3.2.4) Operator

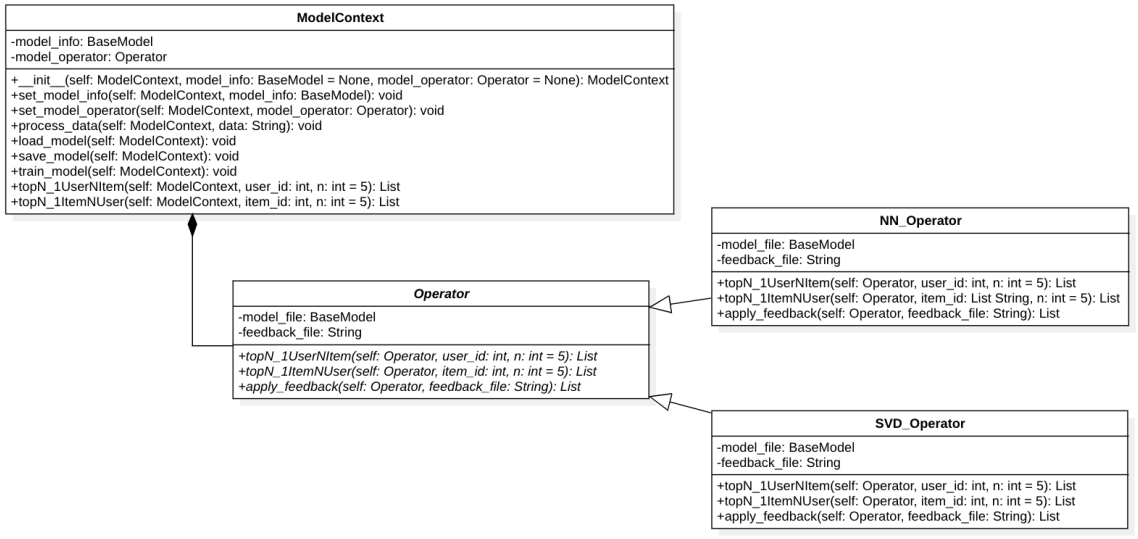

Figura 18: Operator

Descrizione:
La classe BaseOperator è una classe astratta che definisce un'interfaccia comune per gli operatori specifici dei modelli di raccomandazione. Le classi NN_Operator e SVD_Operator sono entrambe sottoclassi di BaseOperator; forniscono implementazioni specifiche per due diversi operatori di modelli, uno basato su reti neurali (NN) e l'altro basato su svd (Singular Value Decomposition). Infine, ModelContext agisce come un mediatore che connette queste classi di modelli con il mondo esterno. Astrae i dettagli specifici del trattamento del modello e fornisce un'interfaccia unificata per il caricamento, il salvataggio e l'addestramento dei modelli e delega le operazioni effettive alla classe di modello appropriata (SVD_Model o NN_Model) in base all'operatore di modello fornito.

Metodi:

- BaseOperator:
  1. 'ratings_float2int' : metodo astratto che si occupa di convertire i rating previsti da valori float a valori interi;
  2. 'apply_feedback' : metodo astratto che si occupa di applicare il feedback ricevuto (su utenti o elementi) ai rating previsti dal modello;
  3. 'topN_1UserNItem' : metodo astratto che restituisce i migliori N elementi per un dato utente in base alle previsioni del modello;
  4. 'topN_1ItemNUser' : metodo astratto che restituisce i migliori N utenti per un dato elemento in base alle previsioni del modello.
- NN_Operator:
  1. 'ratings_float2int' : Implementazione di 'ratings_float2int' di BaseOperator, converte le previsioni dei rating da valori float a valori interi, utilizzando una trasformazione lineare;
  2. 'apply_feedback' : Implementazione di 'apply_feedback' di BaseOperator, applica il feedback ricevuto (su utenti o elementi) ai rating previsti dal modello;
  3. 'topN_1UserNItem' : Implementazione di 'topN_1UserNItem' di BaseOperator, restituisce i migliori N elementi per un dato utente in base alle previsioni del modello;
  4. 'topN_1ItemNUser' : Implementazione di 'topN_1ItemNUser' di BaseOperator, restituisce i migliori N utenti per un dato elemento in base alle previsioni del modello.

