Specifica tecnica

v0.9.0

Farmacode

farmacode.swe.unipd@gmail.com

Registro delle modifiche

Versione	Data	Scrittori	Revisori	Descrizione
0.9.0	2024-04-02	Passarella Alessandro	Baggio Matteo	Stesura documentazione API
0.8.0	2024-04-01	Carraro Alessandro	Favaron Riccardo	Seconda stesura sezione Front-end/ Application Logic
0.7.2	2024-03-24	Rosson Lorenzo	Pandolfo Mattia	Correzioni sezione Persistence Logic e DB
0.7.1	2024-03-20	Rosson Lorenzo	Pandolfo Mattia	Aggiunta diagrammi classi
0.7.0	2024-03-20	Pandolfo Mattia	Favaron Riccardo	Stesura sottosezione Componenti
0.5.0	2024-03-16	Rosson Lorenzo	Favaron Riccardo	Seconda stesura sezione Architettura, e correzioni alla struttura del documento
0.4.1	2024-03-12	Favaron Riccardo	Baggio Matteo	Migliorie sezione Tecnologie
0.4.0	2024-03-06	Pandolfo Mattia	Baggio Matteo	Prima stesura architettura Back-end
0.3.0	2024-03-06	Bomben Filippo	Baggio Matteo	Architettura Front-end
0.2.0	2024-03-03	Bomben Filippo	Favaron Riccardo	Stesura sezione 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 di specifica tecnica per un progetto software ha lo scopo di delineare in dettaglio tutti gli aspetti tecnici del sistema che si intende sviluppare. È essenzialmente una guida tecnica che fornisce una visione completa dell'architettura, delle tecnologie utilizzate, delle componenti e delle interazioni del sistema.
Innanzitutto, il documento descrive le tecnologie che verranno impiegate nel progetto. Questo può includere informazioni su linguaggi di programmazione, framework, database, librerie esterne e altri strumenti tecnologici rilevanti. Questa sezione fornisce una panoramica delle risorse tecniche necessarie per lo sviluppo del sistema.

Successivamente, il documento dettaglia l'architettura del sistema, suddividendola in varie componenti logiche. La suddivisione che abbiamo prestabilito include:

- persistence logic, che gestisce l'interazione con il database e la persistenza dei dati;
- la business logic, che implementa le regole di business e la logica di elaborazione;
- l'application logic, che si occupa dell'interfaccia utente e della gestione delle richieste dell'utente.

Un altro aspetto importante del documento è la descrizione delle interfacce di programmazione delle applicazioni (API) utilizzate nel sistema. Le API definiscono i metodi e le operazioni che possono essere eseguite sul sistema e forniscono un modo standardizzato per la comunicazione e l'integrazione con altre applicazioni. Infine, il documento cataloga i requisiti funzionali coperti del sistema. Questi sono i compiti o le funzionalità che il sistema è in grado di svolgere per soddisfare le esigenze degli utenti.
Complessivamente, il documento di specifica tecnica è uno strumento essenziale per il team di sviluppo, fornendo una guida chiara e condivisa su come il sistema dovrebbe essere progettato e implementato. Aiuta a garantire coerenza, comprensione e conformità agli obiettivi del progetto, fornendo una base solida per il successo dello sviluppo del software.

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*. Versione di riferimento v2.0.0.

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 (data di ultimo accesso: 2023/12/12);
- Regolamento progetto ditattico
  https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/PD2.pdf (data di ultimo accesso: 2023/12/12).

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 (data di ultimo accesso: 2024/03/10);
- Progettazione: I pattern architetturali (slide del corso di Ingegneria del Software)
  https://www.math.unipd.it/~rcardin/swea/2022/Software%20Architecture%20Patterns.pdf (2024/03/22);
- 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 (data di ultimo accesso: 2024/03/02);
- 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 (data di ultimo accesso: 2024/03/02);
- 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 (data di ultimo accesso: 2024/03/10);
- Documentazione Docker
  https://docs.docker.com/ (data di ultimo accesso: 2024/02/27).

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 versione della tecnologia usata;
- La descrizione del ruolo che la tecnologia ha avuto all'interno del progetto.

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	Versione	Descrizione
HTML	5	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.
CSS	3	Linguaggio per la formattazione dei documenti HTML, il suo scopo è di gestire lo stile e il design del sito.
JavaScript	TD	Linguaggio di programmazione per la gestione degli eventi dell'utente e per la comunicazione con l'API.
Python	3.11.5	Linguaggio di programmazione usato per la creazione del sistema di raccomandazione.
SQL	TD	Linguaggio di interrogazione per la creazione e gestione del database.

Tabella 1: Linguaggi

2.1.2) Librerie e framework

Tecnologia	Versione	Descrizione
Pandas	2.1.1	Libreria per Python utilizzata per la manipolazione e l'analisi dei dati
Surprise	1.1.3	Libreria per Python utilizzata per semplificare lo sviluppo di sistemi di raccomandazione e valutare le prestazioni di algoritmi di filtraggio collaborativo

React.js	18.2.0	Libreria JavaScript utilizzata per semplificare lo sviluppo front-end, consentendo una gestione modulare delle componenti grafiche.
PrimeReact	10.5.1	Suite per l'User Interface per React.js che utilizza componenti già definiti e ben strutturati.
Express	4.18.2	Libreria di JavaScript utilizzata per lo sviluppo back-end del sito, in particolare per la comunicazione tra la il front-end e il database.
Flask	3.0.3	Framework per lo sviluppo di applicazioni web in Python che fornisce strumenti per la gestione delle richieste HTTP. Utilizzata per la comunicazione tra il front-end e il sistema di raccomandazione.
NumPy	1.26.0	Libreria per Python utilizzata per la manipolazione di array e matrici multidimensionali.
PyTorch	2.2.2	Framework per l'apprendimento automatico basato su Python che offre tensori potenti, grafi computazionali dinamici e autograd.
Tailwind CSS	3.4.1	Framework per css utilizzato per lo sviluppo di interfacce utente.
Axios	1.6.8	Libreria JavaScript utilizzata per effettuare richieste HTTP sia lato client che lato server.

Tabella 2: Librerie e framework

2.1.3) Strumenti e servizi

Tecnologia	Versione	Descrizione
MySQL	Latest	RDBMS per la creazione e gestione dei database in SQL.
Node.js	18.16.1	Ambiente di runtime open-source per l'esecuzione di codice JavaScript lato server tramite appositi script.
NPM	9.5.1	Gestore di pacchetti (Node Package Manager) per JavaScript all'interno di Node.js.
VS Code	Latest	IDE di programmazione gratuito ricco estensioni esterne.
Docker	Latest	Creatore di ambienti di sviluppo tramite container per la gestione delle dipendenze.

Git	Latest	Sistema di controllo e versionamento utilizzato per la gestione del codice.
Anaconda	Latest	Gestore e distributore per Python dei pacchetti per la gestione delle versioni.

Tabella 3: Strumenti e servizi

2.2) Tecnologie per l'analisi del codice

2.2.1) Analisi statica

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

Tabella 4: Analisi statica

2.2.2) Analisi dinamica

Tecnologia	Descrizione	Versione
Pytest	8.0.x	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.
Jest	27.5.1	Framework di test basato su JavaScript con funzionalità di creazione di mock e il testing del codice in modo asincrono.
GitHub Action	/	Servizio di CI/CD per automatizzare il processo di build, test e deploy del progetto software.

Tabella 5: Analisi dinamica

3) Architettura

Premessa: Come menzionato nei documenti "Norme di Progetto" e "Piano di Qualifica", il gruppo ha deciso di utilizzare in ambito progettuale un approccio top-down, del quale si riportano brevemente i punti di forza individuati:

- Visione d'insieme: questo approccio consente di avere una visione completa del progetto fin dall'inizio, permettendo di identificare i requisiti principali e di pianificare di conseguenza;
- Struttura modulare: la progettazione top-down favorisce la suddivisione del progetto in moduli o componenti più piccoli, semplificando così lo sviluppo e la gestione del software;
- Facilità di gestione del cambiamento: poiché i dettagli sono definiti solo dopo che l'architettura generale è stata stabilita, è più facile apportare modifiche durante le fasi iniziali del progetto senza dover ridisegnare completamente il sistema;
- Riduzione della complessità: concentrandosi sui concetti fondamentali e sulla logica generale, la progettazione top-down aiuta a ridurre la complessità del progetto, rendendo più facile la comprensione e la manutenzione del software;
- Collaborazione efficace: la divisione del progetto in moduli facilita la collaborazione tra i membri del team, consentendo a ciascuno di lavorare su parti specifiche del progetto in modo indipendente;
- Testabilità: la suddivisione del sistema in moduli facilita l'individuazione e l'isolamento dei bug, semplificando il processo di testing e debug;
- Scalabilità: una volta definita l'architettura generale, è più semplice scalare il sistema aggiungendo nuovi moduli o migliorando quelli esistenti senza dover riprogettare l'intero sistema.

3.1) Docker e Containerizzazione

Nonostante non rientrasse nei requisiti obbligatori espressi dal propronente, il gruppo ha deciso di adottare i vari servizi che Docker fornisce per la gestione dell'infrastruttura del prodotto. I vantaggi che quest'ultimo offre hanno indotto facilmente alla scelta:

- Isolamento: offre un'isolamento leggero e portatile delle applicazioni tramite i container, consentendo loro di essere eseguiti in ambienti virtualizzati senza il peso delle macchine virtuali tradizionali. Questo significa che le applicazioni possono essere eseguite in modo consistente su qualsiasi ambiente, sia esso locale, in cloud o in ambienti di produzione;
- Velocità di distribuzione: i container possono essere creati e distribuiti in modo rapido e efficiente. Poiché contengono tutto ciò di cui un'applicazione ha bisogno per essere eseguita, è possibile distribuire facilmente le applicazioni senza dover preoccuparsi delle dipendenze del sistema ospite;
- Scalabilità: Docker consente di scalare facilmente le applicazioni orizzontalmente, aggiungendo istanze dei container in risposta a picchi di carico. Questo facilita la gestione della disponibilità e delle prestazioni delle applicazioni in ambienti di produzione ad alta intensità di traffico;
- Ambienti consistenti: utilizzando Docker, è possibile creare ambienti di sviluppo, test e produzione consistenti. Questo favorisce la collaborazione tra team di sviluppo e semplifica la distribuzione delle applicazioni attraverso i vari ambienti;

- Gestione semplificata: Docker fornisce strumenti potenti per la gestione dei contenitori, inclusi Docker Compose per la definizione e l'esecuzione di applicazioni multi-contenitore.

Nel sorgente del progetto è possibile ispezionare un file docker-compose.yml, il quale contiene la suddivisione in container del progetto. Inoltre nelle relative working directory è possibile visionare i vari Dockerfile contenti le specifiche e dipendenze di ogni container. Sono state ideate e containerizzate quattro componenti principali:

1. DB: container che rappresenta e istanzia il database contente il dataset complessivo del intero progetto. Con dati utili sia all'interfaccia del prodotto, sia alla componente di logica composta dall'algoritmo di raccomandazione;
2. Python-api: container che contiene l'algoritmo di raccomandazione, e le API, realizzate in Flask, per la comunicazione con le altre componenti;
3. Express: rappresenta le API utili all'interconnessione tra database ed interfaccia utente;
4. React-app: questo container contiene l'applicazione React e quindi l'interfaccia grafica del prodotto.

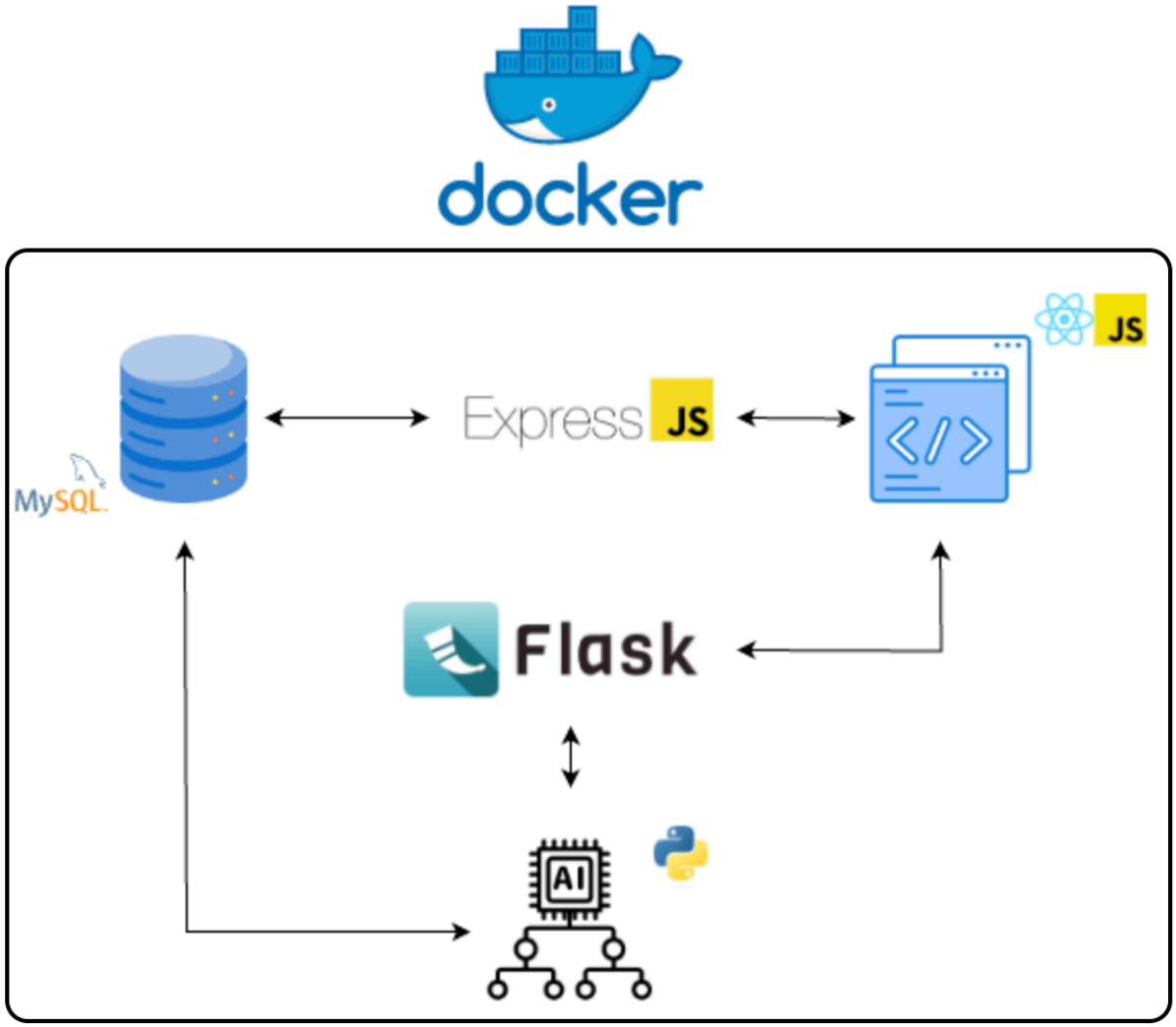

Figura 1: Docker environment

Il repository contiene inoltre le immagini di tutte le versioni del prodotto, disponibili per il download nella sezione package di GitHub.

3.2) Pattern architetturali - Architettura a Microservizi

L'architettura a microservizi presenta caratteristiche e crismi che si discostanto da quella meno recente ma comunque valida, monolitica. Quest'ultima è stata il paradigma dominante per lo sviluppo software per molti anni, soprattutto nelle prime fasi dello sviluppo di applicazioni web e enterprise.

In un'applicazione monolitica, l'intera applicazione è sviluppata, implementata e distribuita come un'unica entità. Tutte le funzionalità sono solitamente raggruppate all'interno di un singolo codice sorgente e eseguite all'interno di un unico processo.

Un'applicazione basata su microservizi è composta invece, come deducibile dalla nomenclatura, da molti piccoli servizi, ciascuno dei quali si occupa di una funzionalità specifica. Questi sono alcuni degli aspetti chiave che la contraddistinguono:

- Decomposizione modulare: l'applicazione viene scomposta in moduli autonomi e indipendenti, ognuno dei quali è un microservizio. Questi servizi possono essere sviluppati, testati e distribuiti separatamente;
- Indipendenza dei servizi: ogni microservizio è autosufficiente e può essere sviluppato, implementato e gestito in modo indipendente dagli altri. Ciò consente un rapido sviluppo e aggiornamento delle funzionalità senza influire sul resto dell'applicazione;
- Comunicazione tramite API: i microservizi comunicano tra loro attraverso interfacce di programmazione delle applicazioni (API), che possono essere sincrone o asincrone. Questo permette loro di cooperare e scambiare dati in modo efficiente;
- Scalabilità e resilienza: poiché i microservizi sono distribuiti, è possibile scalare e gestire le risorse in modo indipendente per ciascun servizio. Inoltre, se un microservizio fallisce, non compromette l'intera applicazione, ma solo la parte specifica che gestisce;
- Gestione dei dati: ogni microservizio può avere il proprio database, adatto alle sue esigenze specifiche. Questo favorisce una maggiore flessibilità nella scelta dei tipi di database e nella gestione dei dati.

Alcuni contro, che solitamente la caratterizzano sono invece:

- Complessità della gestione: gestire un ecosistema di microservizi richiede una maggiore complessità rispetto a un'applicazione monolitica. È necessario gestire la distribuzione, il monitoraggio, la scalabilità e la coordinazione dei servizi in modo accurato;
- Overhead di comunicazione: poiché i microservizi comunicano tra loro tramite API, può verificarsi un overhead di comunicazione, specialmente in sistemi distribuiti complessi. Questo può influire sulle prestazioni complessive dell'applicazione;
- Complessità dello sviluppo: lo sviluppo di un'applicazione basata su microservizi può essere più complesso rispetto a un'applicazione monolitica, poiché richiede una maggiore pianificazione e coordinazione tra i team di sviluppo. Inoltre, la gestione delle dipendenze tra i servizi può essere complicata;
- Consistenza dei dati: con i dati distribuiti tra diversi microservizi, garantire la coerenza e l'integrità dei dati può essere un compito complesso. È necessario implementare strategie di gestione dei dati distribuiti, come transazioni distribuite o modelli di consistenza eventualmente consistenti;

- Test e debugging: testare e debuggare un sistema basato su microservizi può essere più complesso rispetto a un'applicazione monolitica, poiché è necessario considerare le interazioni tra i diversi servizi e la loro integrazione complessiva.

Il gruppo, ha quindi decisio di adottare un'architettura a microservizi per lo sviluppo del prodotto pensando anche ad un possibile futuro Deployment.
La scelta di questa precisa architettura è derivata dalla natura ben separata e predefinita dei ruoli delle varie componenti del progetto, nonché dai numerosi pregi e benefici che ne derivano (come sopra elencati). Abbiamo pianificato di suddividere il sistema nel seguente modo:

- **Persistence logic**: Questa parte del sistema si occupa della gestione dei dati e della loro persistenza nel database MySQL. Il database contiene l'intero dataset necessario per il funzionamento delle altre parti del prodotto. La persistence logic definisce la struttura dei dati, le relazioni tra di essi e le regole di validazione dei dati. Inoltre, gestisce anche le operazioni di accesso al database, come la connessione e la gestione delle transazioni. È fondamentale che questa parte del sistema sia efficiente, affidabile e in grado di gestire grandi volumi di dati in modo sicuro.
- **Business logic**: Questa componente rappresenta il cuore del sistema, in quanto implementa le regole di business e la logica di elaborazione dei dati. Al centro di questa logica si trova l'algoritmo di raccomandazione, che analizza i dati presenti nel database e genera raccomandazioni personalizzate nelle varie modalità previste. L'infrastruttura di classi che compone l'algoritmo di raccomandazione definisce come vengono elaborati i dati, quali fattori vengono considerati nell'analisi e come vengono calcolate le raccomandazioni. Inoltre, questa parte del sistema gestisce eventuali operazioni di filtraggio, ordinamento o trasformazione dei dati in base alle esigenze specifiche del prodotto.
- **Application logic**: Questa parte del sistema gestisce l'interfaccia utente del prodotto, che è realizzata utilizzando React e JavaScript. Si occupa di presentare i dati e le funzionalità del sistema agli utenti in modo intuitivo e interattivo. Questa componente definisce la struttura e il comportamento delle pagine web inclusi layout, componenti, navigazione e gestione degli eventi utente. Utilizza le informazioni fornite dalla business logic per visualizzare i dati in modo appropriato e per consentire agli utenti di interagire con il sistema attraverso azioni come la ricerca, la selezione e l'inserimento di dati (feedback).

Come già descritto, i vari servizi comunicano tra loro tramite l'utilizzo di API REST realizzate ad hoc (descritte con maggiore dettaglio nella sezione apposita).

3.3) Persistence Logic

3.3.1) Introduzione

In questa sezione è possibile visionare tutte le scelte attuate durante la fase di progettazione e successivo sviluppo relative al database che si occupa della persistenza dei dati.