- SVD_Operator:
  1. ‘ratings_float2int’ : Implementazione di ‘ratings_float2int’ di BaseOperator, converte le previsioni dei rating da valori float a valori interi, utilizzando una trasformazione lineare;
  2. ‘apply_feedback’ : Implementazione di ‘apply_feedback’ di BaseOperator, applica il feedback ricevuto (su utenti o elementi) ai rating previsti dal modello;
  3. ‘topN_1UserNItem’ : Implementazione di ‘topN_1UserNItem’ di BaseOperator, restituisce i migliori N elementi per un dato utente in base alle previsioni del modello;
  4. ‘topN_1ItemNUser’ : Implementazione di ‘topN_1ItemNUser’ di BaseOperator, restituisce i migliori N utenti per un dato elemento in base alle previsioni del modello.

  ▸ ModelContext:
  1. ‘set_model_info’ : Questo metodo consente di impostare le informazioni sul modello (model_info) dell’oggetto ModelContext. Accetta un argomento model_info, che viene quindi assegnato all’attributo model_info;
  2. ‘set_model_operator’ : Simile a set_model_info, questo metodo consente di impostare l’operatore di modello (model_operator) dell’oggetto ModelContext. Accetta un argomento model_operator, che viene quindi assegnato all’attributo model_operator;
  3. ‘process_data’ : Questo metodo permette di elaborare i dati attraverso le informazioni sul modello. Accetta un argomento data rappresentante i dati da elaborare. Utilizza l’attributo model_info per chiamare il metodo load_data, per caricare i dati nel modello;
  4. ‘load_model’ : Metodo responsabile del caricamento di un modello;
  5. ‘save_model’ : Metodo responsabile del salvataggio di un modello;
  6. ‘train_model’ : Metodo responsabile dell’addestramento di un modello;
  7. ‘topN_1UserNItem’ : Questo metodo restituisce i migliori N elementi per un dato utente in base alle previsioni del modello;
  8. ‘topN_1ItemNUser’ : Simile a topN_1UserNItem, ma restituisce i migliori N utenti per un dato elemento in base alle previsioni del modello.

3.2.3.2.5) Librerie esterne

Descrizione:

Metodi:

3.2.4) Documentazione API

La sezione seguente fornisce una panoramica delle API create dal team Farmacode per comunicare con l’applicazione, delineando brevemente le operazioni disponibili e i dati accessibili. Una descrizione dettagliata della loro struttura è disponibile nel documento “Manuale Sviluppatore v.1.0.0”; questo permette agli sviluppatori interessati di comprendere appieno il software e di implementare nuove funzionalità in modo automatizzato, senza dover interagire manualmente con l’interfaccia utente. Pertanto questa sezione fornisce un’illustrazione generale e indicativa delle API disponibili.

3.2.4.1) Chiamate GET

1. /
   - **Descrizione:**
     Verifica se il server è in esecuzione.
   - **Parametri:**
     Nessuno.

- **Ritorno:**
  Stringa “Server is running!”.
- **Codici di stato HTTP:**

Esito	**HTTP**	**Descrizione**
Positivo	200: OK	La richiesta è stata elaborata correttamente.

Tabella 6: Esito della richiesta di verifica dello stato del server.

2. /login/:use :
   - **Descrizione:**
     Ritorna i dati di login di uno specifico utente.
   - **Parametri:**
     - :use (parametro nella URL): Il nome utente dell’utente che sta tentando di effettuare l’accesso.
   - **Ritorno:**
     Informazioni di login per l’utente.
   - **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La richiesta è stata elaborata correttamente.
Negativo	404: Not Found	L’utente non è stato trovato.
Errore	500: Internal Server Error	Errore durante il recupero delle informazioni di login.

Tabella 7: Esito della richiesta di login dell’utente.

3. /users :
   - **Descrizione:**
     Ritorna la lista completa degli utenti.
   - **Parametri:**
     Nessuno.
   - **Ritorno:**
     La risposta conterrà la lista completa degli utenti registrati nel sistema.
   - **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La lista degli utenti è stata recuperata con successo.

Errore	500: Internal Server Error	Si è verificato un errore durante il recupero della lista degli utenti.

Tabella 8: Esito della richiesta di recupero della lista degli utenti.

4. /users/:id :

- **Descrizione:**
  Recupera le informazioni dell'utente corrispondente all'ID specificato.
- **Parametri:**
  ▸ :id (parametro nella URL): ID dell'utente.
- **Ritorno:**
  Le informazioni dell'utente corrispondente all'ID specificato.
- **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	Le informazioni dell'utente sono state recuperate con successo.
Errore	404: Not Found	L'utente specificato non è stato trovato.
Errore	500: Internal Server Error	Si è verificato un errore durante il recupero delle informazioni dell'utente.

Tabella 9: Esito della richiesta di recupero delle informazioni dell'utente.

5. /items :

- **Descrizione:**
  Ritorna la lista completa dei prodotti con codice articolo e descrizione.
- **Parametri:**
  Nessuno.
- **Ritorno:**
  La lista degli articoli.
- **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La lista degli articoli è stata recuperata con successo.
Errore	500: Internal Server Error	Si è verificato un errore durante il recupero della lista degli articoli.

Tabella 10: Esito della richiesta di recupero della lista degli articoli.

6. /items/:id :

- **Descrizione:**
  Ritorna le informazioni di uno specifico prodotto.
- **Parametri:**
  ▸ :id (parametro nella URL): ID dell'articolo.
- **Ritorno:**
  Le informazioni dell'articolo corrispondente all'ID specificato.
- **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	Le informazioni dell'articolo sono state recuperate con successo.
Errore	404: Not Found	L'articolo specificato non è stato trovato.
Errore	500: Internal Server Error	Si è verificato un errore durante il recupero delle informazioni dell'articolo.

Tabella 11: Esito della richiesta di recupero delle informazioni dell'articolo.

7. /prodotti :
   - **Descrizione:**
     Ritorna la lista completa dei prodotti con tutte le informazioni, ordinata in base al codice.
   - **Parametri:**
     Nessuno.
   - **Ritorno:**
     La lista dei prodotti.
   - **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La lista dei prodotti è stata recuperata con successo.
Errore	500: Internal Server Error	Si è verificato un errore durante il recupero della lista dei prodotti.

Tabella 12: Esito della richiesta di recupero della lista dei prodotti.

8. /prodotti/lineecommerciali :
   - **Descrizione:**
     Recupera la lista completa delle linee commerciali dei prodotti.
   - **Parametri:**
     Nessuno.

- **Ritorno:**
  Se la richiesta ha successo, la risposta sarà un array JSON contenente tutte le linee commerciali dei prodotti presenti nel sistema.
- **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La lista delle linee commerciali dei prodotti è stata recuperata con successo.
Errore	500: Internal Server Error	Si è verificato un errore durante il recupero delle linee commerciali dei prodotti.

Tabella 13: Esito della richiesta di recupero delle linee commerciali dei prodotti.

9. /prodotti/settoricommerciali :
   - **Descrizione:**
     Recupera la lista completa dei settori commerciali dei prodotti.
   - **Parametri:**
     Nessuno.
   - **Ritorno:**
     Se la richiesta ha successo, la risposta sarà un array JSON contenente tutti i settori commerciali dei prodotti presenti nel sistema.
   - **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La lista dei settori commerciali dei prodotti è stata recuperata con successo.
Errore	500: Internal Server Error	Si è verificato un errore durante il recupero dei settori commerciali dei prodotti.

Tabella 14: Esito della richiesta di recupero dei settori commerciali dei prodotti.

10. /prodotti/famigliecommerciali :
    - **Descrizione:**
      Recupera la lista completa delle famiglie commerciali dei prodotti.
    - **Parametri:**
      Nessuno.
    - **Ritorno:**
      Se la richiesta ha successo, la risposta sarà un array JSON contenente tutte le famiglie commerciali dei prodotti presenti nel sistema.
    - **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La lista delle famiglie commerciali dei prodotti è stata recuperata con successo.
Errore	500: Internal Server Error	Si è verificato un errore durante il recupero delle famiglie commerciali dei prodotti.

Tabella 15: Esito della richiesta di recupero delle famiglie commerciali dei prodotti.

11. /prodotti/sottofamigliecommerciali :

- **Descrizione:**
  Recupera la lista completa delle sottofamiglie commerciali dei prodotti.
- **Parametri:**
  Nessuno.
- **Ritorno:**
  Se la richiesta ha successo, la risposta sarà un array JSON contenente tutte le sottofamiglie commerciali dei prodotti presenti nel sistema.
- **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La lista delle sottofamiglie commerciali dei prodotti è stata recuperata con successo.
Errore	500: Internal Server Error	Si è verificato un errore durante il recupero delle sottofamiglie commerciali dei prodotti.