3.3.2) Schema base di dati

Il database che funge da Persistence Logic per il nostro prodotto è stato ideato e costruito a partire dal dataset fornitoci dall'azienda sotto forma di file csv. Alcune tabelle sono quindi state aggiunte successivamente per necessità del progetto, più precisamente: ute, ordclidet_feedback e cronologia. Si noti che per i nomi dei campi e delle tabelle sono stati utlizzati i nomi originali denominanti le colonne e i nomi degli stessi file csv per mantenere una coerenza tra il database del prodotto e quello utilizzato dal proponente. Data la grande mole di dati ed il formato del

dataset per l'inserimento iniziale è stata prediletta la modalità 'LOAD DATA INFILE'. Un'istruzione SQL utilizzata in MySQL per caricare dati da un file direttamente in una tabella del database. Questa istruzione è utile quando si desidera importare grandi quantità di dati da un file di testo delimitato direttamente nel database senza dover passare attraverso un processo di inserimento record per record. La tabella più corposa, quella degli ordini (ordclidet), ammonta a circa 2 milioni di record.

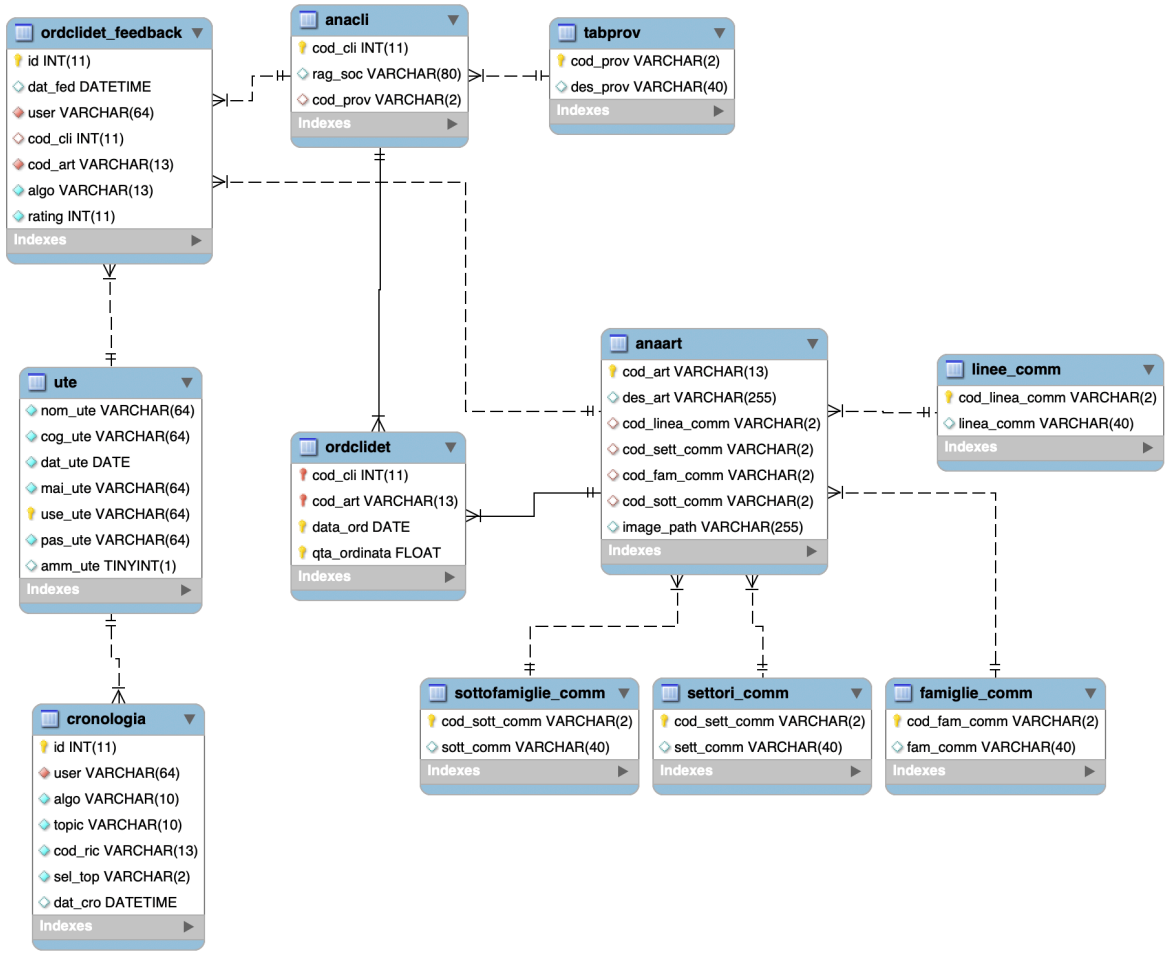

Figura 2: ER diagram

Descriviamo più nel dettaglio questa composizione:

ute			
Descrizione: Rappresenta i singoli utenti utilizzatori del prodotto			
Nome	**Tipo**	**Proprietà**	**Descrizione**
use_ute	VARCHAR(64)	PRIMARY KEY	L'username dell'utente
nom_ute	VARCHAR(64)	NOT NULL	Il nome dell'utente
dat_ute	DATE	NOT NULL	La data di nascita dell'utente

mai_ute	VARCHAR(64)	NOT NULL UNIQUE	La mail dell'utente
pas_ute	VARCHAR(64)	NOT NULL	La password criptata dell'utente
amm_ute	BOOLEAN	DEFAULT FALSE	Informazione se l'utente ha i permessi di amministratore

Tabella 6: Tabella ute

tabprov			
Descrizione: Rappresenta le provincie dei clienti			
Nome	**Tipo**	**Proprietà**	**Descrizione**
cod_prov	VARCHAR(2)	PRIMARY KEY	Il codice della provincia, intesa come sigla
des_prov	VARCHAR(40)		La descrizione della provincia, intesa come nome

Tabella 7: Tabella tabprov

anacli			
Descrizione: Rappresenta i clienti			
Nome	**Tipo**	**Proprietà**	**Descrizione**
cod_cli	INT	PRIMARY KEY	Il codice del cliente
rag_soc	VARCHAR(80)		La ragione sociale del cliente
cod_prov	VARCHAR(2)	FOREIGN KEY ON UPDATE CASCADE ON DELETE SET NULL	Il codice della provincia, intesa come sigla

Tabella 8: Tabella anacli

linee_comm			
Descrizione: Rappresenta le linee commerciali dei prodotti			
Nome	**Tipo**	**Proprietà**	**Descrizione**

cod_linea_comm	VARCHAR(2)	PRIMARY KEY	Il codice della linea commerciale
linea_comm	VARCHAR(40)		La descrizione della linea commerciale

Tabella 9: Tabella linee_comm

settori_comm			
Descrizione: Rappresenta i settori commerciali dei prodotti			
Nome	**Tipo**	**Proprietà**	**Descrizione**
cod_sett_comm	VARCHAR(2)	PRIMARY KEY	Il codice del settore commerciale
sett_comm	VARCHAR(40)		La descrizione del settore commerciale

Tabella 10: Tabella settori_comm

famiglie_comm			
Descrizione: Rappresenta le famiglie commerciali dei prodotti			
Nome	**Tipo**	**Proprietà**	**Descrizione**
cod_fam_comm	VARCHAR(2)	PRIMARY KEY	Il codice della famiglia commerciale
fam_comm	VARCHAR(40)		La descrizione della famiglia commerciale

Tabella 11: Tabella famiglie_comm

sottofamiglie_comm			
Descrizione: Rappresenta le sottofamiglie commerciali dei prodotti			
Nome	**Tipo**	**Proprietà**	**Descrizione**
cod_sott_comm	VARCHAR(2)	PRIMARY KEY	Il codice della sottofamiglia commerciale
sott_comm	VARCHAR(40)		La descrizione della sottofamiglia commerciale

Tabella 12: Tabella sottofamiglie_comm

anaart			
Descrizione: Rappresenta i singoli prodotti			
Nome	**Tipo**	**Proprietà**	**Descrizione**
cod_art	VARCHAR(13)	PRIMARY KEY	Il codice del prodotto
des_art	VARCHAR(255)		La descrizione del prodotto
cod_linea_comm	VARCHAR(2)	FOREIGN KEY ON UPDATE CASCADE ON DELETE SET NULL	Il codice della linea commerciale
cod_sett_comm	VARCHAR(2)	FOREIGN KEY ON UPDATE CASCADE ON DELETE SET NULL	Il codice del settore commerciale
cod_fam_comm	VARCHAR(2)	FOREIGN KEY ON UPDATE CASCADE ON DELETE SET NULL	Il codice della famiglia commerciale
cod_sott_comm	VARCHAR(2)	FOREIGN KEY ON UPDATE CASCADE ON DELETE SET NULL	Il codice della sottofamiglia commerciale
image_path	VARCHAR(255)		Il percorso dell'immagine del prodotto

Tabella 13: Tabella anaart

ordclidet			
Descrizione: Rappresenta i singoli ordini			
Nome	**Tipo**	**Proprietà**	**Descrizione**

cod_cli	INT	PRIMARY KEY FOREIGN KEY ON UPDATE CASCADE ON DELETE CASCADE	Il codice del cliente che ha effettuato l'ordine
cod_art	VARCHAR(13)	PRIMARY KEY FOREIGN KEY ON UPDATE CASCADE ON DELETE CASCADE	Il codice del prodotto appartenente all'ordine
data_ord	DATE	PRIMARY KEY	La data dell'ordine
qta_ordinata	FLOAT	PRIMARY KEY	La quantità di prodotto ordinata

Tabella 14: Tabella ordclidet

cronologia			
Descrizione: Rappresenta le singole ricerche effettuatuate da un utente			
Nome	**Tipo**	**Proprietà**	**Descrizione**
id	INT	PRIMARY KEY AUTO_INCREMENT	L'indentificativo di una ricerca
user	VARCHAR(64)	NOT NULL FOREIGN KEY ON UPDATE CASCADE ON DELETE CASCADE	L'utente che ha effettuato la ricerca
algo	VARCHAR(10)	NOT NULL	L'algoritmo utilizzato nella ricerca
topic	VARCHAR(10)	NOT NULL	Il topic della ricerca, inteso come ricerca per cliente o prodotto
cor_ric	VARCHAR(13)	NOT NULL	Il codice ricercato
sel_top	VARCHAR(2)	NOT NULL	Il numero di top risultati
dat_cro	DATETIME	DEFAULT NOW()	La data della ricerca

Tabella 15: Tabella cronologia

ordclidet_feedback			
Descrizione: Rappresenta il singolo feedback relativo ad un singolo risultato di ricerca			
Nome	**Tipo**	**Proprietà**	**Descrizione**
id	INT	PRIMARY KEY AUTO_INCREMENT	L'indentificativo di un feedback
dat_fed	DATETIME	DEFAULT NOW()	La data del feedback
user	VARCHAR(64)	NOT NULL FOREIGN KEY ON UPDATE CASCADE ON DELETE CASCADE	L'utente che ha effettuato il feedback
cod_cli	INT	FOREIGN KEY ON UPDATE CASCADE ON DELETE CASCADE	Il codice del cliente relativo
cod_art	VARCHAR(13)	FOREIGN KEY ON UPDATE CASCADE ON DELETE CASCADE	Il codice del prodotto relativo
algo	VARCHAR(10)	NOT NULL	L'algoritmo utilizzato nella ricerca relativa
rating	INT	NOT NULL DEFAULT 1	Il rating del feedback