Tabella 16: Esito della richiesta di recupero delle sottofamiglie commerciali dei prodotti.

12. /clienti :

- **Descrizione:**
  Ritorna la lista completa dei clienti ordinata in base al codice.
- **Parametri:**
  Nessuno.
- **Ritorno:**
  Se la richiesta ha successo, la risposta sarà un array JSON contenente tutti i clienti presenti nel sistema, insieme ai dettagli del relativo fornitore.
- **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La lista dei clienti è stata recuperata con successo.

Errore	500: Internal Server Error	Si è verificato un errore durante il recupero dei dati dei clienti.

Tabella 17: Esito della richiesta di recupero dei clienti.

13. /clienti/province :

- **Descrizione:**
  Recupera la lista completa delle province dei clienti.
- **Parametri:**
  Nessuno.
- **Ritorno:**
  Se la richiesta ha successo, la risposta sarà un array JSON contenente tutte le province dei clienti presenti nel sistema.
- **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La lista delle province dei clienti è stata recuperata con successo.
Errore	500: Internal Server Error	Si è verificato un errore durante il recupero delle province dei clienti.

Tabella 18: Esito della richiesta di recupero delle province dei clienti.

14. /cronologia :

- **Descrizione:**
  Recupera la cronologia delle attività degli utenti.
- **Parametri:**
  Nessuno.
- **Ritorno:**
  Se la richiesta ha successo, la risposta sarà un array JSON contenente la cronologia delle attività degli utenti, ordinata in base alla data cronologica in ordine crescente.
- **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La cronologia delle attività degli utenti è stata recuperata con successo.
Errore	500: Internal Server Error	Si è verificato un errore durante il recupero della cronologia delle attività degli utenti.

Tabella 19: Esito della richiesta di recupero della cronologia delle attività degli utenti.

15. /feedback :

- **Descrizione:**
  Recupera la lista dei feedback degli ordini dei clienti.
- **Parametri:**
  Nessuno.
- **Ritorno:**
  Se la richiesta ha successo, la risposta sarà un array JSON contenente i feedback degli ordini dei clienti, ordinati in base alla data del feedback in ordine decrescente.
- **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La lista dei feedback degli ordini dei clienti è stata recuperata con successo.
Errore	500: Internal Server Error	Si è verificato un errore durante il recupero dei feedback degli ordini dei clienti.

Tabella 20: Esito della richiesta di recupero dei feedback degli ordini dei clienti.

3.2.4.2) Chiamate PUT

1. /cronologia/new :
   - **Descrizione:**
     Aggiunge una nuova voce alla cronologia delle attività degli utenti.
   - **Parametri:**
     - user (string): L'utente che ha eseguito l'attività.
     - algo (string): L'algoritmo utilizzato per l'attività.
     - topic (string): Il topic dell'attività.
     - cod_ric (string): Il codice relativo all'attività.
     - top_sel (string): Il top selezionato per l'attività.
   - **Ritorno:**
     Se la richiesta ha successo, la risposta sarà un oggetto JSON con un messaggio di successo.
   - **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La nuova voce è stata inserita con successo nella cronologia delle attività degli utenti.
Errore	400: Bad Request	Uno o più parametri richiesti sono mancanti o non validi.
Errore	500: Internal Server Error	Si è verificato un errore durante l'inserimento della nuova voce nella cronologia delle attività degli utenti.

Tabella 21: Esito della richiesta di inserimento di una nuova voce nella cronologia delle attività degli utenti.

2. /feedback/newUser :

   - **Descrizione:**
     Aggiunge un nuovo feedback per un ordine di un cliente.
   - **Parametri:**
     - user (string): L'utente che ha fornito il feedback.
     - id (string): L'identificatore dell'articolo associato al feedback.
     - idRic (string): L'identificatore dell'ordine cliente associato al feedback.
     - algoType (string): Il tipo di algoritmo utilizzato.
   - **Ritorno:**
     Se la richiesta ha successo, la risposta sarà un oggetto JSON con un messaggio di successo.
   - **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	Il nuovo feedback è stato inserito con successo per l'ordine del cliente.
Errore	400: Bad Request	Uno o più parametri richiesti sono mancanti o non validi.
Errore	500: Internal Server Error	Si è verificato un errore durante l'inserimento del nuovo feedback per l'ordine del cliente.