Tabella 16: Tabella ordclidet_feedback

3.3.3) Query e indicizzazione

Si riportano qui sotto alcune delle query, principalmente quelle che riteniamo essere più interessanti, eseguite:

- Selezione di tutti i prodotti in ordine di codice (e relative caratteristiche):

```
SELECT * FROM anaart LEFT JOIN linee_comm ON anaart.cod_linea_comm = linee_comm.cod_linea_comm LEFT JOIN settori_comm ON anaart.cod_sett_comm = settori_comm.cod_sett_comm LEFT JOIN famiglie_comm ON anaart.cod_fam_comm = famiglie_comm.cod_fam_comm LEFT JOIN sottofamiglie_comm ON anaart.cod_sott_comm = sottofamiglie_comm.cod_sott_comm ORDER BY cod_art ;
```

- Selezione di tutti i clienti in ordine di codice (e relative caratteristiche):

```
SELECT * FROM anacli LEFT JOIN tabprov ON anacli.cod_prov = tabprov.cod_prov ORDER BY cod_cli ;
```

- Selezione di tutti i feedback (ordinati in modo inverso rispetto alla data di inserimento):

```
SELECT id, dat_fed, user, cod_cli, cod_art, algo FROM ordclidet_feedback ORDER BY dat_fed ASC ;
```

- Selezione di tutte le ricerche (ordinata in modo inverso rispetto alla data di ricerca):

```
SELECT user, algo, topic, cod_ric, sel_top, dat_cro FROM cronologia ORDER BY dat_cro ASC ;
```

- Inserimento di una nuova ricerca nella tabella cronologia:

```
INSERT INTO cronologia (user, algo, topic, cod_ric, sel_top) VALUES (?,?,?,?,?) .
```

Inoltre sono stati previsti degli indici per l'indicizzazione ed ottimizzaione delle operazioni:

- `CREATE INDEX idx_cod_prov ON anacli(cod_prov);`
- `CREATE INDEX idx_cod_linea_comm ON anaart(cod_linea_comm);`
- `CREATE INDEX idx_cod_sett_comm ON anaart(cod_sett_comm);`
- `CREATE INDEX idx_cod_fam_comm ON anaart(cod_fam_comm);`
- `CREATE INDEX idx_cod_sott_comm ON anaart(cod_sott_comm);`
- `CREATE INDEX idx_cod_cli ON ordclidet(cod_cli);`
- `CREATE INDEX idx_cod_art ON ordclidet(cod_art);`
- `CREATE INDEX idx_user ON cronologia(user);`
- `CREATE INDEX idx_dat_cro ON cronologia(dat_cro).`

Abbiamo deciso di predisporre degli indici per le tabelle di maggiori dimensioni, pensando anche a future possibili implementazioni, andandone a valutare l'integrazione o meno utilizzando il comando sql EXPLAIN, utile per ottenere informazioni sul piano di esecuzione delle varie query.

3.4) Business Logic

3.4.1) Introduzione

In questa sezione è possibile visionare tutte le scelte attuate durante la fase di progettazione e successivo sviluppo relative al codice che compone la Business logic del prodotto.

3.4.1.1) Diagramma delle classi

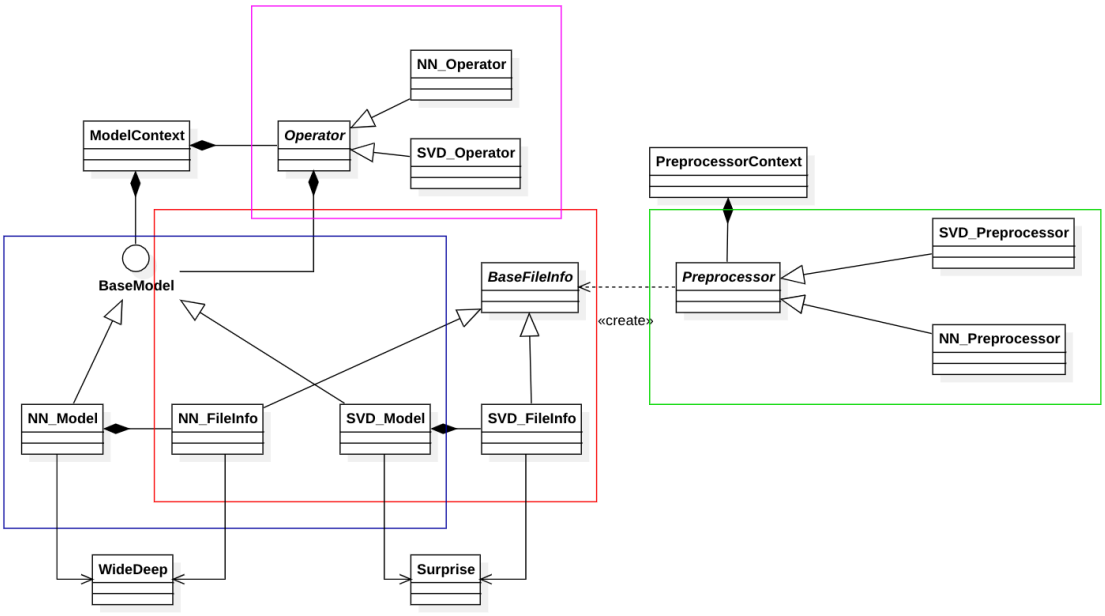

Figura 3: Diagramma algoritmo (totale)

Descrizione:
Nel diagramma sopra riportato è possibile esaminare nel totale la struttura di classi che costituisce la Business Logic del prodotto, ciascuna "componente" verrà analizzata in dettaglio di seguito. Abbiamo deliberatamente optato per l'utilizzo di Python come linguaggio di programmazione orientato agli oggetti, implementando una struttura basata su classi. Questa scelta è stata guidata da diversi fattori che includono la volontà di garantire una maggiore modularità nel nostro sistema. L'approccio orientato agli oggetti favorisce la riusabilità del codice, consentendo la definizione di classi e metodi che possono essere riutilizzati in diverse parti del progetto. Inoltre, l'incapsulamento dei dati e dei comportamenti all'interno delle classi contribuisce a garantire l'integrità del sistema, limitando l'accesso diretto agli attributi e ai metodi. Questa progettazione modulare e organizzata facilita l'estensibilità del sistema, consentendo l'aggiunta di nuove funzionalità senza dover modificare il codice esistente. Il nostro obiettivo primario, sin dall'inizio del progetto, è stato e rimane quello di garantire flessibilità e manutenibilità nel tempo. Pertanto, anche se inizialmente non contemplato, abbiamo deciso di sviluppare una struttura in grado di accogliere e gestire più strategie e algoritmi di raccomandazione. È evidente una suddivisione in quattro principali "componenti", studiata appositamente per assicurare una chiara separazione delle diverse responsabilità e funzionalità del sistema.

Pattern:

- **Dependency Injection**: design pattern utilizzato principalmente nella programmazione orientata agli oggetti per gestire le dipendenze tra gli oggetti. In breve, la dependency injection prevede che gli oggetti esterni siano responsabili di fornire le dipendenze di un oggetto anziché sia l'oggetto stesso a crearle internamente.
  - Constructor Injection:
    In questa prima casistica le dipendenze vengono dichiarate come parametri direttamente nel costruttore. Il vantaggio principale è quello di poter necessariamente creare oggetti in modo immediato, rendendoli contemporaneamente immutabili;

- Setter Injection:
  Questo approccio prevede invece la dichiarazione delle dipendenze come metodi setter. Ciò si sposa meglio con la presenza di gerarchie tra classi, permettendo una maggiore flessibilità durante l'escuzione ed una generale riduzione dell'accoppiamento.

Applicazione ed uso:
Nel contesto del nostro progetto, la seconda derivazione di questo pattern, ci ha permesso di garantire una maggiore flessibilità tra le parti ed è principalmente per questo che è stato favorito alla sua controparte nonostante i suoi difetti. Siamo consapevoli che questo approccio possa rendere gli oggetti più soggetti a stati inconsistenti se non vengono correttamente inizializzati tramite i setter. Abbiamo quindi riposto attenzione su quest'ultimo aspetto.

- **Strategy**: design pattern comportamentale che consente di definire una famiglia di algoritmi, incapsularli e renderli intercambiabili. In pratica, si definiscono più algoritmi all'interno di classi separate, ciascuna delle quali rappresenta una strategia specifica. Questo pattern promuove la modularità, l'estensibilità e la manutenibilità del codice, in quanto consente di separare gli algoritmi dalle classi client e di modificare o aggiungere nuove strategie senza dover modificare il codice di queste ultime.

Applicazione ed uso:
Le caratteristiche principali di questo pattern l'hanno reso il candidato ideale per applicabilità all'interno del nostro progetto. L'utilizzo di questo pattern, ed in generale l'uso dell'ereditarietà tra classi, ci hanno permesso di rendere il codice più flessibile ed estensibile, prerogative che fin da subito abbiamo fissato come fondamentali all'interno del nostro progetto.

3.4.1.2) Componenti:

3.4.1.2.1) Preprocessor

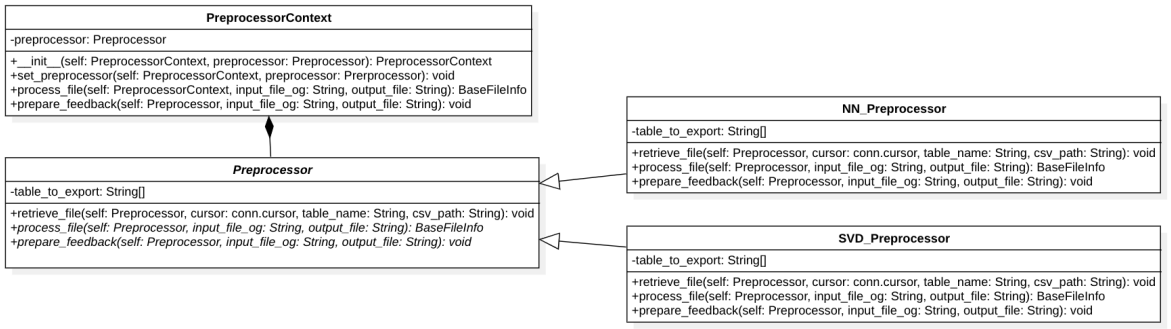

Figura 4: Results

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 ereditano dalla classe Preprocessor le funzionalità di base e forniscono implementazioni specifiche per il preprocessing dei dati utilizzando diverse modalità. Ogni preprocessor attua della azioni di preprocessing dei dati in base all'algoritmo di raccomandazioni che ne andrà poi ad usufruire. 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' : metodo che prende in input 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 scrivere i risultati in un file CSV;
  2. '*process_file*' : metodo astratto che prende in input 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 prestabilito;
  3. '*prepare_feedback*' : metodo astratto simile a process_file, ma specificamente progettato per preparare i dati di feedback i quali richiedono una diversa elaborazione.
- PreprocessorContext:
  1. 'set_preprocessor' : metodo che imposta il preprocessor da utilizzare;
  2. 'process_file' : metodo che prende in input 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.

3.4.1.2.2) FileInfo

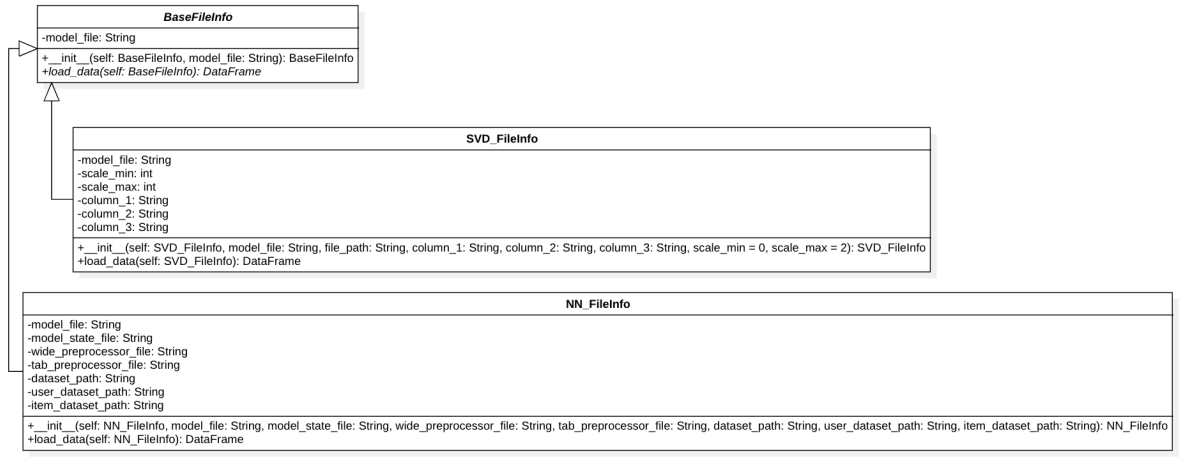

Figura 5: FileInfo

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 sotto-classi 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*' : 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, utilmente 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, utilmente per modelli basati su decomposizione singolare che operano su dati in formato tabellare.

3.4.1.2.3) Model

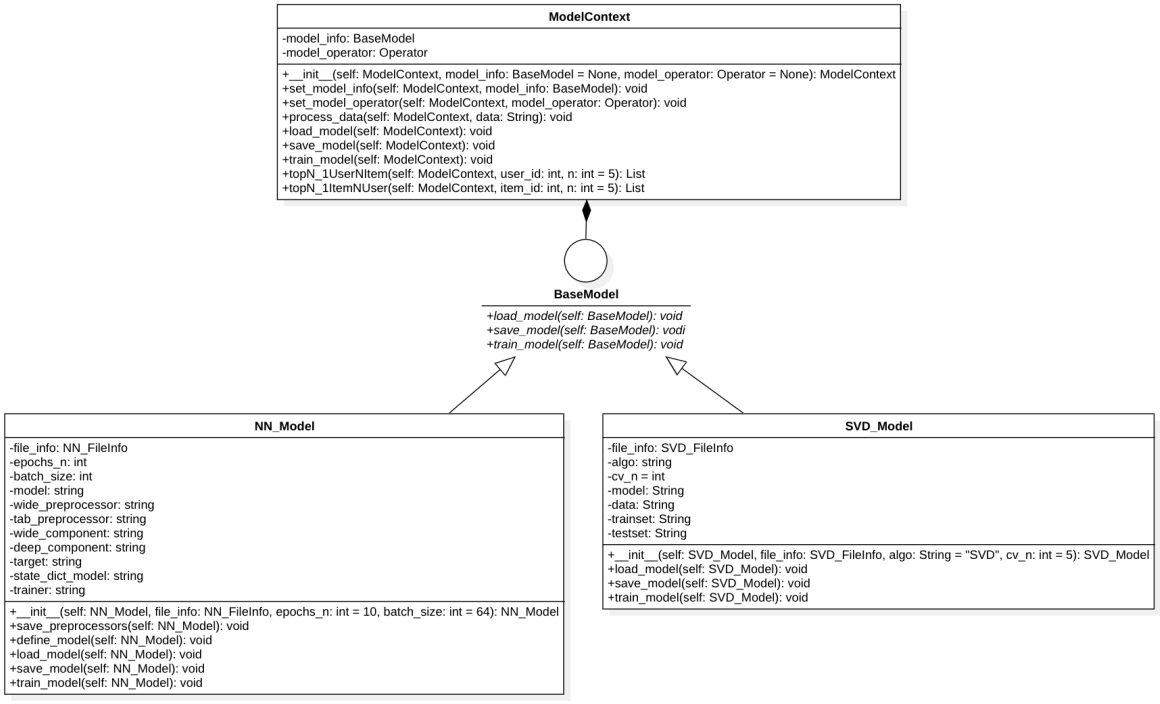

Figura 6: Model

Descrizione:
BaseModel è un *Interfaccia* 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 nel rispettivo 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 preaddestrato 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: (Metodi relativi a BaseModel)
  1. 'set_model_info' : questo metodo consente di impostare le informazioni sul modello (model_info) dell'oggetto ModelContext. Accetta un argomento model_info di tipo BaseModel, che viene quindi assegnato all'attributo model_info;
  2. '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;
  3. 'load_model' : invoca il metodo corrispettivo in base al tipo di BaseModel che contiene.
  4. 'save_model' : invoca il metodo corrispettivo in base al tipo di BaseModel che contiene.
  5. 'train_model' : invoca il metodo corrispettivo in base al tipo di BaseModel che contiene.

3.4.1.2.4) Operator

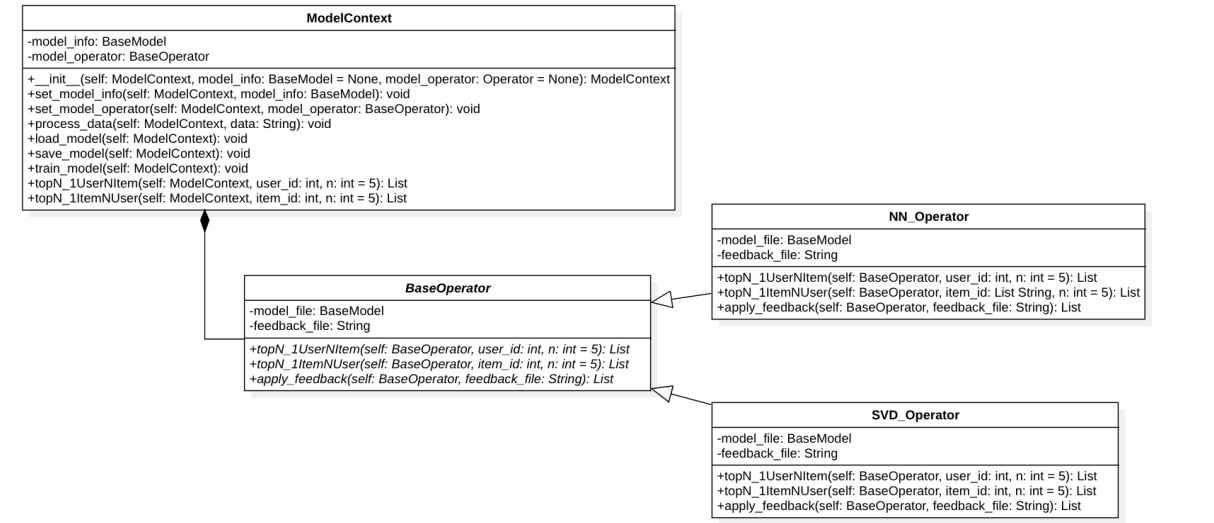

Figura 7: Operator

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 applicando anche conversioni di scala preservando il valore del voto;
  2. '*apply_feedback*' : metodo astratto che si occupa di applicare il feedback ricevuto (su utenti o prodotti) ai rating previsti dal modello;
  3. '*topN_1UserNItem*' : metodo astratto che restituisce i migliori N prodotti per un dato utente in base alle previsioni del modello;
  4. '*topN_1ItemNUser*' : metodo astratto che restituisce i migliori N utenti per un dato prodotto 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 utilizzando la scala appropriata;
  2. 'apply_feedback' : implementazione di 'apply_feedback' di BaseOperator, applica il feedback ricevuto (su utenti o prodotti) ai rating previsti dal modello;
  3. 'topN_1UserNItem' : implementazione di 'topN_1UserNItem' di BaseOperator, restituisce i migliori N prodotti per un dato utente in base alle previsioni del modello (NN);

4. ‘topN_1ItemNUser’ : implementazione di ‘topN_1ItemNUser’ di BaseOperator, restituisce i migliori N utenti per un dato prodotto in base alle previsioni del modello (NN).

- 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 utilizzando la scala appropriata;
  2. ‘apply_feedback’ : implementazione di ‘apply_feedback’ di BaseOperator, applica il feedback ricevuto (su utenti o prodottis) ai rating previsti dal modello;
  3. ‘topN_1UserNItem’ : implementazione di ‘topN_1UserNItem’ di BaseOperator, restituisce i migliori N prodotti 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 prodotto in base alle previsioni del modello.

  ▸ ModelContext: (Metodi relativi a BaseOperator)
  1. ‘topN_1UserNItem’ : invoca il metodo corrispettivo in base al tipo di BaseOperator che contiene;
  2. ‘topN_1ItemNUser’ : invoca il metodo corrispettivo in base al tipo di BaseOperator che contiene.

3.4.1.2.5) Librerie esterne

Descrizione:
Per l’implementazione dei due algoritmi di raccomandazioni presenti sono state utilizzate due librerie di python già citate all’interno del documento. Per quanto riguarda l’algoritmo SVD è stata utilizzata la libreria Surprise individuata fin da subito dal proponente. L’implementazione della rete neurale si è invece poggiata su una sotto libreria di PyTorch, una delle librerie più conosciute nell’ambito, denominanta widedeep.

3.5) Application Logic

In questa sezione è possibile visionare tutte le scelte attuate durante la fase di progettazione e successivo sviluppo relative al codice che compone l’Application Logic del prodotto.

3.5.1) 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.