Tabella 22: Esito della richiesta di inserimento di un nuovo feedback per l'ordine del cliente.

3. /feedback/newItem :

   - **Descrizione:**
     Aggiunge un nuovo feedback per un articolo.
   - **Parametri:**
     - user (string): L'utente che ha fornito il feedback.
     - id (string): L'identificatore dell'ordine cliente associato al feedback.
     - idRic (string): L'identificatore dell'articolo associato al feedback.
     - algoType (string): Il tipo di algoritmo utilizzato.
   - **Ritorno:**
     Se la richiesta ha successo, la risposta sarà un oggetto JSON con un messaggio di successo.
   - **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	Il nuovo feedback è stato inserito con successo per l'articolo.

Errore	400: Bad Request	Uno o più parametri richiesti sono mancanti o non validi.
Errore	500: Internal Server Error	Si è verificato un errore durante l'inserimento del nuovo feedback per l'articolo.

Tabella 23: Esito della richiesta di inserimento di un nuovo feedback per l'articolo.

4. /feedback/delFeed :

- **Descrizione:**
  Elimina un feedback specifico.
- **Parametri:**
  ▸ id_feed (string): L'identificatore del feedback da eliminare.
- **Ritorno:**
  Se la richiesta ha successo, la risposta sarà un oggetto JSON con un messaggio di successo.
- **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	Il feedback è stato eliminato con successo.
Errore	400: Bad Request	Il parametro id_feed è mancante o non valido.
Errore	500: Internal Server Error	Si è verificato un errore durante l'eliminazione del feedback.

Tabella 24: Esito della richiesta di eliminazione di un feedback.

5. /userana/:use :

- **Descrizione:**
  Recupera i dettagli di un utente specifico usando il suo identificatore unico.
- **Parametri:**
  ▸ use (string): L'identificatore dell'utente da cercare.
- **Ritorno:** Se l'utente è trovato, la risposta sarà un array JSON contenente un oggetto con i dettagli dell'utente.
- **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	I dettagli dell'utente sono restituiti correttamente.
Errore	404: Not Found	L'utente con l'ID specificato non è stato trovato.

Errore	500: Internal Server Error	Si è verificato un errore durante il recupero dei dati.

Tabella 25: Esito della richiesta di dettagli di un utente.

6. /userana/:use/email :

- **Descrizione:**
  Aggiorna l'indirizzo email di un utente specifico.
- **Parametri:**
  - use (string): L'identificatore univoco dell'utente.
  - newEmail (string, nel corpo della richiesta): Il nuovo indirizzo email da assegnare all'utente.
- **Ritorno:** Restituisce un messaggio di successo se l'email è stata aggiornata correttamente.
- **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	L'indirizzo email dell'utente è stato aggiornato con successo.
Errore	400: Bad Request	Errore nella richiesta, manca l'indirizzo email nel corpo della richiesta.
Errore	500: Internal Server Error	Si è verificato un errore durante l'aggiornamento dell'indirizzo email.

Tabella 26: Esito dell'aggiornamento dell'indirizzo email di un utente.

7. /userana/:use/password :

- **Descrizione:**
  Aggiorna la password di un utente specifico.
- **Parametri:**
  - use (string): L'identificatore univoco dell'utente.
  - newPassword (string, nel corpo della richiesta): La nuova password da assegnare all'utente.
- **Ritorno:** Restituisce un messaggio di successo se la password è stata aggiornata correttamente.
- **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La password dell'utente è stata aggiornata con successo.

Errore	400: Bad Request	Errore nella richiesta, manca la nuova password nel corpo della richiesta.
Errore	500: Internal Server Error	Si è verificato un errore durante l'aggiornamento della password.

Tabella 27: Esito dell'aggiornamento della password di un utente.

3.2.4.3) Chiamate Route

1. /train/:algo :

   - **Descrizione:**
     Avvia l'addestramento del modello machine learning basato sull'algoritmo specificato.
   - **Parametri:**
     algo (string): L'identificatore dell'algoritmo di machine learning da addestrare. I valori accettati sono "SVD" o "NN".
   - **Ritorno:**
     Restituisce un messaggio di successo se l'addestramento è completato correttamente.
   - **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	L'addestramento dell'algoritmo è stato completato con successo.
Errore	400: Bad Request	Errore nella richiesta, algoritmo non specificato o non valido.
Errore	500: Internal Server Error	Si è verificato un errore durante l'addestramento dell'algoritmo.