3.5.2) Pagine

3.5.2.1) Clienti

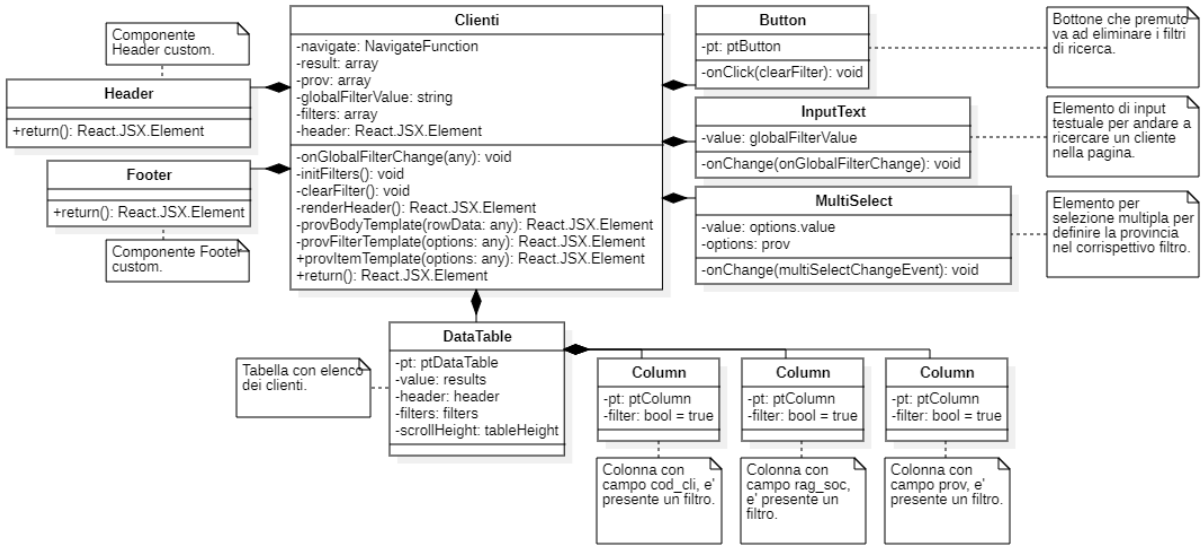

Figura 8: Clienti

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.

3.5.2.2) Cronologia

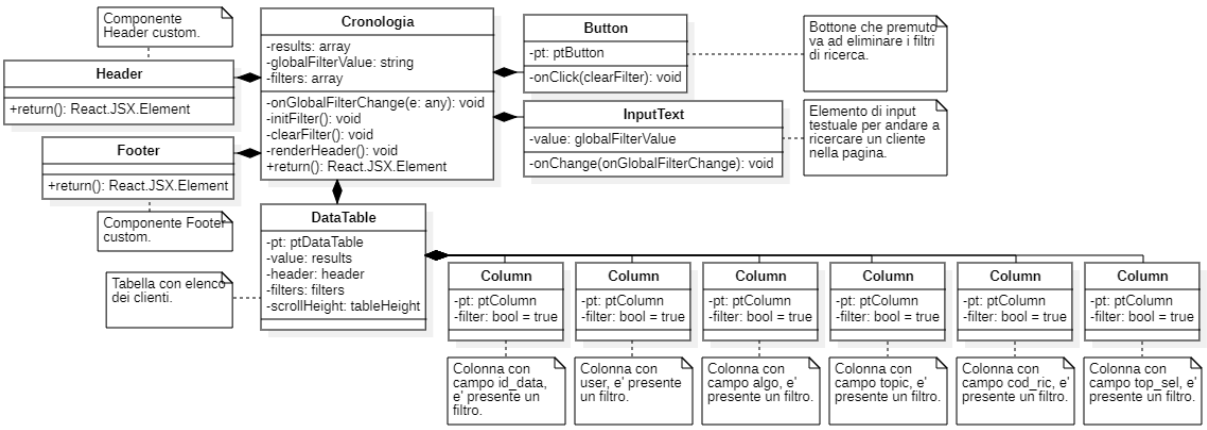

Figura 9: Cronologia

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.

3.5.2.3) Feedback

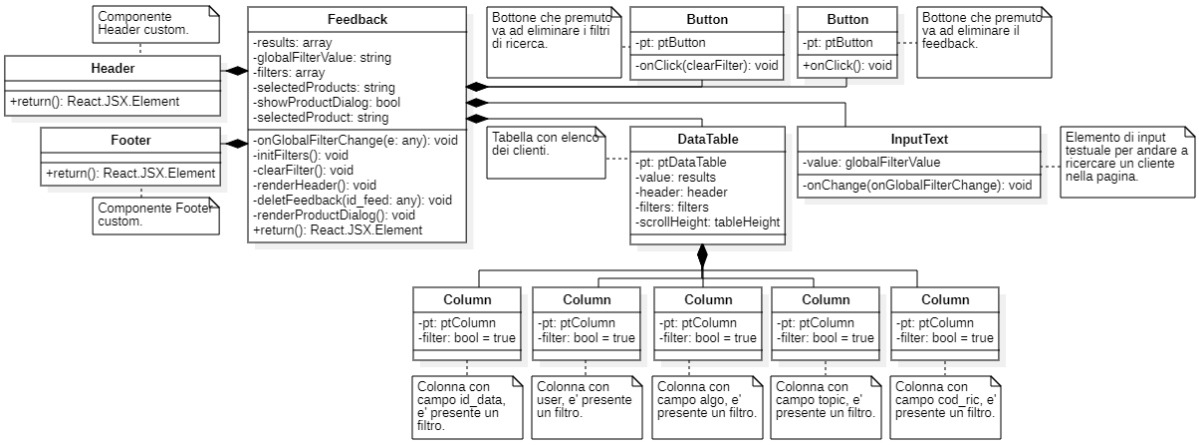

Figura 10: Feedback

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.

3.5.2.4) Login

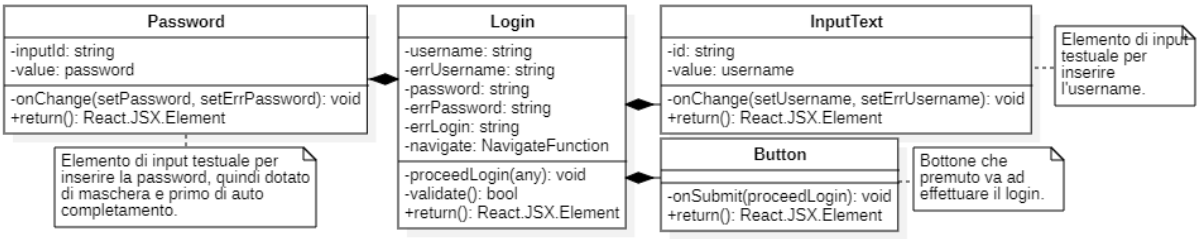

Figura 11: Login

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.

3.5.2.5) Pagina Non Trovata

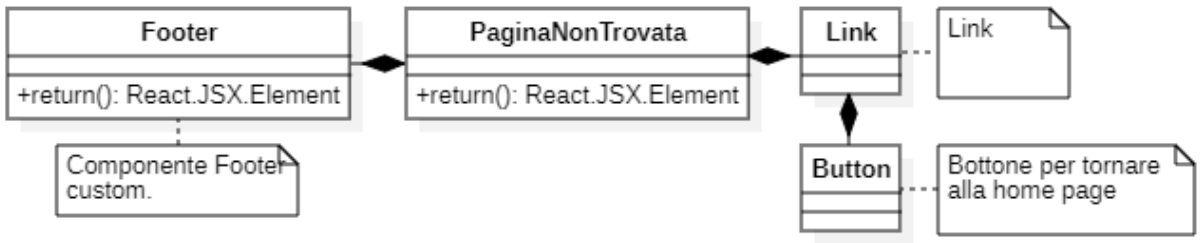

Figura 12: Pagina Non Trovata

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.

3.5.2.6) Prodotti

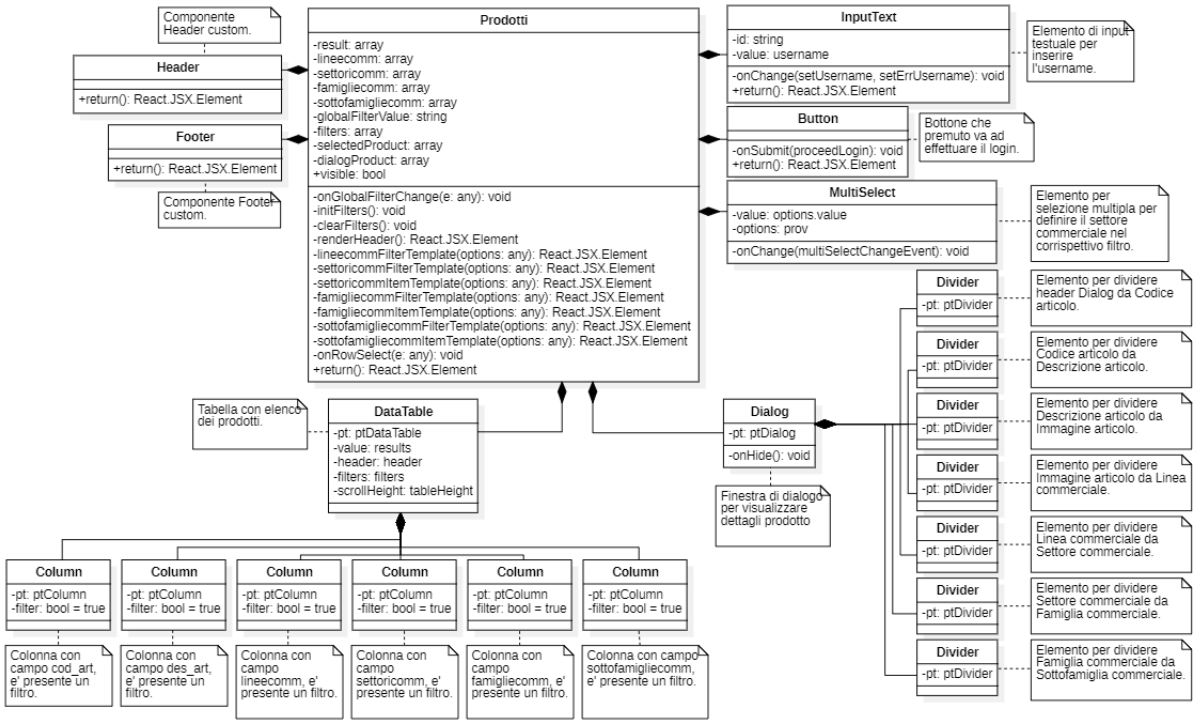

Figura 13: Prodotti

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.

3.5.2.7) Profilo

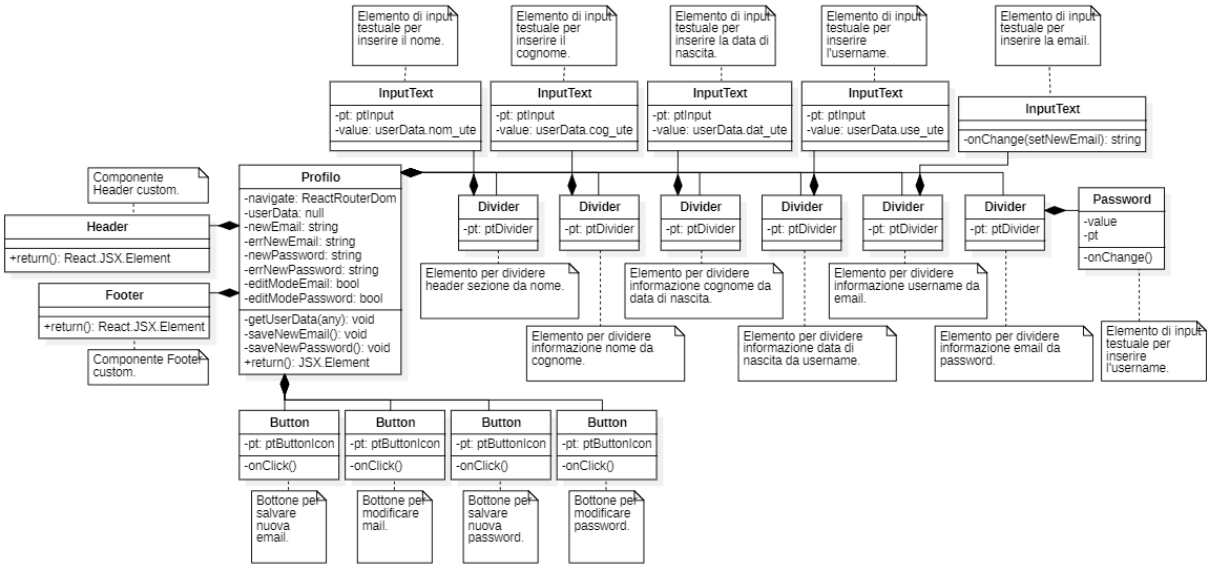

Figura 14: Profilo

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.

3.5.2.8) Ricerca

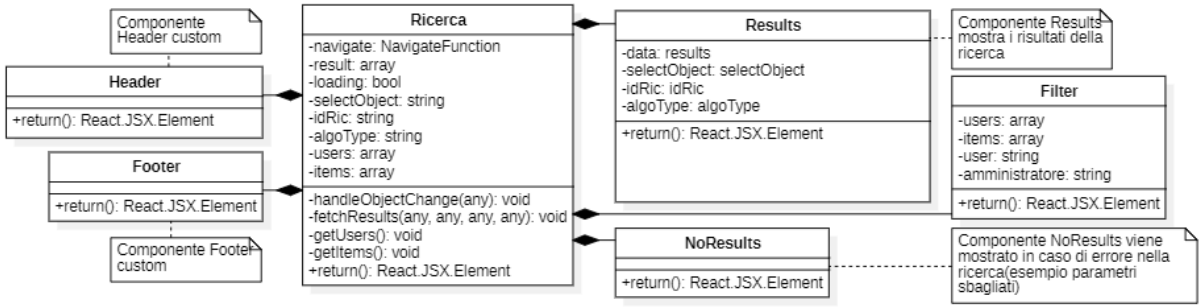

Figura 15: Ricerca

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*.

3.5.3) Componenti

3.5.3.1) Filtri

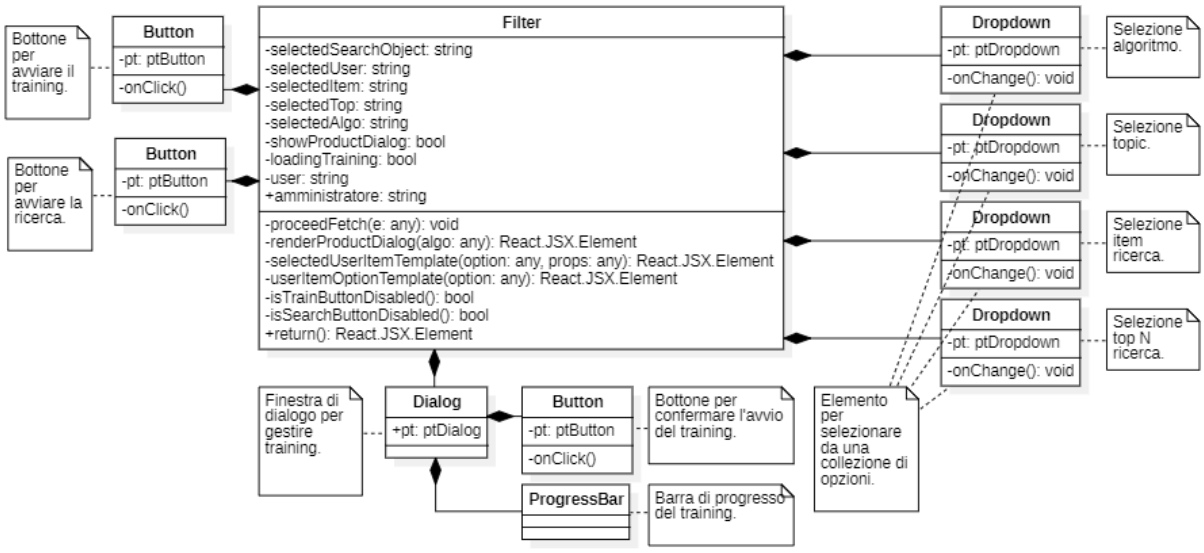

Figura 16: Filtri

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.

3.5.3.2) Footer

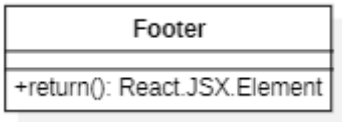

Figura 17: Footer

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

3.5.3.3) Header

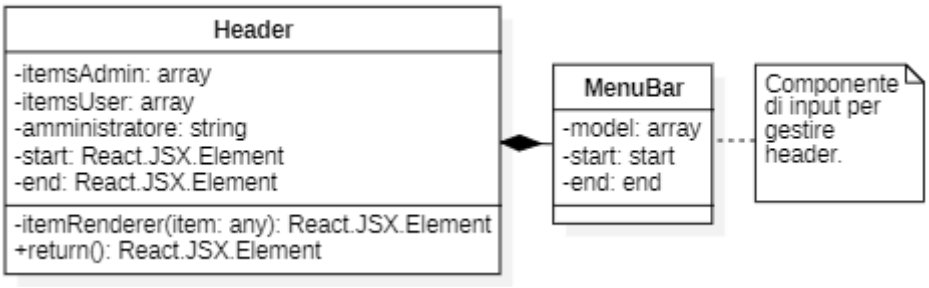

Figura 18: Header

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

3.5.3.4) No Results

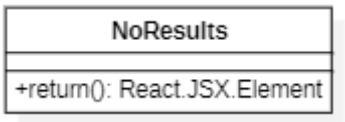

Figura 19: No Results

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

3.5.3.5) Results

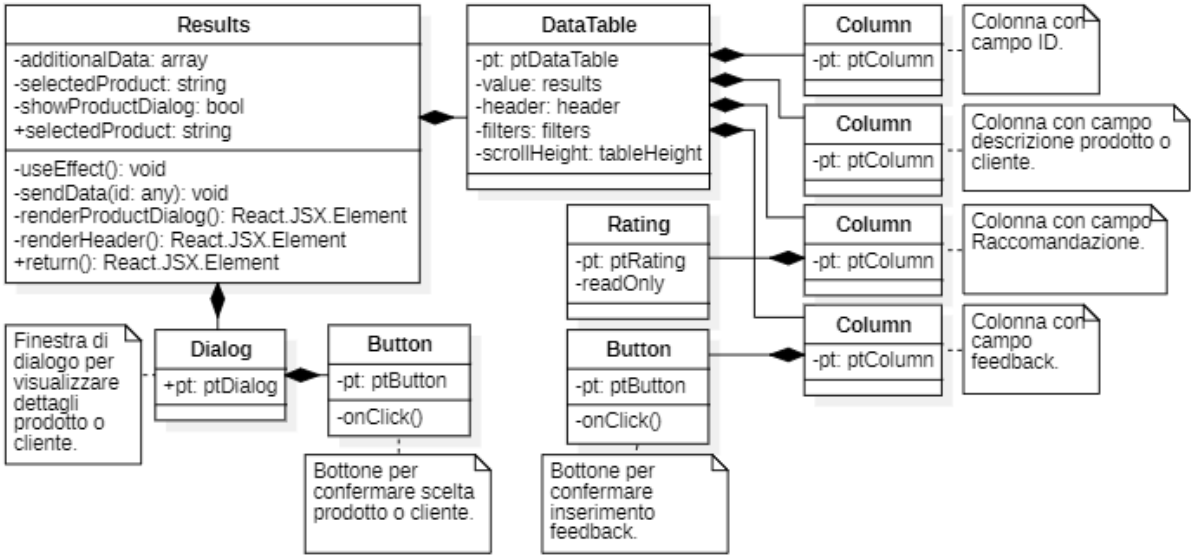

Figura 20: Results

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.

3.5.4) Documentazione API

Per il nostro progetto abbiamo utilizzato diversi tipi di API:
- Per metodi GET con Express;
- Per metodi GET con Flask (python);
- Per metodi PUT con Express;
- Per metodi POST con Flask (python).

L'utilizzo di Flask e le relative API riguardano la parte dell'algoritmo, training (POST) e ricerca (GET), mentre la parte di Express riguarda l'interfaccia web.

3.5.4.1) 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 17: 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 18: 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 19: 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 20: 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 21: 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 22: 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 23: 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 24: 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 25: 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 26: 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 27: 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 28: 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 29: 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 30: 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 31: Esito della richiesta di recupero dei feedback degli ordini dei clienti.

16. /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 32: Esito della ricerca utilizzando un algoritmo specifico.

3.5.4.2) 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 33: 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 34: 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 35: 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 36: 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 37: 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 38: 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 39: Esito dell'aggiornamento della password di un utente.

3.5.4.3) Chiamate POST

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 40: Esito dell'addestramento di un algoritmo di machine learning.