Tabella 28: Esito dell'addestramento di un algoritmo di machine learning.

1. /search/:algo/:oggetto/:id/:n :

   - **Descrizione:**
     Esegue una ricerca utilizzando un algoritmo specificato su un oggetto specifico per un dato ID e restituisce i migliori N risultati.
   - **Parametri:**
     - algo (string): L'algoritmo utilizzato per la ricerca. I valori accettati sono "SVD" o "NN".
     - oggetto (string): L'oggetto su cui eseguire la ricerca. Può essere "user" o "item".
     - id (string): L'identificatore univoco dell'oggetto su cui eseguire la ricerca.
     - n (string): Il numero di risultati da restituire.
   - **Ritorno:** Restituisce una lista di risultati, o un messaggio di errore se si verificano problemi durante l'esecuzione della ricerca.
   - **Codici di stato HTTP:**

Esito	HTTP	Descrizione
Positivo	200: OK	La ricerca è stata completata con successo.
Errore	400: Bad Request	Errore nella richiesta, ad esempio parametri mancanti o non validi.
Errore	500: Internal Server Error	Si è verificato un errore durante l'esecuzione della ricerca.

Tabella 29: Esito della ricerca utilizzando un algoritmo specifico.

4) Stato requisiti funzionali

4.1) Tabella requisiti funzionali

Codice	Descrizione	Fonti
ROF 1	L'utente per potere accedere all'applicazione deve autenticarsi all'interno del sistema.	Soddisfatto
ROF 2	L'utente deve fornire la propria email personale, nel campo email, per procedere con l'autenticazione nella pagina di Login.	Soddisfatto
ROF 3	L'utente deve fornire la propria password, nel campo password, per procedere con l'autenticazione nella pagina di Login.	Soddisfatto
RDF 4	Nel caso il sito sia in manutenzione è necessario che l'utente riceva un messaggio che esplicità l'impossibilità di usare l'applicazione.	
RDF 5	Nel caso l'autenticazione fallisse, è necessario che l'utente riceva un messaggio con dettagli che ne indicano il motivo.	
ROF 6	L'utente, una volta autenticato, deve poter accedere alla funzione "Profilo Utente" nella pagina principale del sito.	Soddisfatto
ROF 7	L'utente, una volta entrato nella sezione "Profilo Utente", deve poter visualizzare i dati utente o modificarli.	Soddisfatto
ROF 8	L'utente che ha scelto di visulizzare i "dati utente" deve visulizzare, l'anagrafica, l'email, l'username e la password.	
ROF 9	L'utente che ha scelto di modificare i dati utente, deve poter modificare l'email e la password.	
ROF 10	L'utente, una volta autenticato, deve poter effettuare il Logout tramite il pulsante presente nella pagina principale del sito.	Soddisfatto
ROF 11	L'utente, una volta autenticato, deve poter accedere alla funzione "Ricerca" nella pagina principale del sito.	Soddisfatto
ROF 12	L'utente una volta entrato nella sezione "Ricerca", deve poter effettuare una ricerca e visualizzarne i risultati.	Soddisfatto
ROF 13	L'utente che ha scelto di effettuare una ricerca, deve compilare tutti i campi per effettuarla e poter visualizzare i risultati.	Soddisfatto
ROF 14	L'utente che compila la scelta del topic della ricerca, può scegliere tra prodtto per cliente o cliente per prodotto, per poi compilare i successivi campi.	Soddisfatto

RDF 15	L'utente che compila la scelta del topic della ricerca, può scegliere la ricerca per cronologia, per poi compilare i successivi campi.	
RDF 16	L'utente che cimpila la scelta degli "N risultati", può scegliere tra i 5 migliori risultati (Top 5) o tra i migliori 10 (Top 10).	Soddisfatto
ROF 17	L'utente che ha effettuato una ricerca e ne visualizza i risultati, deve poter visualizzare, l'ID, il nome e lo score assegnato alla raccomandazione.	Soddisfatto
RDF 18	Nel caso la ricerca non andasse a buon fine, l'utente deve visualizzare un messaggio di errore che indica che la ricerca non è terminata correttamente. Il messaggio di errore deve essere mostrato in caso di errore anche per le ricerche di RDF 23, RDF 29, RDF 37, RDF 41.	Soddisfatto
ROF 19	L'utente che ha visualizzato i risultati della ricerca, deve poter inserire un feedback delle raccomandazioni mostrate.	
ROF 20	L'utente se decide di assegnare un feedback ad una raccomandazione, dovrà compilare i campi di "valutazione" e "commento".	
RDF 21	L'utente, una volta autenticato, deve poter accedere alla funzione "Catalogo Prodotti" nella pagina principale del sito.	Soddisfatto
RDF 22	L'utente una volta entrato nella sezione "Catalogo Prodotti", deve poter effettuare una ricerca e visualizzarne i risultati.	
RDF 23	L'utente che ha scelto di effetuare una ricerca dever compilare i campi Codice prodotto, Linea commerciale, Settore commerciale e Marca prodotto.	
RDF 24	L'utente che ha effettuato una ricerca e ne visualizza i risultati, deve poter visualizzare, l'immagine, l'ID e il nome del prodotto.	
RDF 25	L'utente che ha visualizzato i risultati della ricerca, può visualizzare i dettagli di un prodotto, cliccando sul bottone a finaco delle riga della ricerca.	
RDF 26	L'utente se decide di visualizzare i dettagli di un prodotto, deve poter vedere l'immagine, l'ID, il nome, la liena commerciale, il settore commerciale, la marca e la provenienza del prodotto.	
RDF 27	L'utente, una volta autenticato, deve poter accedere alla funzione "Lista clienti" nella pagina principale del sito.	Soddisfatto

RDF 28	L'utente una volta entrato nella sezione "Lista clienti", deve poter effettuare una ricerca e visualizzarne i risultati.	Soddisfatto
RDF 29	L'utente che ha scelto di effettuare una ricerca, deve compilare i campi Nome, Cognome e Provincia.	
RDF 30	L'utente che effettutato una ricerca e ne visualizza i risultati, deve poter visualizzare l'ID, il nome, il cognome e la provincia del cliente	
RDF 31	L'utente, una volta autenticato, deve poter accedere alla funzione "Statistiche mensili" nella pagina principale del sito.	
RDF 32	L'utente una volta entrato nella sezione "Statistiche mensili", deve poter visualizzare il grafico e la lista di raccomandazioni utili.	
RDF 33	L'utente che ha scelto di visualizzare il grafico, visualizza sull'asse delle X i giorni e sull'asse delle Y le raccomandazioni utili.	
RDF 34	L'utente che ha scelto di visualizzare le raccomandazioni utili, deve poter visualizzare l'ID del prodotto, l'ID del cliente e lo score assegnato alla raccomandazione.	
RDF 35	L'utente, una volta autenticato, deve poter accedere alla funzione "Cronologia ricerche" nella pagina principale del sito.	
RDF 36	L'utente una volta entrato nella sezione "Cronologia ricerche", deve poter effettuare una ricerca e visualizzarne i risultati.	
RDF 37	L'utente che ha scelto di effettuare una ricerca, deve compilare i campi "Data" e "Username".	
RDF 38	L'utente che effettutato una ricerca e ne visualizza i risultati, deve poter visualizzare la data, l'username e i criteri di ricerca riguardanti la cronologia della ricerca.	
RDF 39	L'utente, una volta autenticato, deve poter accedere alla funzione "Cronologia feedback" nella pagina principale del sito.	
RDF 40	L'utente una volta entrato nella sezione "Cronologia feedback", deve poter effettuare una ricerca e visualizzarne i risultati.	
RDF 41	L'utente che ha scelto di effettuare una ricerca, deve compilare i campi "Data" e "Username".	

RDF 42	L'utente che effettutato una ricerca e ne visualizza i risultati, deve poter visualizzare la data, l'username e il contenuto del feedback riguardanti la cronologia del feedback.	
RDF 43	L'utente, una volta autenticato, deve poter accedere alla funzione "Carica dataset" e caricare un dataset esterno all'interno dell'applicazione.	
RDF 44	L'utente, se ha caricato un dataset esterno, dever poter avviare il training del dataset in maniera da poterlo usare per le raccomandazioni.	

Tabella 28: Requisiti funzionali

4.2) Grafico requisiti funzionali

5) Elenco delle immagini

6) Elenco delle tabelle