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 l'autenticazione fallisse, è necessario che l'utente riceva un messaggio con dettagli che ne indicano il motivo.	Soddisfatto
RDF 5	Nel caso un utente autenticato cercasse di entrare all'interno di pagine del sito di cui non ha l'autorizzazione, è necessario che venga indirizzato alla pagina 404.	Soddisfatto
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.	Soddisfatto
ROF 9	L'utente che ha scelto di modificare i dati utente, deve poter modificare l'email e la password.	Soddisfatto
RDF 10	L'utente nel caso inserisca dei nuovi dati inadeguati, visualizza un messaggio di errore	Soddisfatto
ROF 11	L'utente inserisce la nuova email nel campo email, per procedere a modificare i dati.	Soddisfatto

ROF 12	L'utente inserisce la nuova password nel campo password, per procedere a modificare i dati.	Soddisfatto
ROF 13	L'utente, una volta autenticato, deve poter effettuare il Logout tramite il pulsante presente nella pagina principale del sito.	Soddisfatto
ROF 14	L'utente, una volta autenticato, deve poter accedere alla funzione "Ricerca" nella pagina principale del sito.	Soddisfatto
ROF 15	L'utente una volta entrato nella sezione "Ricerca", deve poter effettuare una ricerca, filtrarne i risultati, visualizzarne i risultati e eseguire il training dell'algoritmo.	Soddisfatto
ROF 16	L'utente che ha scelto di effettuare una ricerca, deve compilare tutti i campi per effettuarla e poter visualizzare i risultati.	Soddisfatto
RDF 17	L'utente che compila la scelta dell'algoritmo, può scegliere tra SVD e NN.	Soddisfatto
ROF 18	L'utente che compila la scelta del topic della ricerca, può scegliere tra prodotto per cliente o cliente per prodotto, per poi compilare i successivi campi.	Soddisfatto
ROF 19	L'utente che compila la scelta degli "N risultati", può scegliere tra i 5 migliori risultati (Top 5), tra i migliori 10 (Top 10) o i migliori 20 (Top 20).	Soddisfatto
RDF 20	L'utente che ha eseguito la ricerca, può filtrarne i risultati delle colonne con i risultati.	Non soddisfatto
ROF 21	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 22	L'utente può decidere se avviare un training del modello tramite il pulsante Training nella barra di ricerca	Soddisfatto
RDF 23	Nel caso un'utente cercasse di eseguire una ricerca con l'algortimo in training, deve visualizzare un messaggio di avviso.	Soddisfatto
RDF 24	Nel caso la ricerca non andasse a buon fine, l'utente deve visualizzare un messaggio di errore che indica	Soddisfatto

	che la ricerca non è terminata correttamente. Il messaggio di errore deve essere mostrato in caso di errore anche per le ricerche di RDF 29, RDF 35, RDF 39, RDF 43.	
ROF 25	L'utente che ha visualizzato i risultati della ricerca, deve poter inserire un feedback delle raccomandazioni mostrate.	Soddisfatto
RDF 26	L'utente, una volta autenticato, deve poter accedere alla funzione "Catalogo Prodotti" nella pagina principale del sito.	Soddisfatto
RDF 27	L'utente una volta entrato nella sezione "Catalogo Prodotti", deve poter effettuare un filtraggio e visualizzarne i risultati.	Soddisfatto
RDF 28	L'utente che ha scelto di effetuare un fltraggio può compilare i campi filtri che desidera.	Soddisfatto
RDF 29	L'utente che ha effettuato un filtraggio e non, deve poter visualizzare tutti i campi della tabella con i risultati: Codice Articolo, Descrizione Articolo, Linea commerciale, Settore commerciale, Famiglia commersciale e Sotto-famiglia commerciale.	Soddisfatto
RDF 30	L'utente che ha visualizzato i risultati della ricerca, può visualizzare i dettagli di un prodotto, cliccando sulla riga corrispondente al prodotto che vuole visualizzare i dettagli.	Soddisfatto
RDF 31	L'utente se decide di visualizzare i dettagli di un prodotto, deve poter vedere il Codice e descrizione articolo, immagine articolo, Codice e descrizione Linea commerciale, Codice e descrizione Settore commerciale, Codice e descrizione Famiglia commerciale e Codice e descrizione Sotto-famiglia commerciale.	Soddisfatto
RDF 32	L'utente, una volta autenticato, deve poter accedere alla funzione "Lista clienti" nella pagina principale del sito.	Soddisfatto
RDF 33	L'utente una volta entrato nella sezione "Lista clienti", deve poter effettuare un filtraggio e visualizzarne i risultati.	Soddisfatto

RDF 34	L'utente che ha scelto di effettuare un filtraggio, può compilare i campi Codice cliente, Ragione sociale e Provincia.	Soddisfatto
RDF 35	L'utente che effettutato un filtraggio e non, deve poter visualizzare il Codice cliente, la Ragione sociale e la provincia del cliente	Soddisfatto
RDF 36	L'utente, una volta autenticato, deve poter accedere alla funzione "Cronologia" nella pagina principale del sito.	Soddisfatto
RDF 37	L'utente una volta entrato nella sezione "Cronologia", deve poter effettuare una filtraggio e visualizzarne i risultati.	Soddisfatto
RDF 38	L'utente che ha scelto di effettuare un filtraggio, può compilare i campi Data, Utente, Algoritmo, Topic, Codice Cliente/Prodotto e Top.	Soddisfatto
RDF 39	L'utente che effettutato un filtraggio e non, deve poter visualizzare la data, l'utente l'algoritmo, il topic, il codice Cliente/Prodotto e il top.	Soddisfatto
RDF 40	L'utente, una volta autenticato, deve poter accedere alla funzione "Feedback" nella pagina principale del sito.	Soddisfatto
RDF 41	L'utente una volta entrato nella sezione "Feedback", può effettuare un filtraggio e visualizzarne i risultati.	Soddisfatto
RDF 42	L'utente che ha scelto di effettuare un filtraggio, deve compilare i campi Data, Utente, ID Cliente, ID Prodotto, Algoritmo.	Soddisfatto
RDF 43	L'utente che effettutato un filtraggio e non, deve poter visualizzare la data, l'utente, l'ID cliente, l'ID prodotto e l'algoritmo del Feedback	Soddisfatto
RDF 44	L'utente, una volta autenticato, deve poter accedere alla funzione "Carica dataset" e caricare un dataset esterno all'interno dell'applicazione.	Non soddisfatto
RDF 45	L'utente, se ha caricato un dataset esterno, dever poter avviare il training del dataset in maniera da poterlo usare per le raccomandazioni.	Non soddisfatto

RDF 46	L'utente, una volta autenticato, deve poter visualizzare la funzione statistiche mensili nel menù principale.	Non soddisfatto
RDF 47	L'utente una volta dentro la funzionalità statistiche mensili, deve poterne visualizzare il grafico e le raccomandazioni più attendibili.	Non soddisfatto
RDF 48	L'utente se visualizza il grafico, deve vedere sull'asse delle x le vendite mensili e sull'asse delle y le raccomandazioni proposte.	Non soddisfatto
RDF 49	L'utente se vsualizza le raccomandazioni più attendibili, deve poter vedere l'ID Cliente, l'ID prodotto della raccomandazione.	Non soddisfatto
RDF 50	L'utente se vuole effettuare una ricerca tramite la vista "Ricerca", la può effettuare tramite la cronologia delle ricerche precedenti.	Non soddisfatto
RDF 51	L'utente se esegue una ricerca tramite la cronologia deve selezionare la riga con la ricerca all'interno delle cronologia.	Non soddisfatto

Tabella 1: Requisiti funzionali

4.2) Grafico requisiti funzionali

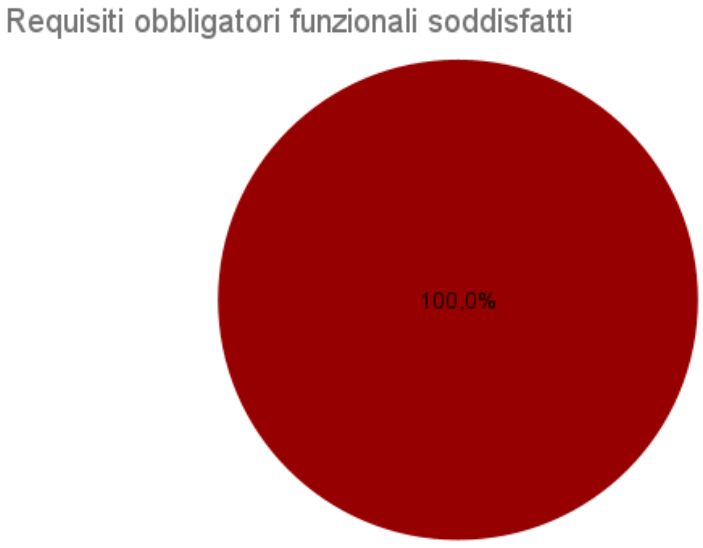

Figura 21: Requsiti obbligatori funzionali soddisfatti

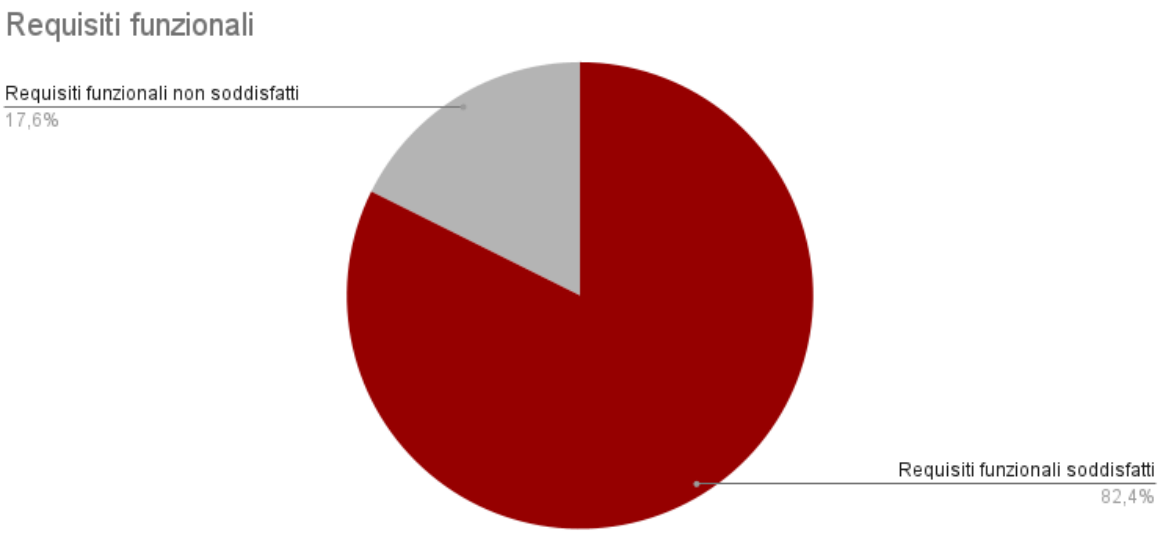

Figura 22: Requisiti funzionali

5) Elenco delle immagini

6) Elenco delle tabelle